N° 075-2003 Year 2003

Doctor thesis (translated from French)

UNIVERSITÉ CLAUDE BERNARD - LYON 1, FRANCE
20 June 2003

Martin JAMBON

A bioinformatic system for searching functional similarities in 3D structures of proteins

(*Un système bioinformatique de recherche de similitudes fonctionnelles dans les structures 3D de protéines*)

Directeur de thèse:	Dr Christophe Geourjon
Jury:	Pr Alain-Jean Cozzone, Président
	Pr Alexander Bockmayr, Rapporteur
	Pr Joël Janin, Rapporteur
	M François Delfaud
	Dr Christophe Geourjon
	Dr Serge Pérez

Abstract

SuMo is a bioinformatic system for comparing 3D structures of proteins. This approach was designed to help along the exploration of structural data by biologists and the formulation of accurate hypotheses concerning the biological implication of proteins.

As opposed to existing approaches in this field, SuMo does not solve a formal problem that would derive from a model for biological function. This allows constant development of the heuristics, by getting closer to intuitive concepts for biological function rather than stick to a mathematical model with poor relevance.

The heuristics that has been developed is based on a representation of macromolecules using chemical groups that are associated with heterogeneous geometrical properties and grouped into triplets.

SuMo can be used directly from the web server `http://sumo-pbil.ibcp.fr`. Screening SuMo's database of ligand binding sites allows in a convenient way to discover potential therapeutic targets.

Résumé

SuMo est un système bioinformatique de comparaison de structures 3D de protéines. Le but de cette approche est de faciliter l'exploration des données structurales par les biologistes et la formulation d'hypothèses fines sur l'implication biologique des protéines.

Contrairement aux approches existantes dans ce domaine, SuMo ne s'attache pas à résoudre un problème formel dérivant d'une modélisation de la notion de fonction biologique. Ceci autorise des efforts constants de développement de l'heuristique mise en oeuvre, permettant de se rapprocher des notions intuitives de fonction biologique plutôt que de coller à un modèle mathématique peu réaliste.

L'heuristique développée est basée sur la représentation des macromolécules par des groupements chimiques aux propriétés géométriques hétérogènes et associés en triplets.

L'utilisation de SuMo s'effectue directement à partir du serveur web `http://sumo-pbil.ibcp.fr`. Le criblage de la banque de sites de fixation de ligands générée par et pour SuMo permet de repérer de façon conviviale de potentielles cibles thérapeutiques.

Contents

List of Figures **11**

List of Tables **13**

List of algorithms **15**

1 Introduction **17**

2 Scientific and methodological context **21**

- 2.1 Molecular and cellular bioinformatics 21
- 2.2 Structure-function relation in proteins 23
  - 2.2.1 General remarks 23
    - 2.2.1.1 Notion of site 23
    - 2.2.1.2 Competition for interactions 23
  - 2.2.2 Search for functional sites using comparisons 24
    - 2.2.2.1 From the amino acid sequence 25
      - Alignment of homologous sequences 26
      - Searching patterns 26
    - 2.2.2.2 From the 3D structure 26
      - Modeling protein surface 27
      - Modeling using point objects 27
- 2.3 Programming tools 29
  - 2.3.1 General-purpose programming language 30
  - 2.3.2 Specialized syntaxes 30
  - 2.3.3 Storing data 30
  - 2.3.4 Communication 31

3 Description of the SuMo system **33**

- 3.1 General architecture 34
  - 3.1.1 The different levels 34
    - 3.1.1.1 Lower level: programming language 34

3.1.1.2 Medium level: SuMo language 36
3.1.1.3 Higher level: SuMoQ comparison queries . . . 37
3.1.2 SuMo from different points of view 38
3.1.2.1 A user's point of view 38
3.1.2.2 The administrator's point of view 40
3.1.2.3 The programmer's point of view 41
3.2 Pairwise comparison of 3D structures 41
3.2.1 Chemical groups 44
3.2.1.1 Preliminaries 44
Identification of the molecules 44
Detecting hydrogen bonds 46
Definition of ligand 47
3.2.1.2 Chemical groups 48
Types 48
Coefficients 48
Positional informations 48
Shared additional data 49
Type-specific data 50
Annotation of chemical groups 52
Syntax for defining types of chemical groups . . 53
Phantom chemical groups 55
3.2.2 Association into triplets 55
3.2.2.1 Choosing the triplets 56
Length of edges 56
Sum of edge length 57
Angles 57
3.2.2.2 Properties of the triplets 57
Triplet coordinate system 58
Properties from chemical groups 58
Length of edges 59
Orientation against the molecular surface 59
3.2.3 The graph of triplets 59
3.2.3.1 Vertices 59
3.2.3.2 Edges 60
3.2.4 Data storage 61
3.2.4.1 Format 61
3.2.4.2 Compression 61
3.2.5 Core comparison 62
3.2.5.1 Similar triplets 62
Preliminary note 63
Triplets of the same type 63

Length of the edges 63
Placement of the plane 64
Burial 64
Orientation of the chemical groups 64
Comparison of local shape 64
3.2.5.2 Connection of pairs 64
Double neighboring 65
Angle between triplets 65
3.2.5.3 Isolation of independent subgraphs 65
Obtaining independent subgraphs 65
Nature of the result 65
3.2.5.4 Filtering 65
Notion of functional flexibility 66
Deformation instead of superposition 66
3.3 Multiple comparison 66
3.3.1 General principle 67
3.3.2 The steps of the multiple comparison 67
3.3.2.1 Obtaining matched chemical groups 67
3.3.2.2 Obtaining sites 69
3.3.2.3 Grouping significantly overlapping sites 69
Overlap between 2 sites 69
Overlap between n sites 69
Characteristic sites 69
3.3.2.4 Matching characteristic sites 70
3.3.2.5 Graph of characteristic sites 70
3.3.2.6 Families of characteristic sites 70
3.4 Databases 70
3.4.1 Potential target structures 71
3.4.1.1 Removal of redundancies 71
Preserving the environment 71
Preserving boundary regions 72
3.4.1.2 Identification of redundant chains 72
3.4.1.3 Final size of the database 73
3.4.2 Ligand binding sites 73
3.4.2.1 Selection 73
3.4.2.2 Removal of the redundancies 73
3.4.2.3 Removal of tiny sites 73
3.4.2.4 Sequence of types of triplets 73
3.4.2.5 Size of the database 74
3.5 Predictive annotation 74
3.5.1 Families of ligands 74

    3.5.2 Apparent specificity 75
    3.5.3 Application: prediction and annotation 76
    3.5.4 Application: self validation 76
        3.5.4.1 Specificity at the level of the functional site . 76
        3.5.4.2 Computation 77
3.6 Details on heuristics 77
    3.6.1 Atomic density 78
    3.6.2 Comparison of local shape 79
        3.6.2.1 Formulation of the problem 79
        3.6.2.2 General scoring function 81
        3.6.2.3 Volume function 83
        3.6.2.4 Computation of the volume 84
    3.6.3 Estimation of deformation 85
        3.6.3.1 Nature of the problems 85
        3.6.3.2 Deciding what is local 86
        3.6.3.3 Solution 86
            Case of point objects 86
            Extension to multi-point objects without symmetry 88
            Adaptation for symmetric objects 89
    3.6.4 Incomplete cliques 92
        3.6.4.1 Definitions 92
        3.6.4.2 Algorithms 95
3.7 User interfaces 95
    3.7.1 Native SuMo scripts 96
        3.7.1.1 The SuMo language 96
            Program 96
            Expressions 96
            Declarations 97
            Functions 97
            Built-in types 97
            Constructors 98
            Comments 98
            Keywords and special characters 99
        3.7.1.2 The primitives of the SuMo language 99
        3.7.1.3 System for 3D selection 101
            The basic predicate constructors 101
            The `around` operator 102
            The classic boolean operators 103
            Priorities 103
            Complex examples 103

3.7.2 CGI/HTML interface 103
3.7.2.1 Interactive queries 103
3.7.2.2 SuMoQ queries 104
Introduction 104
Syntax 105
Semantics of SuMoQ 106
3.7.2.3 Display of the results 109
HTML 109
Saving the results 109
Export 110
3.7.2.4 Online help 112
3.7.2.5 Development of the system 112
Languages and external libraries 114
Printfer syntax extension 115
3.8 Task management 117
3.8.1 Job queue 117
3.8.1.1 Architecture 117
3.8.1.2 Priority management 119
3.8.1.3 Daemon startup 119
3.8.2 Parallel execution on a multi-processor machine 119
3.8.3 Distribution of the tasks over a cluster of computers . . 120
3.9 Frequently asked questions (FAQ) 120
3.9.1 About the method 120
Q1 Why triplets? 120
Q2 What about the flexibility of lateral chains? 120
Q3 Can we change any parameter? 121
Q4 Any statistical validation? 121
Q5 What about similar but distinct chemical groups? . 121
Q6 Are specific interactions taken into account? 121
Q7 Can SuMo be used on models? 122
Q8 Stability of the results after molecular dynamics? . 122
3.9.2 About the implementation 122
Q9 Isn't Caml a problem for industrialization? 122
Q10 Isn't Caml slower than C++? 122

4 Results of comparisons 123
4.1 Family of the legume lectins 123
4.2 Systematic comparison of sites 125
4.2.1 General data 125
4.2.2 Definition of families of ligands 126
4.2.3 Results 126

4.2.4 Comments 129
4.2.4.1 Representativity of the ligands 129
4.2.4.2 Problem of the infinite specificities 130
4.2.4.3 Problems related to the composition of the PDB 130

5 Discussion 131
5.1 Future of the software 133
5.1.1 Usage requirements 133
5.1.2 Perennity 134
5.1.3 Reusability 134
5.1.3.1 Heuristics 134
5.1.3.2 Languages 134
Specialized languages 134
Generic languages 135
5.1.4 Future developments 135
5.2 Current limits of the system 136
5.2.1 The localization problem 136
5.2.2 The scale problem 137
5.3 Conclusion 138

6 Involvement in other projects 139
6.1 Anti-apoptotic protein Nr-13 139
6.2 Geno3D 139

7 Publications 141
7.1 Articles 141
7.2 Patent 141
7.3 Oral communications 141
7.4 Posters 142

Annexes 144

A Definition of the chemical groups 145

B SuMo server's help 155

C Copy of the publications 167

Bibliography 191

Index 197

List of Figures

1.1 Search for ligand binding sites 18
1.2 Presentation of screening results 19

2.1 Biological data that are commonly used in molecular and cellular bioinformatics 22
2.2 Bioinformatic strategies for searching functional sites 25

3.1 Dependency graph between the modules of `sumo` 35
3.2 Command-line options for `sumo` 37
3.3 SuMo from a user's point of view 39
3.4 Interactions at the OS level 40
3.5 Colored tree representation of SuMo sources 42
3.6 Major steps in SuMo comparisons 43
3.7 Identification of hydrogen bonds 46
3.8 Examples of geometric variants 50
3.9 Quasi-adjacent triangles 60
3.10 Steps of the multiple comparison 68
3.11 Density of a set of points 80
3.12 Union and intersection of sets of spheres 82
3.13 Example of estimation of deformation 91
3.14 Maximal stable f-cliques in different examples 94
3.15 Specification of the SuMoQ language 108
3.16 Table of contents of SuMo's online help 113
3.17 Job management system: `jobqueue` 118

4.1 View of the site used to scan the legume lectin family 124
4.2 Results after scanning the legume lectin family 125
4.3 Definition of the current families of ligands 126

5.1 Position of SuMo in a process of drug design 133
5.2 Informatics and experimental sciences 138

List of Tables

3.1 The 3 level structure of SuMo 34
3.2 Type constructors and geometric variants 51
3.3 Functions of the SuMo language 100
3.4 External software that is used for the development of the SuMo web server 114
3.5 Main constructs of the Printfer syntax extension 116

4.1 Mean specificity of the ligand binding sites 127

5.1 Rough estimations of the length of the comparisons 136

List of algorithms

1 Identification of independent subgraphs 45
2 Extraction of the maximal f-cliques 96

Chapter 1

Introduction

Around 20,000 tridimensional structures of biological macromolecules are currently known and stored inside of the Protein Data Bank (PDB) [8], an international public database. This database contains a majority of protein structures (about 90 %), often complexed with partners of diverse compositions.

Today, the main biological mechanisms that lead to the production of proteins by the living cells are well-known by the biologists. The techniques of the molecular biology, that were developed in the 1980's, are mainly based on the manipulation of these mechanisms. Hence the notion of amino acid as the core component of proteins became central in the description of proteins in molecular biology. However, proteins are not the only components involved in biological mechanisms. Proteins themselves are frequently found in stable or unstable association with compounds of various origins. Although the notion of amino acid is a fundamental concept in the classic approaches of genetics, it should not be so while studying molecular interactions. The choice of relevant models for our biological problems has been crucial during the development of the bioinformatic system that will be described.

The project was developed for the following reasons:

- necessity of the tool dedicated to analysis and comparison of 3D structures of proteins that highlights similarities and differences that appear to be critical for the biological activity;
- such a tool should be easy to use for a beginner;
- its use should not require any specific material or complex expertise.

From these requirements, many strategies may be designed. There is a priori no need neither for minimizing any mathematically defined parameter such as an energy model, nor for superimposing optimally structures of proteins, nor for considering amino acids as an essential entity that forms polypeptides.

The method that has been chosen is *heuristic*: data structures and algorithms that have been developed were chosen without a formal proof of their relevance. This choice has been motivated by the impossibility of defining strictly what is a biological function. Related problems such as *Can this ligand form a stable complex with this protein of given 3D structure?* or *Is this site seldom?* can be expressed in a formal manner, but are not our matter of study. This system for detecting functional similarities among 3D structures of proteins has been named *SuMo*[1].

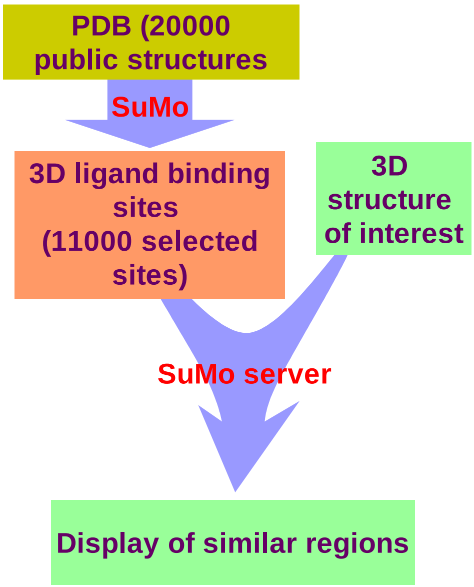
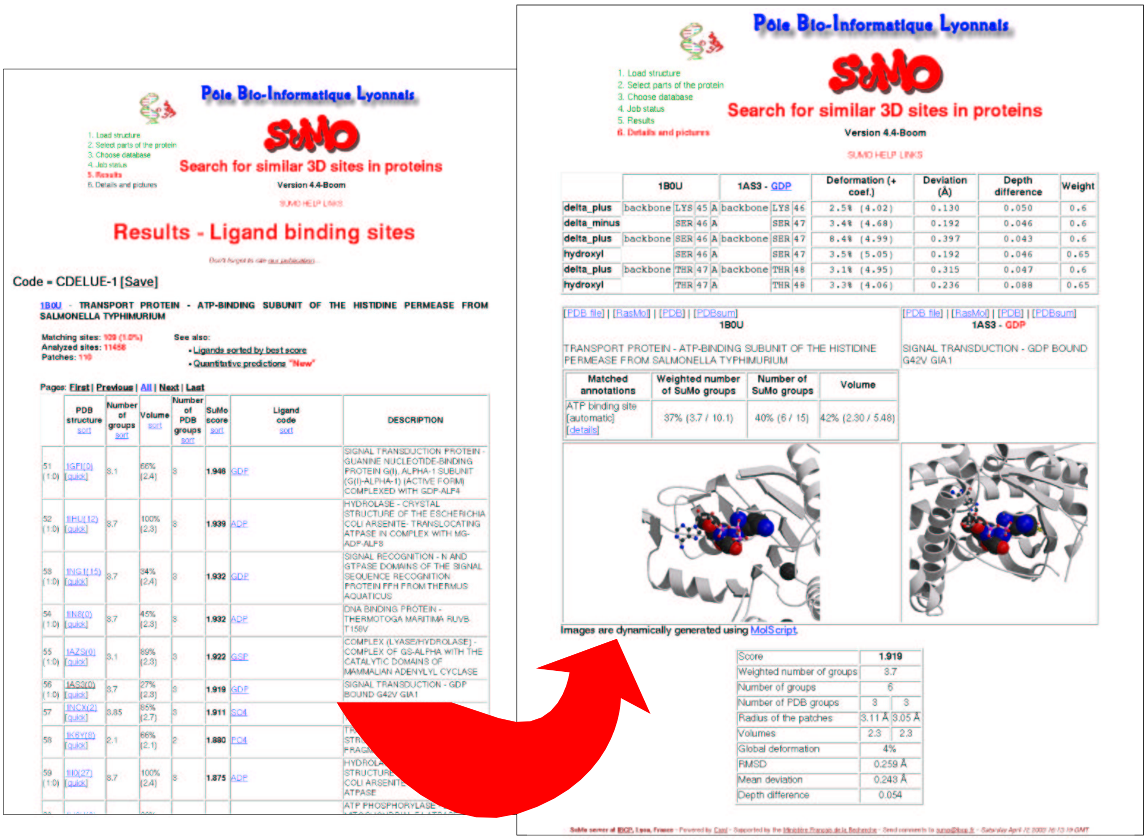

PDB (20000 public structures
SuMo
3D ligand binding sites (11000 selected sites)
3D structure of interest
SuMo server
Display of similar regions

Figure 1.1: Search for ligand binding sites in a query structure by SuMo

The SuMo project started in January 2000 and is designed to extract functional similarities from 3D structures of stable macromolecular complexes. SuMo aims at being as generic as possible, eventhough it is currently centered on the rigid protein/flexible ligand model. SuMo in its current state allows the screening of database of 11,000 3D ligand binding sites against a given 3D structure of protein. Results returned by SuMo propose potential binding sites of known ligands or fragments of these sites. This predictive approach is illustrated by figure 1.1 page 18 and the presentation of the results is shown on figure 1.2 page 19. It is a step in the identification of potential

---

[1] initially meant *Surfing the Molecules*



Pôle Bio-Informatique Lyonnais

SuMo

1. Load structure
2. Select parts of the protein
3. Choose database
4. Job status
5. Results
6. Details and pictures

Search for similar 3D sites in proteins

Version 4.4-Boom

SUMO HELP LINKS

Results - Ligand binding sites

Don't forget to cite our publication...

Code = CDELUE-1 [Save]

1B0U - TRANSPORT PROTEIN - ATP-BINDING SUBUNIT OF THE HISTIDINE PERMEASE FROM SALMONELLA TYPHIMURIUM

Matching sites: 109 (1.0%)
Analyzed sites: 11458
Patches: 110

See also:
- Ligands sorted by best score
- Quantitative predictions "New"

Pages: First | Previous | All | Next | Last

	PDB structure sort	Number of groups sort	Volume sort	Number of PDB groups sort	SuMo score sort	Ligand code sort	DESCRIPTION
51 (1:0)	1GFI(0) [quick]	3.1	66% (2.4)	3	**1.946**	GDP	SIGNAL TRANSDUCTION PROTEIN - GUANINE NUCLEOTIDE-BINDING PROTEIN G(I), ALPHA-1 SUBUNIT (G(I)-ALPHA-1) (ACTIVE FORM) COMPLEXED WITH GDP-ALF4
52 (1:0)	1IHU(12) [quick]	3.7	100% (2.3)	3	**1.939**	ADP	HYDROLASE - CRYSTAL STRUCTURE OF THE ESCHERICHIA COLI ARSENITE- TRANSLOCATING ATPASE IN COMPLEX WITH MG-ADP-ALF3
53 (1:0)	1NG1(15) [quick]	3.7	34% (2.4)	3	**1.932**	GDP	SIGNAL RECOGNITION - N AND GTPASE DOMAINS OF THE SIGNAL SEQUENCE RECOGNITION PROTEIN FFH FROM THERMUS AQUATICUS
54 (1:0)	1IN8(0) [quick]	3.7	45% (2.3)	3	**1.932**	ADP	DNA BINDING PROTEIN - THERMOTOGA MARITIMA RUVB T158V
55 (1:0)	1AZS(0) [quick]	3.1	89% (2.3)	3	**1.922**	GSP	COMPLEX (LYASE/HYDROLASE) - COMPLEX OF GS-ALPHA WITH THE CATALYTIC DOMAINS OF MAMMALIAN ADENYLYL CYCLASE
56 (1:0)	1AS3(0) [quick]	3.7	27% (2.3)	3	**1.919**	GDP	SIGNAL TRANSDUCTION - GDP BOUND G42V GIA1
57	1NCX(2) [quick]	3.85	85% (2.7)	3	**1.911**	SO4	
58	1K6Y(8) [quick]	2.1	66% (2.1)	2	**1.880**	PO4	TR STR FRAGM
59 (1:0)	1II0(27) [quick]	3.7	100% (2.4)	3	**1.875**	ADP	HYDROLA STRUCTURE COLI ARSENITE ATPASE
							ATP PHOSPHORYLASE

Pôle Bio-Informatique Lyonnais

SuMo

1. Load structure
2. Select parts of the protein
3. Choose database
4. Job status
5. Results
6. Details and pictures

Search for similar 3D sites in proteins

Version 4.4-Boom

SUMO HELP LINKS

	1B0U				1AS3 - GDP			Deformation (+ coef.)	Deviation (Å)	Depth difference	Weight
delta_plus	backbone	LYS	45	A	backbone	LYS	46	2.5% (4.02)	0.130	0.050	0.6
delta_minus		SER	46	A		SER	47	3.4% (4.68)	0.192	0.046	0.6
delta_plus	backbone	SER	46	A	backbone	SER	47	8.4% (4.99)	0.397	0.043	0.6
hydroxyl		SER	46	A		SER	47	3.5% (5.05)	0.192	0.046	0.65
delta_plus	backbone	THR	47	A	backbone	THR	48	3.1% (4.95)	0.315	0.047	0.6
hydroxyl		THR	47	A		THR	48	3.3% (4.06)	0.236	0.088	0.65

[PDB file] | [RasMol] | [PDB] | [PDBsum]
1B0U
TRANSPORT PROTEIN - ATP-BINDING SUBUNIT OF THE HISTIDINE PERMEASE FROM SALMONELLA TYPHIMURIUM

[PDB file] | [RasMol] | [PDB] | [PDBsum]
1AS3 - GDP
SIGNAL TRANSDUCTION - GDP BOUND G42V GIA1

Matched annotations	Weighted number of SuMo groups	Number of SuMo groups	Volume
ATP binding site [automatic] [details]	37% (3.7 / 10.1)	40% (6 / 15)	42% (2.30 / 5.48)

Images are dynamically generated using MolScript.

Score	**1.919**	
Weighted number of groups	3.7	
Number of groups	6	
Number of PDB groups	3	3
Radius of the patches	3.11 Å	3.05 Å
Volumes	2.3	2.3
Global deformation	4%	
RMSD	0.259 Å	
Mean deviation	0.243 Å	
Depth difference	0.054	

SuMo server at IBCP, Lyon, France - Powered by Caml - Supported by the Ministère Français de la Recherche - Send comments to sumo@ibcp.fr - Saturday April 12 2003 16:13:19 GMT

Figure 1.2: Presentation results obtained with SuMo. Example of the screening of the ligand binding site database using an ATP-binding protein.

therapeutic targets. Generally speaking, it should help the understanding of putative biological functions in proteins that are poorly understood, such as those studied by *structural genomics*.

Before reading this report, it is essential to try the SuMo system, from the following address:

> `http://sumo-pbil.ibcp.fr`

To use SuMo, the only requirements are to be connected to the Internet and to know the 3D structure of a protein. A quick view of functionalities proposed by the web server is considered as a part of the illustrations of this report.

This report describes the conceptual and technical fundamentals of the SuMo system. The use of SuMo and its application to specific biological problems is outside of the scope of this report.

The version which is described is SuMo 4.4, the first official public version, released in March 2003. SuMo is not a definitive methodology. Thus, modifications at any level may appear in the future versions of the system.

Chapter 2

Scientific and methodological context

2.1 Molecular and cellular bioinformatics

Figure 2.1 page 22 illustrates in a very simple manner the major data types that are routinely used in molecular and cellular biology, and that are easy to model and to store into computer-based databases. The different steps shown on the diagram are centered around proteins: the genetic information is stored in the cells using nucleic acids (DNA, RNA) and can be recorded as sequences, like for protein sequences. This convenient way to model the genetic information derives directly from experimental results but is not enough to understand the biological mechanisms, either in the case of proteins or in the case of genetic material.

Current bioinformatic tools do not allow to predict a biochemical pathway from the genome of a given organism. This is mainly due to the self-controlled nature of living organisms and the ignorance of the initial conditions that would be required to start the simulation of a living being.

Nowadays, bioinformatics proposes means to handle large or complex experimental data according to predefined and reproducible strategies. These approaches provide predictions and hypotheses that could help the biologist for his research.

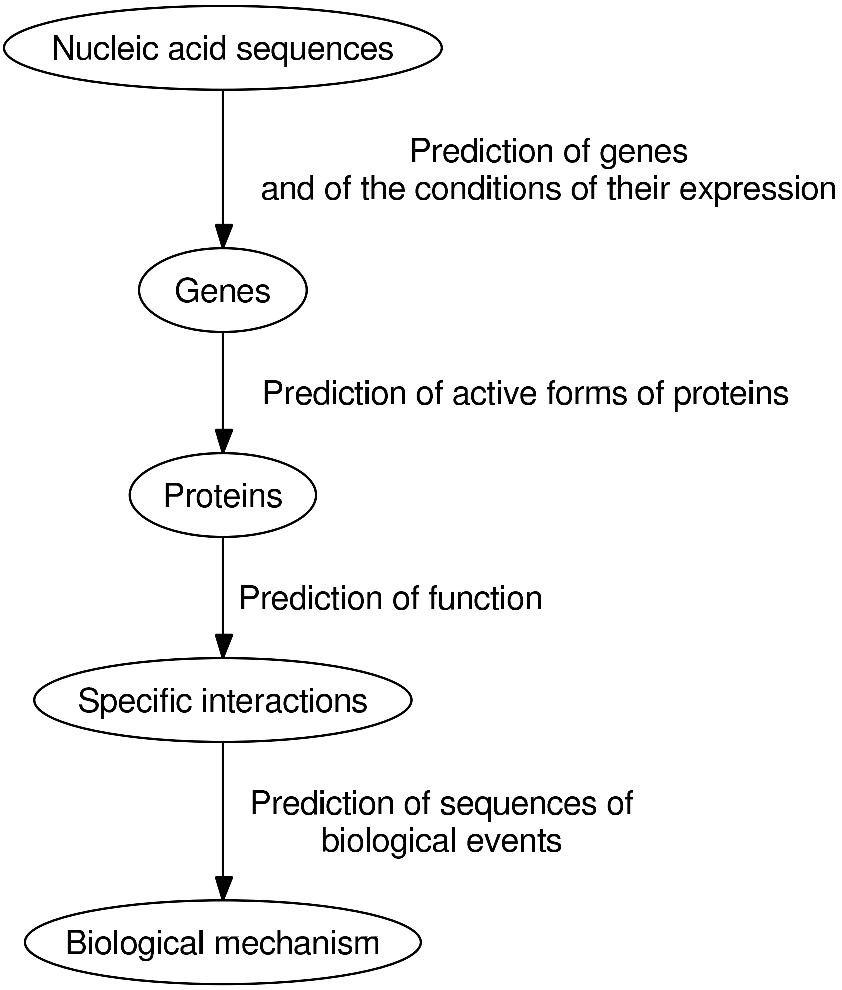

Nucleic acid sequences
Prediction of genes
and of the conditions of their expression
Genes
Prediction of active forms of proteins
Proteins
Prediction of function
Specific interactions
Prediction of sequences of
biological events
Biological mechanism

Figure 2.1: Biological data that are commonly used in molecular and cellular bioinformatics

2.2 Structure-function relation in proteins

2.2.1 General remarks

2.2.1.1 Notion of site

Proteins are natural linear heteropolymers, which exist in all known living organisms and whose monomers are generally the 20 classic amino acids. In many cases, proteins are subject to covalent modifications, by connection with other molecules, by clivage or by internal bridges like disulfide bridges between cysteines. Many proteins, under given conditions, acquire a stable conformation. A stable conformation is a specific fold with structural fluctuations that are small compared to the size of the protein. We will call 3D structure the stable conformation deduced from experimental data, when the error on atom coordinates is lower than the interatomic distances. Of course, all regions of a given protein don't share the same stability. In particular, the chains of atoms that interact with the external medium (solvent, ligands) and that have few steric constraints have more chances to be mobile. The major techniques that let us know the 3D structure of proteins are currently X-ray crystallography and nuclear magnetic resonance spectroscopy (NMR).

Most of proteins have an influence on the life of the organism where they are produced. We will call *function of a protein* the description of this influence. Protein functions may be various and their description is informal. However, numerous proteins interact with specific partners. These interactions generally only involve one part of the protein. Such regions are called *functional sites* and more precisely sites of interaction when they are only a contact region between the both partners. The study of protein families with similar sequences or similar functions often reveals amino acids or chemical groups that were particularly conserved through evolution. These regions may be considered as parts of functional sites, either in sites of direct interaction or in interactions that stabilize the protein's conformation.

2.2.1.2 Competition for interactions

Proteins from living organisms are located in dense environments (liquids, lipid membranes, crystals, ...). Thus they are in perpetual interaction with other molecules and may form complexes during highly variable periods. If a protein interacts in a significantly stable way with two different molecules at two overlapping sites, then there will be a *competition* between the two molecules. If a molecule interacts with certain proteins in a way which is more stable than the average, then this molecule is called *ligand* of these proteins.

A more precise definition of this notion will be given in section 3.2.1.1 page 47. Some ligands induce changes in the conformation of the proteins that they bind. This can lead to a change in the affinity of the same protein for a given ligand at a different site.

These remarks lead us to notice that the nature of the environment around a given protein will determine the probability for this protein to be bound to a given ligand at a given time t. It is particularly important to take this fact into account during the design of drugs that are supposed to act as competitive inhibitors at a given site in a given biological context.

2.2.2 Search for functional sites using comparisons

It is often interesting to know the functional sites of a given protein. In this field, any approach is allowed. We may distinguish experimental approaches—the ones that involve physical manipulation of the protein of interest—from informatic approaches.

Informatic approaches consist in converting experimental data using a more or less automated system in order to highlight interesting information in the data, information that is not trivial if we have a look at the raw experimental data. Only this kind of approaches will be our matter of interest, though they are complementary and indissociable from experimental approaches.

Proteins may be modeled in a simple way from experimental data:

- sequence of amino acids of the protein,
- 3D structure of the protein as set of cartesian coordinates of all or some atoms.

Sequences of the active forms of proteins of interest can be obtained from different methods with more or less confidence. Today, the public and generalistic database of non-redundant protein sequences Swiss-Prot [10] contains about 125,000 entries. This database is manually annotated with a maximum of experimental information and references to other sources of documentation. Other databases contain a larger amount of sequences; for instance PIR-NREF [5] puts around 1,200,000 non-redundant sequences together. 3D structures of proteins that are stored in the public database PDB constitute a set of around 19,000 entries, including several structures of identical proteins.

Different types of bioinformatic approaches allow, from sequences or structures, to predict functions and to localize functional sites in proteins with a variable quality. At this level, we may distinguish two kinds of strategies:

- strategies that are based on the comparison of known functional sites,
- ab initio strategies, i.e. those that do not use experimental informations of the same type as those that we wish to predict as initial conditions or constraints of the system.

Ab initio strategies will not be touched on here. Such approaches would have the advantage to highlight completely new biological functions. Regarding this topic, we may cite pure *docking* simulation without knowing potential regions that the given ligand could bind.

Figure 2.2 page 25 shows different types of bioinformatic approaches that can be used to highlight conserved sites in proteins. These approaches are classified into two categories: global alignment techniques or techniques of identification of similar sites, with here a very broad definition of the concept of site.

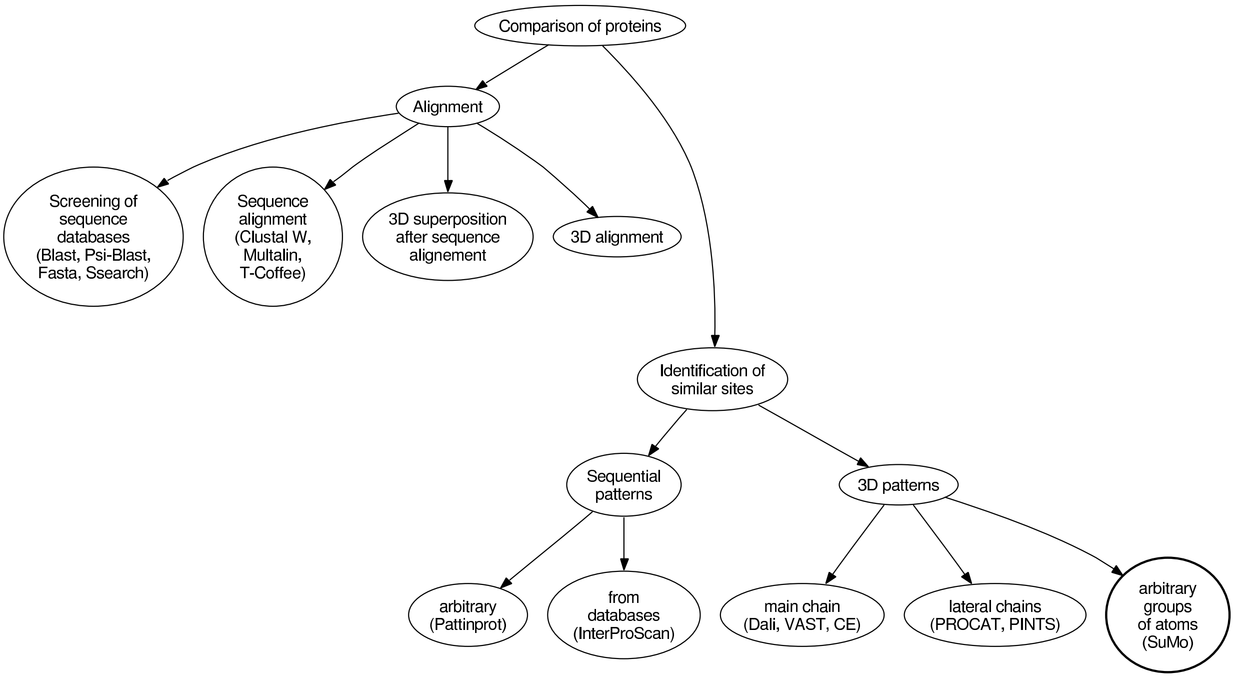

Comparison of proteins
Alignment
Screening of sequence databases (Blast, Psi-Blast, Fasta, Ssearch)
Sequence alignment (Clustal W, Multalin, T-Coffee)
3D superposition after sequence alignement
3D alignment
Identification of similar sites
Sequential patterns
3D patterns
arbitrary (Pattinprot)
from databases (InterProScan)
main chain (Dali, VAST, CE)
lateral chains (PROCAT, PINTS)
arbitrary groups of atoms (SuMo)

Figure 2.2: Bioinformatic strategies for searching functional sites in proteins using comparison methods, plus some examples of related tools

2.2.2.1 From the amino acid sequence

The amino acid sequence of a given protein is a poorer information than the 3D structure, at least because it can be trivially deduced from the 3D structure. This information is however easier to obtain experimentally.

Alignment of homologous sequences When the sequences of a set of proteins of identical function is available, it is possible to analyze which amino acids have been conserved through evolution. One can then consider that the most conserved ones are the most important ones for the protein's function. For this purpose, multiple alignment of homologous protein sequences may be performed with tools such as ClustalW [47] or Multalin [14].

Searching patterns A pattern, in the field of protein sequences, may be a regular expression. It may be obtained by various ways, possibly using multiple alignment of homologous protein sequences. The public database Prosite [4] contains more than 1,800 such patterns and most of them come with detailed documentation. A search in this database can be performed with the Proscan tool.

However the representation of a pattern by a single regular expression using alphabetical characters may not be ideal. Additional data such as effective or predicted secondary structure can be added, as well as other kinds of data known from experiments or predicted. Such a solution is proposed by the profile analysis [19].

2.2.2.2 From the 3D structure

Several bioinformatic tools perform the alignment of protein structures or identify structurally conserved domains regarding the main chain fold. We may cite Dali [20, 21] or CE [42]. These approaches allow the identification of homologies that are hard to detect using sequence alignment techniques. However they do not allow to take into account the position of atoms other than those of the main chain. Thus, we will focus on more sensitive approaches for finding functional sites that are not based on evolutionary data.

Proteins adopt relatively stable structures and thus can be under certain conditions be modeled as geometric objects whose local fluctuations are low compared to the distances that are critical for our study. Various properties can be projected on these objects and may let us discriminate sites using criteria that are not purely geometric.

Two classes can be distinguished among the methods for searching functional sites by comparison of 3D structures. This distinction is based on the primordial reprentation of the protein, i.e. the one which is the most important in the comparison process. These two kinds of representations are the following:

- model of the protein surface,
- representation of the protein using sparse points associated to properties that are independent from one point to another.

We will quickly describe the ground concepts of the existing methods, with a special emphasis on their advantages and limits.

Modeling protein surface Proteins can be seen as solid objects, whose surface shape has cavities and extrusions that may fit with molecules or parts of molecules. This is the key/lock model. On a surface representing a protein, it is possible to project different physico-chemical properties. A review of these methods has been written in 2000 by Via *et al.* [48].

Approaches based on heuristics from machine vision have been developed [18]. That kind of approach requires an abstract representation of the protein surface and then a discretization, i.e. a representation using a finite set of points located along the surface. Unfortunately, the number of points that are required for a fair representation of the theoretical surface is much higher than the number of atoms in contact with the molecular surface. This leads to heavy computations. In order to reduce this cost, a selection of sparse critical points may be performed among the points representing the surface [30]. In docking approaches, this approach has been extended with the introduction of a limited number of hinges in the molecules [41].

Other heuristics [38, 39] are also based on surface representation of proteins and involve a rigid reprentation of the surface.

The major limit of all approaches using a representation of protein surface is the computation time. Indeed, the initial information that is used to compute a surface is a set of atoms, which is a finite number of points whose location is known from experiments. However, the heuristics that are used discretize the surface representation and this results in a scattering of the original information.

Moreover, the choice of how to generate a molecular surface is free. It may be the surface that is in contact with a ball that rolls virtually on the atoms of the molecule. In this case, the radius of the atoms and the radius of the ball must be chosen in a relevant fashion. But several scales may be considered, depending on the various functional properties of the protein, and this results in several definitions of the surface. In addition, some amino acids that are essential for protein function are often buried in the protein.

Modeling using point objects Proteins may be cut directly into a set of elements that are modeled using point objects. A certain number of approaches have been based on the *one amino acid = one functional subunit* paradigm. We can cite [1] an approach that consists in searching isomorphous subgraphs where vertices model the position of C_α atoms and the position of lateral chains resulting.

As opposed to other approaches, the ones from Wallace *et al.* [51, 50] and Russell [40] do not consider only one position for each lateral chain. These approaches use directly the position of some atoms. In both cases, comparison is based on interatomic distances.

Wallace's approach defines for each lateral chain the position of a specific atom. This position is recorded along with information about its environment and this constitutes a hash key that allows fast searching for similar sites in a given database. This method is proposed via the PROCAT [52] server, a database of enzyme active sites.

Russell's approach is based on equivalence between certain lateral chains. The comparison is based on interatomic distances. Here we have the notion of group of atoms that we will also find in SuMo. However, it is here strongly bound to the notion of atom since comparisons involve comparisons of interatomic distances. This methodology is implemented and may be used at the PINTS [44] web server since early 2003. This implementation is associated to a scoring function [44] which purpose is to give a statistical meaning to comparison results. This statistics aims at giving a notion of theoretical degree of how much a site detected with a given RMSD is seldom. Here is the definition of the RMSD:

Definition 1 (RMSD) *Let E and F be subsets of $\mathbf{R}^n$. Function* rmsd *is defined as follows, for every set $L \subset E \times F$:*

$$\mathrm{rmsd}(L) = \sqrt{\frac{1}{|L|} \sum_{(p,q)\in L} \|p-q\|^2}$$

where $|L|$ is the cardinal of set L. Let Φ be the set of rigid transforms from $\mathbf{R}^n$ into $\mathbf{R}^n$, i.e. the transforms that keep distances and oriented angles. Function RMSD *is defined as follows:*

$$\begin{aligned}&\mathrm{RMSD}(\{(p_1,q_1),(p_2,q_2),\dots\}) = \\ &\quad \min\{\mathrm{rmsd}(\{(p_1,\phi(q_1)),(p_2,\phi(q_2)),\dots\}) | \phi \in \Phi\ \}\end{aligned}$$

The purpose of RMSD is to compare objects that are overall superposables. There is no special reason to think that RMSD is the best criterion to express functional similarity between two sites. Indeed, numerous ligands of proteins are flexible and can bind sites of various shapes using the same kind of interactions.

We will see that the SuMo approach is design to obtain results that satisfy the user in the most cases rather than rigourous statistical functions that are not really realistic when confronted to the huge diversity of biological functional sites.

2.3 Programming tools

The choice of well-suited programming tools is crucial in bioinformatics, as it is in all domains of applied informatics. A fairly large amount of time is spent to implement original algorithms and it is important to reduce this time as much as possible.

Our purpose is not to praise a given programming language. There is no proof that a language is better than another for a given task. Let us give simply some general points that may help to think about the choice of a programming language:

- pleasure while programming
- time spent to program
- ease of writing code
- ease of extension of existing code
- size of the programs
- quality of the compilers
- portability of compilers and libraries
- viability of the language
- availability of useful libraries
- reusability of the code
- ease of learning
- quality of the available documentation about the language
- possibility of syntax specialization
- interfacing with other languages
- degree of specialization of the language

The programming language that has be chosen is *Objective Caml* [29]. It is developed at the INRIA[1]-Rocquencourt, France.

2.3.1 General-purpose programming language

Objective Caml is a strongly typed general-purpose programming language. It allows to use together functional, imperative or object programming styles. Memory management is completely automatic. Typing is completely static, thus allowing efficient compiled code and detection of programming errors at compile time. Efficiency of generated code often reaches the one of equivalent C code [3] when using the native code compiler. Compilers and the standard library are available for the most common platforms which are Windows, MacOS and all Unix variants.

Advanced users generally consider that using Objective Caml instead of C or C++ results in productivity increased by a factor between 5 and 10 for professional software development.

2.3.2 Specialized syntaxes

The Objective Caml distribution also provides tools dedicated to the manipulation of specialized syntaxes.

The *Ocamllex/Ocamlyacc* couple is the equivalent of Lex/Yacc that were originally developed for the C language. These tools provide a way to easily manipulate languages defined by the programmer. These tools have been used for all languages defined in the SuMo system.

Camlp4 [15] is a system that allows to extend the syntax of Caml, in order to simplify code writing in some specific cases. It also provides the possibility to define a syntax from scratch, provides a revised syntax for Caml and the possibility to automatically convert code between the classic syntax and the revised one. Only syntaxic extensions have been defined or used in SuMo.

2.3.3 Storing data

SuMo does not require the use of a relational database to store the preformatted data that represent the 3D structures for comparison. On the other hand, these data are quite complex and cyclic. Storing them has been made very easy by the *Marshal* module. This module is included in the standard library of Objective Caml. It provides polymorphic input/output functions for almost any type. Currently, only objects cannot be converted in this format and functions or closures can only be read back by the program that generated the data.

---

[1] translation note: French public research institution for research in informatics and automated systems

Caml is statically typed but the program that reads back the data in the Marshal format must know their type at compile time, and the data that are loaded at runtime must match this type or the program will crash. Using this module incorrecty is one of the very few ways to produce a *Segmentation fault* crash with a pure Objective Caml program. For this reason, data loaded in the Marshal format must be secure. In order to make this system for storing data available to distant users, a system of cryptographic signature has been added. This is described page 110.

2.3.4 Communication

The user interface for SuMo relies on a client-server system: heavy computations are performed on an HTTP server and the client gets web pages (HTML format) in return. The mechanism for running programs on an HTTP server's side according from HTTP clients is called CGI.

Such an interface is more difficult to implement and less use-friendly than a specific graphic interface embedded in a client-side software but has the following advantages:

- there is no need for the user to install any specific software on his computer,
- the user doesn't need to install a database on his machine,
- there is no need to distribute the software,
- the software does not need to very portable,
- the users do not need to worry about updates and everyone uses the same version.

Chapter 3

Description of the SuMo system

The SuMo system consists in a method and its implementation. They have been developed together. Efforts have been spent especially on the following points:

- provide a tool for analysis of 3D structures of proteins,
- develop a tool that is useful for any biologist interested in the function of macromolecules,
- design and implement all of this within the PhD period,
- create a general-purpose system, as far as it is possible,
- develop a software which would be efficient enough to support several user queries every day,
- make it easy to use,
- make it flexible enough to satisfy a wide range of needs.

First, the architecture of the system will be described, using different points of view. The description of the core method is then given but some models, heuristics or algorithms that are especially complex and independent from the concept of macromolecule are described in a separate section. Later is described the design of the user interfaces and how the computations are distributed on different machines. Finally, a *FAQ*[1] give some answers to some questions that have been frequently asked about SuMo and refers as much as possible to the related sections of this document.

---

[1] Frequently Asked Questions

3.1 General architecture

3.1.1 The different levels

SuMo is structured in 3 layers: the higher level layer is the one accessible to the regular user, through the web interface. The lower level layer is the one of the programming language and should be restricted to the developers of SuMo. The intermediate layer allows the developer to perform some tests and an advanced user to perform specific tasks that are not proposed by default at the web interface level. The essential features of these three levels are presented in table 3.1 page 34.

Level	Scale	Kind of users	Language
Lower	Machine/Local system	Programmer	Caml
Medium	File system	Prog. of interfaces	SuMo
Upper	Worldwide	User	None/SuMoQ

Table 3.1: The 3 level structure of SuMo

3.1.1.1 Lower level: programming language

The lower level of use is the programming language layer. A user that modifies SuMo at this level is a developer.

The SuMo system contains several independent executables. It is implemented quasi-exclusively in Objective Caml. There are 285 Caml or Caml-based (Camlp4, Ocamllex, Ocamlyacc) source files, for a total of 30,000 lines in version 4.4. The program that implements the method from the input of structural data to the output of result files is named `sumo`. The `sumo` program itself is composed of 181 source files or 157 modules and 20,000 lines of code. Figure 3.1 page 35 shows the dependency graph between the modules of `sumo` and illustrates the relative complexity of the heuristics that has been developed.

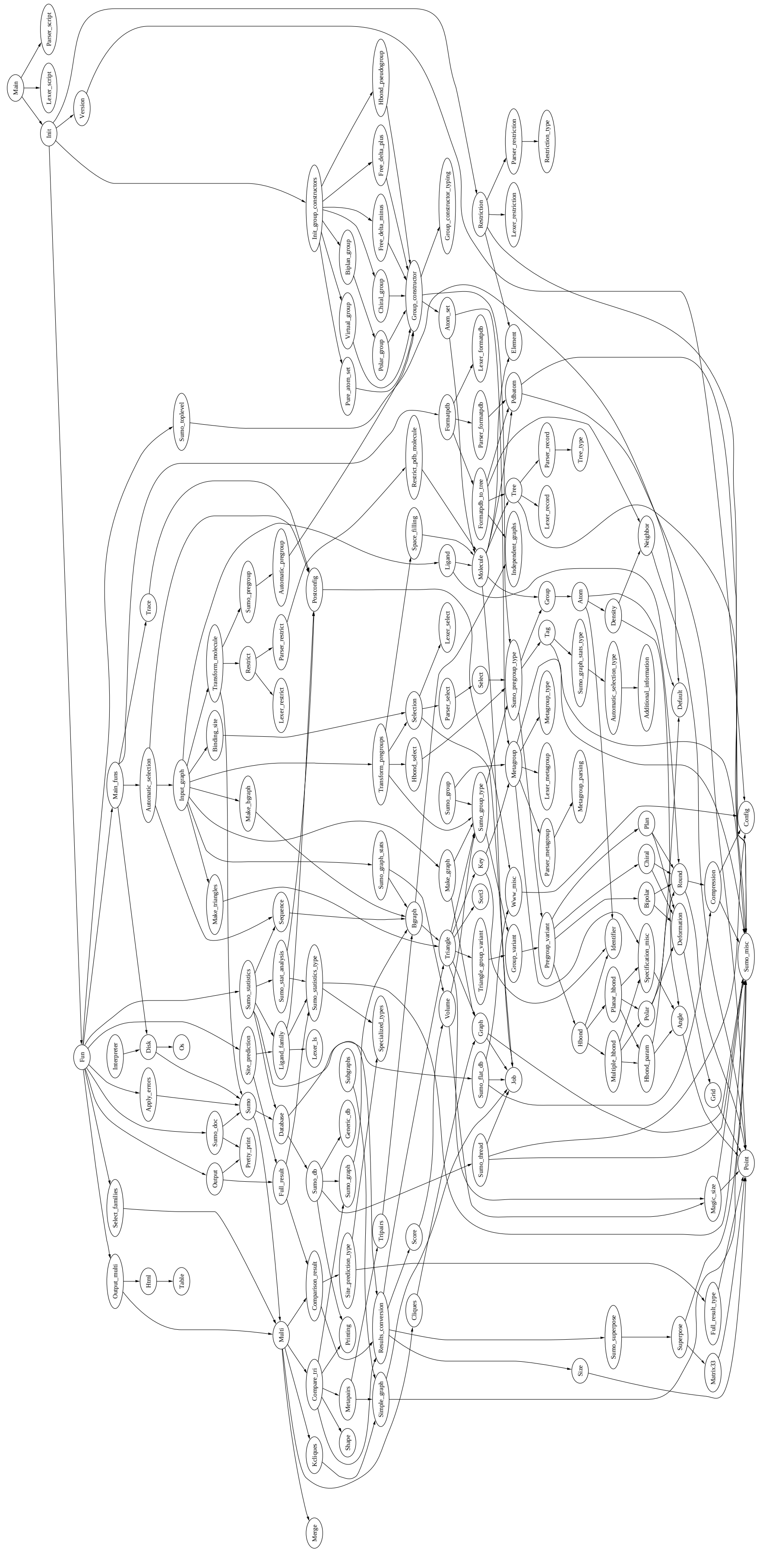

Main
Lexer_script
Parser_script
Init
Version
Init_group_constructors
Hbond_pseudogroup
Free_delta_plus
Free_delta_minus
Biplan_group
Chiral_group
Virtual_group
Polar_group
Pure_atom_set
Group_constructor
Group_constructor_typing
Restriction
Parser_restriction
Restriction_type
Lexer_restriction
Atom_set
Lexer_formatpdb
Element
Pdbatom
Parser_formatpdb
Formatpdb
Sumo_toplevel
Restrict_pdb_molecule
Formatpdb_to_tree
Tree
Parser_record
Tree_type
Lexer_record
Independent_graphs
Neighbor
Space_filling
Ligand
Molecule
Group
Atom
Density
Trace
Sumo_pregroup
Automatic_pregroup
Posconfig
Transform_molecule
Restrict
Parser_restrict
Lexer_restrict
Lexer_select
Selection
Parser_select
Select
Sumo_pregroup_type
Tag
Sumo_graph_stats_type
Automatic_selection_type
Additional_information
Default
Main_funs
Automatic_selection
Input_graph
Binding_site
Transform_pregroups
Hbond_select
Metagroup_type
Metagroup
Lexer_metagroup
Metagroup_parsing
Sumo_group
Sumo_group_type
Make_bggraph
Parser_metagroup
Plan
Config
Chiral
Round
Compression
Make_triangles
Sumo_graph_stats
Make_graph
Key
Www_misc
Bipolar
Sequence
Bggraph
Sort3
Triangle
Triangle_group_variant
Group_variant
Pregroup_variant
Identifier
Specification_misc
Deformation
Sumo_misc
Sumo_statistics
Sumo_stat_analysis
Sumo_statistics_type
Specialized_types
Volume
Graph
Hbond
Planar_hbond
Polar
Angle
Fun
Interpreter
Disk
Os
Site_prediction
Ligand_family
Lexer_ls
Subgraphs
Sumo_flat_db
Job
Multiple_hbond
Hbond_param
Grid
Apply_errors
Sumo
Database
Generic_db
Sumo_doc
Pretty_print
Sumo_db
Sumo_graph
Sumo_thread
Point
Output
Full_result
Magic_size
Select_families
Triplets
Score
Output_multi
Html
Table
Comparison_result
Site_prediction_type
Cliques
Full_result_type
Multi
Printing
Results_conversion
Sumo_superpose
Superpose
Compare_tri
Metapairs
Simple_graph
Size
Matrix33
Kcliques
Shape
Merge

Figure 3.1: Complexity of `sumo`: dependency graph between the 157 modules the `sumo` program, version 4.4. The submodules are not shown. $A \rightarrow B$ indicates that module A depends on module B. This graph representation was generated thanks to Ocamldot and Graphviz.

3.1.1.2 Medium level: SuMo language

The medium level is the `sumo` program, when used without additional interface. Since version 2 (September 2000), the program offers a specific language, in order to manage the various tasks and options provided by the program. This language has been designed especially for SuMo. It is typed and may be used in interactive mode, via scripts stored in independent files, or both. For instance, if we want to record a sequence of commands into a script that works on a given 3D structure, we may:

1. give the PDB file on the command line,
2. use a generic script that reads and converts these data,
3. create and use a specific script recorded in another file.

This results in the following command:

```
sumo -interactive \
  -preamble 'let pdb_file = PDB_file "/pdb/pdb2pel.ent";' \
  my_generic_sumo_script.sumo \
  my_specific_script.sumo
```

where file `my_generic_sumo_script.sumo` may contain the following:

```
(* These commands require pdb_file to be defined! *)
let sumo_data = read pdb_file -densmax 0.5;
```

and file `my_specific_script.sumo` may contain the following:

```
(* See which chemical groups are used *)
print sumo_data -file "my-sumo-groups.txt";

(* Screen my database of full structures with my structure *)
let result =
 compare sumo_data DB ("/usr/local/db/sumo/automatic/full");

(* Output the result in human-readable format *)
print result -file "sumo-results.txt";
```

Command-line options may be displayed with the `-help` option (fig. 3.2 page 37) or from the manpage.

The interaction of `sumo` with the operating system is limited to the file system. There is no restriction concerning the location or the name of files and databases. A detailed description of the SuMo language and its predefined functions is given page 96.

```
[pc-bioinfo1] ~/these/rapport % sumo -help
Usage: sumo [input file] [options]

For further details on the command line options, see ‘man sumo’.
To obtain help with sumo functions, start with ‘sumo -i’ and type ‘help
();’.

Options are:
  -interactive  force interactive mode
  -i  same as -interactive
  -non-interactive  force non interactive mode (default)
  -preamble <prog>  parse these commands before the sumo input files
  -pdb-groups <file>  overrides the environment variable PDB_GROUPS
  -remove <file>  removes file at exit
  -disable-sumo-swap  disables automatic sumo swap
  -memory-compaction  not documented
  -no-memory-compaction  not documented
  -sumo-groups <file>  overrides the environment variable SUMO_GROUPS
  -verbose  displays additional information
  -release  displays the release identifier and exits
  -version  displays the version identifier and exits
  -help   Display this list of options
  --help  Display this list of options
```

Figure 3.2: Command-line options of the `sumo` program

3.1.1.3 Higher level: SuMoQ comparison queries

Regular users use the web interface of SuMo. It provides an access to the SuMo services without any local installation of software, except a web browser. The simplest way to formulate a query is the interactive way, by filling the required fields from page to page. These steps end with the generation of a query-file in a specific format, that will be submitted to the `sumo-run` program which starts the comparisons on the server side, using `sumo`. Although this mechanism is transparent for the user, it allows him to save the query in a text format which is human-readable as well as machine-readable, for later reuse. The language used to described the queries is called *SuMoQ*. It provides a richer set of possible queries than what can obtained through the interactive mode but is still totally independent from the underlying operating system: there is no notion of file in this language. Here is an example of query that may be used to screen the database of ligand binding sites with chain A of PDB structure 2PEL:

```
{
  email = "martin_jambon@emailuser.net";
  title = "My query for scanning ligand binding sites";
  {
    subtitle = "Lectin 2PEL";
    scan = {
      database = "ligands";
      pdb_id = "2PEL";
      selection = "Pdb_chain \"A\"";
    } /scan;
  } /
}
```

A detailed description of SuMoQ queries is given page 104.

3.1.2 SuMo from different points of view

How is SuMo structured? We will answer this question in a synthetic way. However, depending on the person we are asking we may get various answers. We will here consider three classes of people who will have something to do with SuMo:

1. users,
2. system administrators,
3. developers.

Developers of course need to have in mind the points of view of the administrators and the users, but the inverse is not true.

3.1.2.1 A user's point of view

Regular users access SuMo only through the web server. They are not supposed to know how the system was programmed, nor which infrastructure stands behind the HTTP server. Users have access to:

- dynamically generated web pages,
- data that may be retrieved and stored into files or that are accessible using a query identifier,
- links to web pages or to software that are independent from SuMo.

Figure 3.3 page 39 shows the different steps of a SuMo query, as well as input and output data and interfaces to other software. It is important to notice that the system for one-step queries SuMoQ allows any HTTP client software to submit SuMo queries. Integral data may be retrieved in an XML format. Therefore, SuMo may be integrated into any informatic platform—a local software or a web meta-server without modifying SuMo and without running SuMo on a local host.

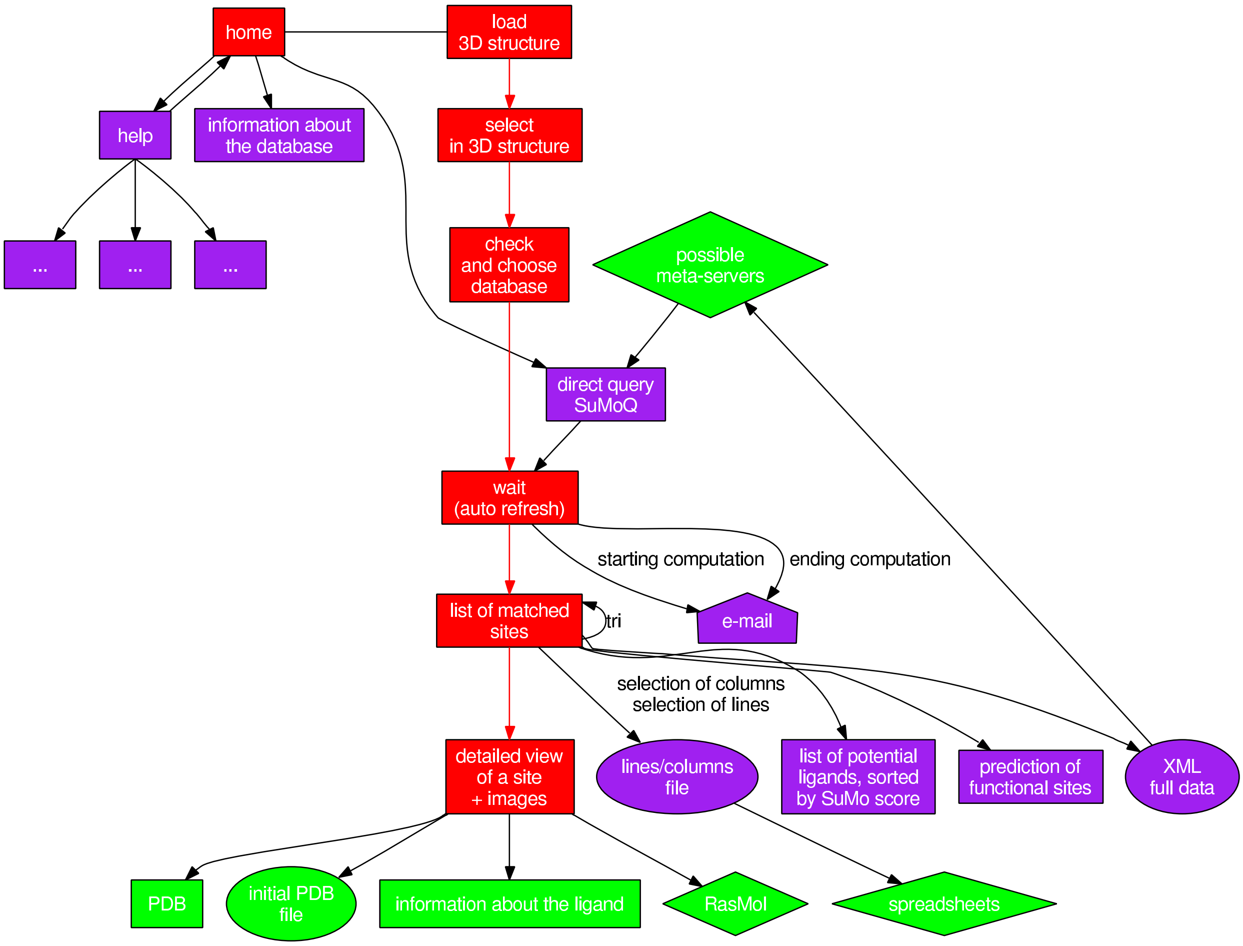

home
load 3D structure
help
information about the database
select in 3D structure
...
...
...
check and choose database
possible meta-servers
direct query SuMoQ
wait (auto refresh)
starting computation
ending computation
list of matched sites
tri
e-mail
selection of columns
selection of lines
detailed view of a site + images
lines/columns file
list of potential ligands, sorted by SuMo score
prediction of functional sites
XML full data
PDB
initial PDB file
information about the ligand
RasMol
spreadsheets

Figure 3.3: SuMo from a user's point of view. The essential steps in a SuMo query are drawn in red, purple stands for optional steps and green stands for systems and data that do not belong to the SuMo system.

3.1.2.2 The administrator's point of view

An administrator is the person who is in charge of installing SuMo on local computers. Figure 3.4 page 40 presents programs, processes, files and signals that are involved during a SuMo query. The configuration shown on this figure uses the priority queue based system developed for SuMo and named `jobqueue` (see section 3.8.1 page 117).

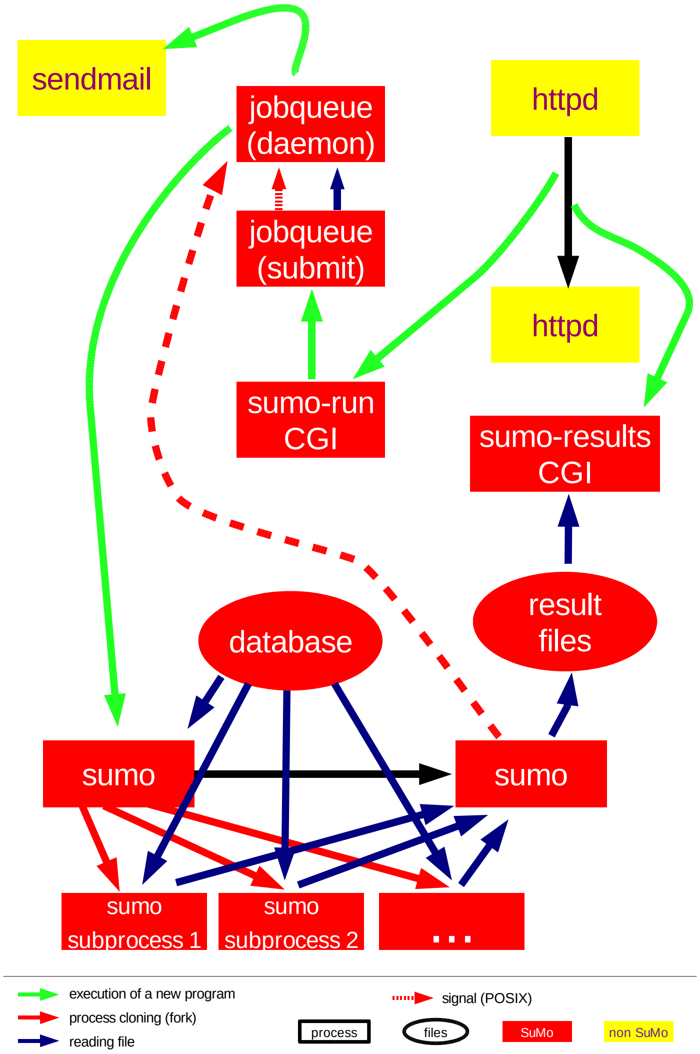

sendmail
jobqueue (daemon)
httpd
jobqueue (submit)
httpd
sumo-run CGI
sumo-results CGI
database
result files
sumo
sumo
sumo subprocess 1
sumo subprocess 2
...
execution of a new program
process cloning (fork)
reading file
signal (POSIX)
process
files
SuMo
non SuMo

Figure 3.4: Interactions at the operating system level during a SuMo query. The option that allows the use of a cluster of computers using PBS is not shown here. Red components are specific to the SuMo system. Red arrows represent an interaction which happens necessarily on the same host.

3.1.2.3 The programmer's point of view

Programmers, even if they are not very familiar with SuMo principles, must have a good overview of the source tree. Figure 3.5 page 42 shows the most important directories and some files from the source tree. The CVS archiving system is used to manage the successive versions of every file. The source files in SuMo may be classified into three categories:

1. compilation accessories,
2. sources for command-line programs,
3. sources for programs used for the web interface.

Compilation utilities that were developed for SuMo include `Makefiles`, the `configure` file, the `VERSION` file, the `substitute`[2] program and the Printfer syntaxic extension, which is described in section 3.7.2.5 page 115.

The different command-line tools allow to perform various tasks. Some are utilities that may be used manually like `build-sumo-db`, the program that generates the database or the `sumo-extend` utility that automatically extends the `pdb_groups`[3] configuration file. Other programs are used by the CGI programs but may also be used directly, such as `sumo`, `cns2pdb`[4], `pdb2tree`[5], `seqinfo`[6], `sumo-sort`[7] and `sumo-columns`[8].

The programs that were written for the web interface of SuMo include the following CGI programs: `sumo-help`, `sumo-database`, `sumo-welcome`, `sumo-select`, `sumo-check`, `sumo-run`, `sumo-results` and `sumo-focus`. Other programs have been written for the SuMo server, namely `sumo-sign`[9], `jobqueue` (see section 3.8.1 page 117) and `sumo-clean`[10].

3.2 Pairwise comparison of 3D structures

This section describes the core of the SuMo system, i.e. how from structural data such as those found in the PDB it can identify similar regions between

---

[2] replaces `@@TOTO@@` with the value of parameter `TOTO`

[3] contains the correspondance between atomic symbols in the PDB format and the elements from the periodic classification

[4] conversion from format X-Plor/CNS PDB format to standard PDB format

[5] conversion from PDB format to a tree format

[6] information about protein sequences in 3D structures

[7] lines and columns sorting

[8] conversion of results from `sumo` into line/columns format

[9] cryptographic signature of data

[10] removal of temporary public files

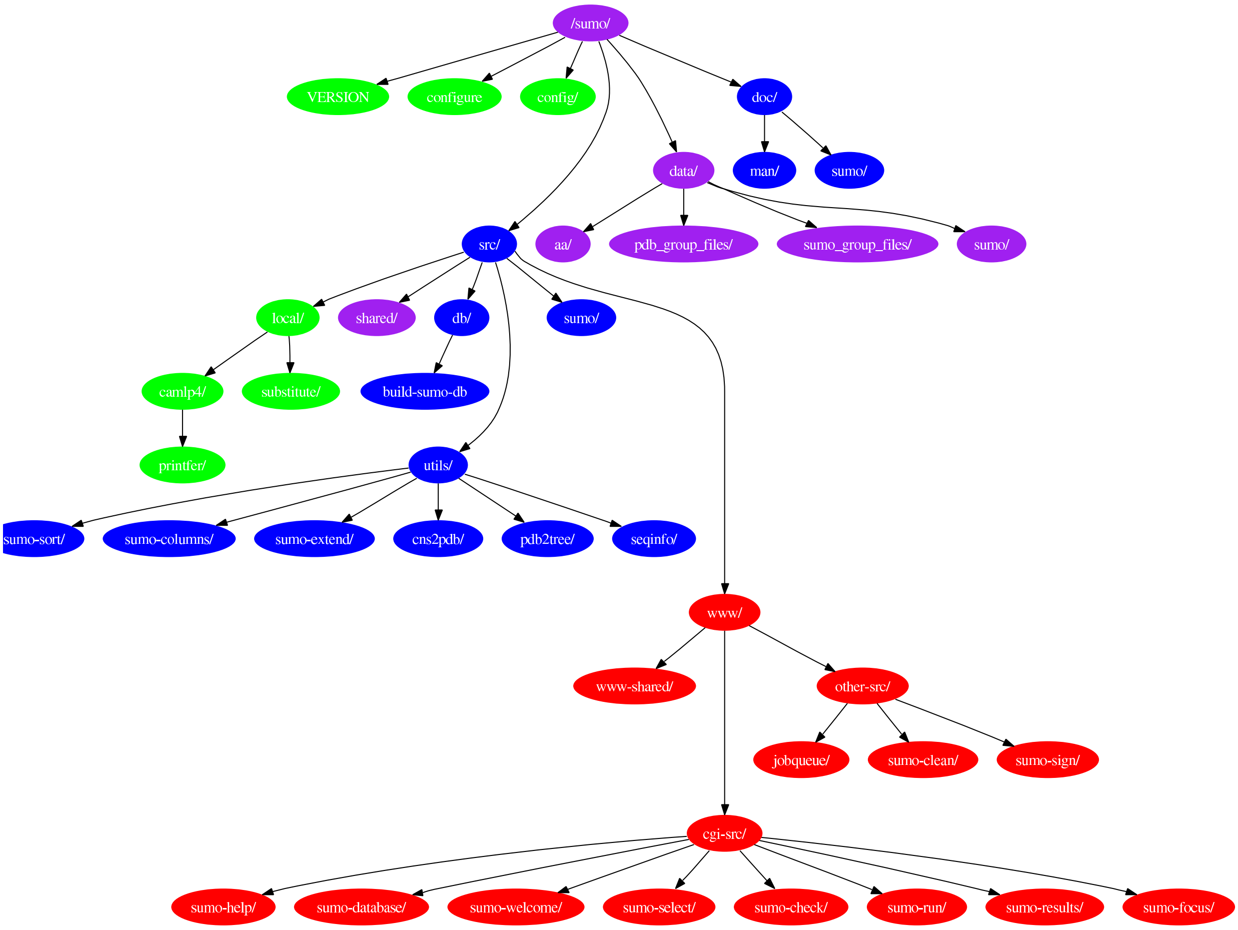

/sumo/
VERSION
configure
config/
doc/
man/
sumo/
data/
aa/
pdb_group_files/
sumo_group_files/
sumo/
src/
local/
shared/
db/
sumo/
camlp4/
substitute/
build-sumo-db
printfer/
utils/
sumo-sort/
sumo-columns/
sumo-extend/
cns2pdb/
pdb2tree/
seqinfo/
www/
www-shared/
other-src/
jobqueue/
sumo-clean/
sumo-sign/
cgi-src/
sumo-help/
sumo-database/
sumo-welcome/
sumo-select/
sumo-check/
sumo-run/
sumo-results/
sumo-focus/

Figure 3.5: Colored tree representation of SuMo source files. In green: compilation tools, in blue: command-line tools, in red: programs written for the web interface, in purple: shared code or data.

two 3D structures of macromolecules. The essential steps are represented on figure 3.6 page 43.

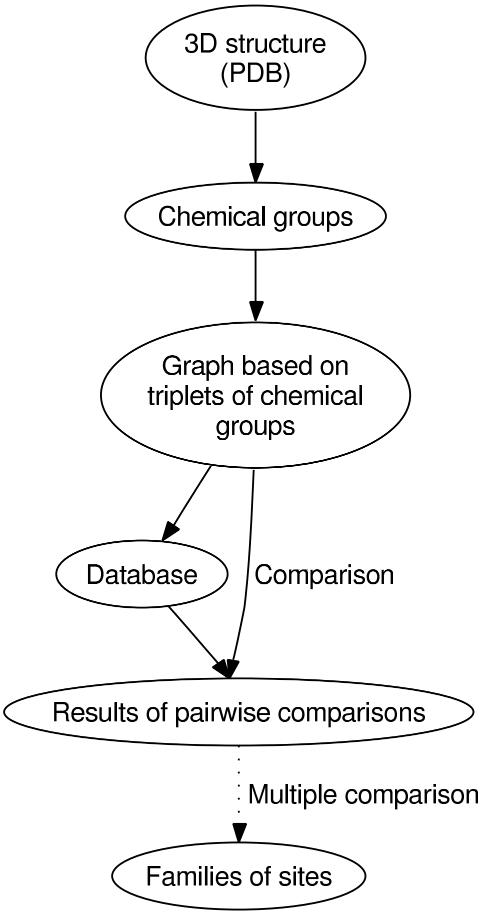

3D structure
(PDB)
Chemical groups
Graph based on triplets of chemical groups
Database
Comparison
Results of pairwise comparisons
Multiple comparison
Families of sites

Figure 3.6: Major steps in SuMo comparisons

The pairwise comparison step is central and the most critical concerning computing time. To make it faster, data are preprocessed and stored so that they are ready to use for any comparison request. The core comparison allows a fast generation of lists of pairs of matched chemical groups from the molecular complexes being compared.

Multiple comparisons and automated prediction of ligand binding sites rely on the processing of results from pairwise comparisons of structures or substructures. These post-processing options are described later.

3.2.1 Chemical groups

SuMo is based on a description of tridimensional macromolecular structures using chemical groups. The concept of *chemical groups* in SuMo corresponds to the modeling of a given region in the tridimensional structure of a set of molecules. The limits of this definition are more defined by the nature of heuristics comparison that will be used rather than by the traditional concept of chemical group. These limits will be discussed later.

3.2.1.1 Preliminaries

Some preliminaries are required before manipulating the coordinates of the atoms from molecules as found in PDB files. We need to distinguish the different molecules from a given complex and need to know their size, in particular to distinguish the parts that should be taken into account by SuMo and the rest.

Identification of the molecules It will be necessary to distinguish the different molecules that are not covalently bound and belong to a given structure.

One important aspect of SuMo is the prediction of ligand binding sites using comparisons against ligand binding sites from the PDB. The first task for SuMo is to discriminate the different molecules in the experimental structure. This information is not given in a reliable manner in the PDB files. PDB files contain a notion of chemical group. These groups that we will call PDB groups represent the classic monomers from the biological macromolecules such as amino acids and nucleic acids (ALA, CYS, ..., A, C, G, T, U). These are tagged with the ATOM keyword. Other PDB groups are tagged with the HETATM keyword and concern all other kinds of molecules (HOH, ATP, CA, GLC, ...). The information indicating the chain in PDB files matches polypeptidic and polynucleotidic chains but its role is not clear concerning ligands. The chemical groups whose atoms are tagged HETATM may be or may not be covalently bound to ATOM or HETATM groups belonging to a protein or any other kind of molecule.

In SuMo, a data structure is generated from the file. This data represents a subset of the PDB data in a tree format using a non-ambiguous syntax and gives some additionally computed information. This data can be exported or imported in the syntax which is described page 105. It may also be exported into an XML format and displayed in a web browser. This data eliminates distinctions between ATOM and HETATM but instead indicates to which molecule each atom belongs, according to SuMo's rules.

In order to determine to which molecule each atom belongs in a given 3D structure, we have to search for all the covalent bonds that are given in the PDB file. The limitation of this system is that non resolved parts of the structure may break a molecule into 2 apparent fragments. However this is most of the time relevant since the non-resolved portions are usually very flexible. In that case, it is not a big mistake to consider that two elements that are only linked by a flexible loop of undefined structure may interact with each other in a similar fashion as if they were effectively two distinct molecules.

The algorithm is based on a graph whose vertices are the n atoms from the structure and the edges represent covalent bonds; it is then only necessary to isolate independent subgraphs:

1. for each atom, neighbors are searched within an $r_{\max}$ radius and connected if a covalent bond is possible between these two atoms. Cost: $O(n)$
2. every vertex is marked with a new identifier unless it is already marked, and the same identifier is used to mark recursively all its neighbors. Cost: $O(n)$

Algorithm 1 page 45 gives the details about the marking procedure.

Algorithm 1 Identification of independent subgraphs

```
Require a graph $G = (V, E)$
function Mark $(v, k)$
    if $v$ is marked then do nothing
    else
        Set-Mark $(v, k)$
        for all $v' \in$ Neighbors $(v)$ do Mark $(v', k)$

$k \leftarrow 1$
Result $\leftarrow \varnothing$
for all $v \in V$ do
    if $v$ is marked then do nothing
    else
        Mark $(v, k)$
        $k \leftarrow k + 1$
```

Covalent bonds are currently attributed according to distance criteria. These criteria are the following: if A and B are 2 atoms, then we consider that a covalent bond exists between them if they are distant from less than

$d_{\max}$:

$$d_{\max} = \begin{cases} 1.3 \text{ Å} & \text{one of the atoms is a hydrogen} \\ 1.9 \text{ Å} & \text{otherwise} \end{cases}$$

Detecting hydrogen bonds We will see that SuMo proposes the definition of chemical groups that model free hydrogen bond donors and acceptors. A *free hydrogen bond donor or acceptor* is not involved in a hydrogen bond or will not be involved in a hydrogen bond after a small local conformational change. A hydrogen bond donor may interact in zero or one hydrogen bond. An acceptor may be involved in several hydrogen bonds.

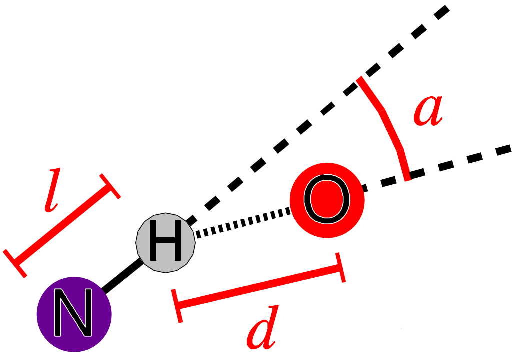


Figure 3.7: Model used for identifying hydrogen bonds. Example of a $NH^{\delta+}$–$O^{\delta-}$ interaction.

Figure 3.7 page 46 introduces the model that was used for detecting hydrogen bonds and shows the example of a $NH^{\delta+}$–$O^{\delta-}$ interaction. The criteria that must be matched for a hydrogen bond are the following:

$$\begin{cases} 0 \leq a \leq \alpha \\ |d - \beta| \leq \gamma \end{cases}$$

where α, β and γ are parameters. The values that were used for hydrogen bonds involving nitrogen or oxygen atoms are the following:

$$\begin{cases} \alpha = 70° \\ \beta = 1.95\text{Å} \\ \gamma = 0.45\text{Å} \end{cases}$$

These parameters are fuzzy enough to catch hydrogen bonds that are far from the most stable configuration. However it is assumed that local conformational changes may lead to a stabilization of the hydrogen bond. It shoud be recalled that the purpose of this approach is to keep hydrogen bond donors and acceptors that are ready to interact with an external partner and to ignore others.

Parameter l is the length of the covalent bond between the δ^+ hydrogen atom and the heavy atom. This distance is imposed: the location of the hydrogen atom is calculated from the direction of the covalent bond and from l, since PDB files usually don't specify this location. The default value for l is 1 Å. The computation of the location of hydrogen atoms is therefore performed when necessary. For some groups such as $–NH_3^+$, choosing the location of the 3 hydrogen atoms is more tricky because of the free rotation of the chemical group. This positioning is performed while considering each of the rotations that forms an optimal hydrogen bond. The rotation which is chosen is the one that maximizes the number of hydrogen bonds.

Definition of ligand The notion of *ligand* as used in the context of SuMo is based on the following criteria:

- a ligand is an individual molecule;
- a ligand may form temporary complexes with biological macromolecules;
- a ligand for a macromolecule should not be necessary for that macromolecule to adopt a stable fold;
- a ligand is not a macromolecule.

The notion of "temporary complex" between a ligand and a macromolecule refers to an interaction without covalent bonds that can be broken without globally destructuring the macromolecule. According to this definition, a ligand is thus not required to stabilize a plurimolecular structure.

For automatically defining ligands in SuMo, the following criteria are used:

- size smaller than 50 non hydrogen atoms,
- at least one of the monomers is not a well-known monomer.

We call *well-known monomer* one of the following:

- one of the 20 classic amino acids ALA, CYS, ASP, GLU, PHE, GLY, HIS, ILE, LYS, LEU, MET, ASN, PRO, GLN, ARG, SER, THR, VAL, TRP, TYR;
- selenomethionine MSE;

- water HOH.

Nucleotides are not considered as well-known monomers. Oligonucleotides will therefore be considered as ligands when they match the size condition. The list of well-known monomers is given in the file that defines SuMo's chemical groups (see section A page 145).

The ligands are named after the sequence of the PDB identifiers of their monomers, ordered by chain identifier and by chemical group number (e.g. CA, GLC-GLC, ATP, ...). This nomenclature has the advantage of being readable but is ambiguous in certain cases. Indeed the monomers of some ligands such as oligosaccharides may be bound in different ways.

3.2.1.2 Chemical groups

In SuMo, the notion of chemical group is fundamental. Every macromolecular structure is first irreversibly converted into a set of chemical groups. Several *types of chemical groups* may be defined. Each of them will represent a given property. Only chemical groups of the same type can be compared and possibly considered as equivalent.

Types An identifier such as `hydroxyl` or `delta_minus` is given to each type of chemical groups. A type of chemical groups must be watched as the representation of a chemical property that can be localized in 3D structures of macromolecules such as proteins. All the chemical groups of a given type must follow the same geometrical model so that they could be compared and possibly superimposed.

Coefficients Each type of chemical groups is given a coefficient. This coefficient depicts the functional importance of this type of chemical groups. It is used in different heuristics inside of SuMo. It is a positive number lower or equal to 1. Any distance which can be seen as a *radius of influence* around a chemical group is beforehand multiplied by this factor.

Positional informations Each chemical group is localized in space. This localization relies on 3 kinds of points:

- the physical location,
- the functional location,
- target locations.

The physical location The *physical location* is a mean position of the physical components—usually atoms—that are responsible for the existence of the chemical group.

This location is used in particular for selecting triangles of chemical groups that will be used to represent the 3D structures.

The functional location The *functional location* is the localization of the functional property of the chemical group. It may or may not be equal to the physical location. This is the location which is used for comparing 3D structures in SuMo. For instance, the functional location of hydrogen bond donor is the location of the potential hydrogen bond acceptor.

The target locations The *target locations* are the positions that should be considered for characterizing the interaction with a ligand. They shall indicate the most probable relative positions of the ligand.

This allows to define relevant sites of interaction, especially ligand binding sites.

For example, the `aromatic` group which is defined in the current version of SuMo has 2 symmetric target locations. They are located on each side of the aromatic ring. For an hydrogen bond donor, there will be one target location which is equal to the functional location.

Shared additional data Some data are defined for any of the types of chemical groups but are optional, i.e. they will be used in the comparison process but their removal does not prevent the core comparison algorithm from being used.

Burial The *burial of a chemical group* is based on the computation of atomic density around its functional location. The properties of the density function that is used are described in details section 3.6.1 page 78.

Orientation Orientation against the surface of the macromolecule is given by the $\overrightarrow{CP}$ vector where P is the functional location of the chemical group and C is the barycenter of the atoms that are considered for the calculation of atomic density. C is called *local center of mass.*

The atomic environment The atomic environment is the set of atoms that surround the functional location, within a radius which is large enough to allow the shape comparison (see sections 3.2.5.1 page 64 and 3.6.2 page 79).

Type-specific data Each type of chemical group may be enhanced with any kind of additional data that could be used in the pairwise comparison of chemical groups. However such modifications cannot be performed by the user without extending the program.

In order to easily define useful types of chemical groups without reprogramming `sumo`, the SuMo provides a language for defining types of chemical groups. These definitions are put into a file that is read at the initialization of any `sumo` process. This language provides a set of predefined type contructors.

To each type of chemical group is constructed with a given parametrized *geometric variant.*

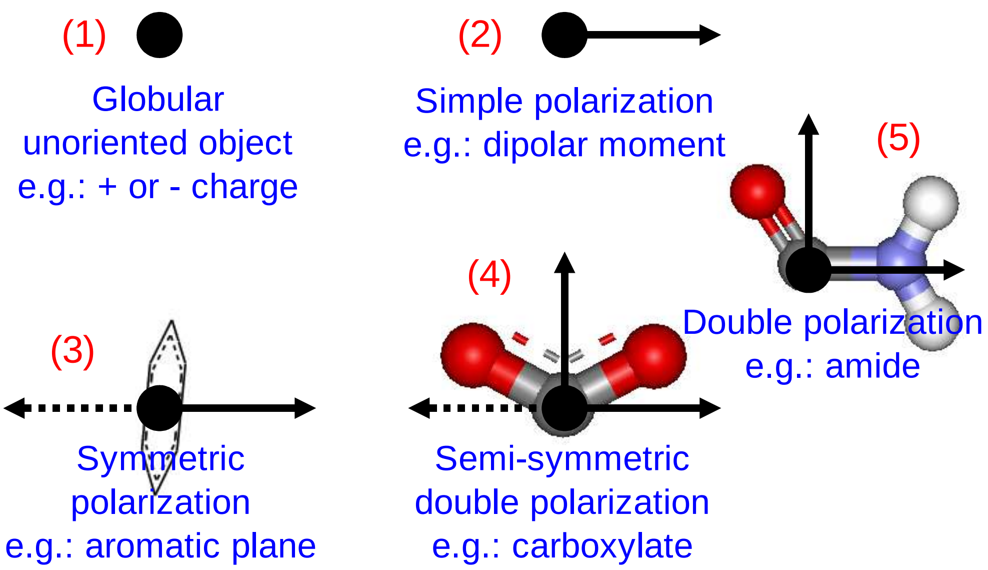

(1)
Globular
unoriented object
e.g.: + or - charge
(2)
Simple polarization
e.g.: dipolar moment
(5)
Double polarization
e.g.: amide
(4)
(3)
Symmetric
polarization
e.g.: aromatic plane
Semi-symmetric
double polarization
e.g.: carboxylate

Figure 3.8: Examples showing the 5 types of geometric variants that are defined and implemented in SuMo.

The geometric variants Currently 5 geometric variants are defined and implemented in SuMo. Figure 3.8 page 50 shows examples of chemical groups for each of these variants. These geometric variants are numbered from 1 to 5:

Variant 1 is the null variant: it does not contain any specific geometric information. All directions of space are equivalent.

Variant 2 consists in one vector, without symmetry.

Variant 3 is composed of 2 opposite vectors that are functionally equivalent.

Variant 4 is composed of 2 orthogonal vectors, and one of them is functionally equivalent to its opposite.

Variant 5 is composed of 2 orthogonal vectors.

The different predefined functions that are provided for defining types of chemical groups are called *type constructors*. Each of these constructors derives one or several chemical groups when a pattern in the atomic structure of the molecules is matched. One given constructor will generate chemical groups that always have the same geometric variant. However, several different constructors may be used to define chemical groups that share the same type. Table 3.2 page 51 shows a list of the constructors that are defined in version 4.4 of SuMo.

Type constructor	Geometric variant
`Point`	1
`Polar`	2
`Biplan`	3
`Plan`	4
`Chiral`	5
`Virtual`	1
`Delta_plus`	2
`Delta_plus_plan`	2
`Delta_plus_multiple`	2
`Delta_minus`	2

Table 3.2: Type constructors and geometric variants for bulding chemical groups.

The main characteristics of the constructors of types of chemical groups are the following:

- `Point` generates chemical groups without specific geometry from a mean position of atoms (type 1 variant).
- `Polar` generates a type 2 variant.
- `Biplan` generates a type 3 variant.
- `Plan` generates a type 4 variant.

- `Chiral` generates a type 5 variant.
- `Virtual` generates a type 1 group with functional and target location different from the physical location. The group is generated only if the space around functional location is empty.
- `Delta_plus` generates a chemical group which is a free hydrogen bond donor, from 2 positions.
- `Delta_plus_plan` generates a chemical group which is a free hydrogen bond donor, from 3 positions and an angle (e.g. amide $-CONH_2$).
- `Delta_plus_multiple` generates several chemical groups which are free hydrogen bond donors, from 2 positions, one angle and the number of hydrogen atoms (e.g. $-NH_3^+$).
- `Delta_minus` generates a chemical group which is a free hydrogen bond acceptor, from 2 positions and from the number of hydrogen bonds that make it saturated.

The parameters Each constructor of types of chemical groups expects parameters. Some of the parameters are optional and can take a default value. The parameters include:

- the definition of target locations,
- the translation from the physical location to the functional location,
- maximum angular deviations that are allowed in the comparison process.

Annotation of chemical groups Complementary information may be added to the chemical groups without being a requirement for the comparison heuristics.

PDB group Most of the time it makes sense to associate a SuMo chemical group non exclusively with a *PDB group*. This information is composed of the chain identifier (e.g. A), the name of the PDB group (e.g. ALA) and its number. In SuMo version 4.4, the process that defines SuMo chemical groups automatically records this information for every chemical group.

For conveniency, this information can be used as a criterion for selecting chemical groups (see section 3.7.1.3 page 101). However, this information is too ambiguous to be used for the determination of the different structural constituents of a macromolecular complex.

Molecule The different molecules that are identified during the preprocessing of the structural data (section 3.2.1.1 page 44) are given a number. Each chemical group is associated with a *molecule* when it is meaningful (it is always the case with the current definition of chemical groups).

This information can be used to select some specific chemical groups. This means of selection nevertheless requires the number that identifies the molecule. Currently this functionality is used for the automatic selection of ligand binding sites.

Labels Chemical groups may be labeled in order to differenciate each of them when the other criteria are not enough. For example, we can define up to three free hydrogen bond acceptors for a given aspartate: one in the main chain and two in the carboxylate group. These chemical groups all belong to the `delta_minus` type, belong to the same polypeptide and to the same amino acid. Therefore we will give them the `backbone`, `e1` and `e2` *labels* so that these chemical groups can be identified and selected without ambiguity.

A label can be used as a selection criterion exactly like the informations that were mentioned in the previous paragraphs. Currently, an automatic labeling scheme is defined together with the definition of the types of chemical groups.

Arbitrary tagged regions Arbitrary sets of chemical groups from a given 3D structure can be annotated. These annotated regions are then displayed with the results. Automatic tagging of ligand binding sites is performed when the database of complete structures is built. Explicit, manual tagging can be performed by the user of SuMoQ queries.

Thus, each chemical group is tagged as belonging to a list of annotated regions. The annotation is an arbitrary character string and some characteristics concerning their structure and their size are automatically computed and recorded. How to define annotated regions is explained in the section about SuMoQ, page 104.

Syntax for defining types of chemical groups The language that has been created is quite complex. We only give the main syntaxic rules. The full listing of the file that is used for the definition of chemical groups in SuMo version 4.4 is given in the appendix page 145.

The definition of types of chemical groups is based on the notion of pattern, at different levels. `<<"ALA" "GLY">>` means (according to the PDB naming standard) "the current PDB group must be either an alanine or a glycine". `<<"ALA">>[-1]` means "the previous PDB group in the sequence is an alanine". Similarly, `<"N" "C" "CA">` means "the given PDB group must have one atom which name is N, C or CA". When a pattern of the `aa.atom` form is matched, the position of the first atom that matches the pattern is returned. A sequence `aa1.atom1 aa2.atom2` returns the mean position of the two atoms that match the pattern. Here is a commented example of a file which has a valid syntax:

```
(* This is a comment *)

(* These are the types of chemical groups that we want to use: *)
Export (my_first_chemical_group
        delta_minus);    (* my_favorite_chemical_group
                            is defined but not used *)

Not_special ("HOH");   (* indicates that water is not an
                          interesting monomer for a ligand *)

aa = <<"ALA" "GLY">>;   (* definition of a set of PDB groups named aa *)
arg = <<"ARG">>;

my_first_chemical_group = (* this is the identifier of the new type of
                             chemical groups *)

    Delta_plus [backbone]    (* [backbone] is an optional label *)
          (aa.<"N">,      (* matches atoms "N" from group "ALA" or "GLY" *)

           aa[-1].<"C"> aa.<"CA">,  (* aa[-1] means any aa in the same chain
                                       with index = current_index - 1 *)
           aa.<"N">,
           target = 0.0,            (* optional field *)
           functional_shift = 2.8,  (* optional field *)
           angle = 140.)            (* optional field *)

  | Delta_plus         (* other pattern *)
             (arg.<"NE">,
              arg.<"CD"> arg.<"CZ">, (* the point is the average of
                                        the 2 atoms matched with
                                        arg.<"CD"> and arg.<"CZ"> *)
              arg.<"NE">,
              target = 0.0,
              functional_shift = 2.8,
              angle = 140.)
;

delta_minus {0.6} =  (* Weight is 0.6, not 1 *)
  Delta_minus [backbone] (aa.<"O">,
                          aa.<"C">,
                          aa.<"O">,
                          1,
                          target = 1.)
;

my_favorite_chemical_group =
  Point                         (* Point is the simplest constructor *)
    (<<"PHE">>.<"CA"> <<"PHE">>[1].<"CA">);


Hbond (my_first_chemical_group, delta_minus); (* we define a hydrogen bond,
                                                 using the default
                                                 parameters *)
```

Phantom chemical groups The *phantom chemical groups* are chemical groups that are automatically generated by SuMo independently from the file containing the definition of normal chemical groups. These phantom groups are not taken into account during the comparison. The purpose of having such groups is to represent the atoms of ligands so that they could be used in some selection processes (see section 3.7.1.3 page 101).

Any atom which is not known to be a hydrogen and which belongs to a PDB group that has not been declared as well-known will constitute a phantom chemical group. A well-known PDB group is a group that has not been selected using the `Not_special` directive. Its location is the position of the atom. Its type is automatically generated from the name of the PDB group as in the following examples:

$$\begin{aligned} \texttt{GLC} &\longrightarrow \texttt{pdb\_glc} \\ \texttt{A} &\longrightarrow \texttt{pdb\_a} \end{aligned}$$

3.2.2 Association into triplets

The comparison heuristics that has been designed for SuMo is based on the construction of triplets of chemical groups. These objects contain different kinds of information and are the vertices of a graph that is the main data structure which is used for comparing 3D structures in SuMo.

This section describes which triplets are chosen, what kind of information they contain and how the graph of triplets is built.

3.2.2.1 Choosing the triplets

The use of triplets is motivated by the fact that they represent micro-sites of fixed size which are easily compared and contain more information than a chemical group alone. Triplets are chosen so that they match structural units that have a potential functional relevance. Their size, i.e. the length of the edges of the triangles should not be too large.

Given n triplets of chemical groups that represent the 3D structure of a molecule, the number of triplets that can be generated is roughly equal to n^3. Because of practical issues, it is not a good idea to generate all possible triplets in an average macromolecule, where the number of chemical groups frequently exceeds 1000.

For the previous reasons, the number of triplets that will be selected will be much smaller than the number of possible triplets. The criteria for selecting the triplets are defined by hand so that the following constraints are satisfied as much as possible:

- representing as many functional sites as possible
- representing as less as possible sites that are not related to a specific biological function,
- comparing fast enough so that screening databases is possible,
- storing and accessing the data within reasonable physical limits.

For this, a limited number of triplets must be generated while trying to keep those that have the more chances to be related to an interesting biological phenomenon.

We will call *physical triangles* the triangles that are made of the physical locations of the chemical groups in the triplets, and *functional triangles* or simply triangles the triangles that are made from the functional positions of the chemical groups.

The selection criteria and their parameters are given in the following text for SuMo version 4.4 but change frequently from a version of SuMo to another.

Length of edges The larger the chemical group is, the broader influence it will have on its environment. Thus, the selection criteria concerning the length of the edges depend on the coefficients that were assigned to the chemical groups. Given 2 chemical groups of physical locations p_1 and p_2

with coefficients k_1 and k_2, any triplet that contains these chemical groups will have to satisfy the following constraint:

$$\frac{k_1 + k_2}{2} \cdot l_{\min} \leq \|p_1 - p_2\| \leq \frac{k_1 + k_2}{2} \cdot l_{\max}$$

where the value of $l_{\max}$ is 8.5 Å and the value of $l_{\min}$ is $0.3.l_{\max}$.

Sum of edge length In order to avoid the creation of triplets of large size with vertices that already belong to enough small triplets, the sum of the length of the edges must be lower than a given threshold.

A minimum size is also required in order to avoid the creation of triplets of chemical groups that all belong to the same rigid fragment of molecule.

Thus any triplet of physical locations p_1, p_2 and p_3 must satisfy the following constraint:

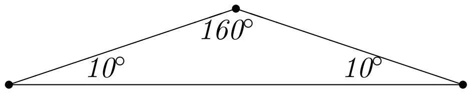
$$s_{\min} \leq \|p_1 - p_2\| + \|p_2 - p_3\| + \|p_3 - p_1\| \leq s_{\max}$$

where the value of $s_{\min}$ is 6 Å and the value of $s_{\max}$ is 20 Å.

Angles The definition of a plane from 3 positions is less precise and less relevant if these points are almost aligned. Here the functional location of chemical groups is used[11] since this is the one which is used when comparing triplets.

In order to avoid almost linear triangles, the angles of the triangles must be neither too large nor too small. Each angle ranges within 10° and 160°.

The following triangle is the limit:



3.2.2.2 Properties of the triplets

A triplet of chemical groups is the minimal entity that is used while comparing 3D structures of macromolecules in SuMo. Each object has of course three chemical groups with all the properties that were described previously, but also higher order properties that we are going to describe.

---

[11] SuMo version 4.5

Triplet coordinate system Each triplet T is associated with a triangle which is made of the functional positions of the chemical groups. For each triplet T, we will define a coordinate system R_T from the functional positions p_1, p_2 and p_3. The purpose is to be able to superpose 2 triplets T_1 and T_2 just by expressing the coordinates of the points respectively in R_{T_1} and in R_{T_2}.

Let O be the mean position of p_1, p_2 and p_3. O is the center of the coordinate system R_T. Let v_1, v_2 and v_3 denote the vectors $\overrightarrow{Op_1}$, $\overrightarrow{Op_2}$ and $\overrightarrow{Op_3}$. Let u be the unit vector that characterizes the plane defined by p_1, p_2 and p_3 so that the triple scalar product (v_1, v_2, u) is positive. We call *standard vectors* of the triplet T the coplanar unit vectors $\tilde{v}_1$, $\tilde{v}_2$ and $\tilde{v}_3$ such that the following sum S is minimum:

$$S = \widehat{v_1, \tilde{v}_1} + \widehat{v_2, \tilde{v}_2} + \widehat{v_3, \tilde{v}_3}$$

where $\widehat{a, b}$ is the non-oriented angle between vectors a and b, which is given by $\cos^{-1} \frac{a.b}{\|a\|.\|b\|}$. Standard vectors $\tilde{v}_1$, $\tilde{v}_2$ and $\tilde{v}_3$ are obtained by rotating vectors v_1, v_2 and v_3 around the axis defined by u respectively by the angles α_1, α_2 and α_3:

$$\begin{aligned}
\alpha_1 &= \frac{1}{2}\left(\widehat{v_1, v_2} - \widehat{v_3, v_1}\right) \\
\alpha_2 &= \frac{1}{2}\left(\widehat{v_2, v_3} - \widehat{v_1, v_2}\right) \\
\alpha_3 &= \frac{1}{2}\left(\widehat{v_3, v_1} - \widehat{v_2, v_3}\right)
\end{aligned}$$

The coordinate system R_T of triplet T is the cartesian coordinate system defined by $(O, \tilde{v}_1, u \wedge \tilde{v}_1, u)$ where $\wedge$ is the cross product.

Properties from chemical groups Several of the properties of the chemical groups can be transposed to the whole triplets.

Type of the triplets The *type of a triplet* of chemical groups is the ordered triplet of the types of the chemical groups. For instance, a triplet made of groups `delta_plus`, `aromatic` and `guanidinium` will have type (`aromatic`, `delta_plus`, `guanidinium`).

Depending on the type of the triplet, a number of permutations which is equal to 1, 2 or 6 exists. 1 permutation when the types of the chemical groups are all different, 2 permutations when 2 of them are equal or 3 permutations when their are all equal.

Orientation of the chemical groups The geometric variants from the chemical groups are kept and expressed in the coordinate system of the triplet.

Atomic environment The atomic environment of every chemical group is merged into one set and their coordinates are expressed in the new system.

> *Given that the position of the surrounding atoms is currently used only to define the shape of the local environment and that the heuristics that is used to compare shapes is not sensitive to the redundancy of the points, there is not a strict requirement for removing the redundant positions of atoms.*

The volume of the atomic environment is precomputed since it is used in the heuristics for shape comparison.

Length of edges The distances between the vertices of the functional triangle of a triplet are recorded in order to be used during the comparison.

Orientation against the molecular surface Each chemical group is associated with a point which is called local center of mass (section 3.2.1.2 page 49). Given a triplet T, the local center of mass is defined as the mean position C of the local centers of mass of the chemical groups located at P_1, P_2 et P_3. The scalar triple product $(\overrightarrow{CP_1}, \overrightarrow{CP_2}, \overrightarrow{CP_3})$ is computed and recorded. It characterizes in some way the orientation of the plane of the triangle against the surface of the macromolecule.

3.2.3 The graph of triplets

The representation of macromolecules that is used for their comparison consists of a graph in which the vertices model triplets of chemical groups and the edges connect triplets that match specific proximity criteria.

3.2.3.1 Vertices

Each vertex of the graph stands for a triplet of chemical groups. Depending on the number of permutations that are associated with this triplet, 1, 2 or 6 subvariants of the triplet are generated.

3.2.3.2 Edges

Edges connect the triplets with functional triangles that are *quasi-adjacent*, i.e. given 2 triplets of positions (p_1, p_2, p_3) and (q_1, q_2, q_3), there must be two and only two pairs of indices (i_p, i_q) and (j_p, j_q) such that:

$$\begin{cases} i_p \neq j_p \\ i_q \neq j_q \\ \left\| p_{i_p} - q_{i_q} \right\| \leq d_{\max} \\ \left\| p_{j_p} - q_{j_q} \right\| \leq d_{\max} \end{cases}$$

where $d_{\max}$ is a distance which is constant and generally small compared to the length of the edges of the triangles. Triangles (p_1, p_2, p_3) and (q_1, q_2, q_3) are then said to be quasi-adjacent at (p_{i_p}, q_{i_q}) and (p_{j_p}, q_{j_q}). This property is illustrated on figure 3.9 page 60.

> *In SuMo version 4.4, the value of* $d_{\max}$ *is 1 Å.*

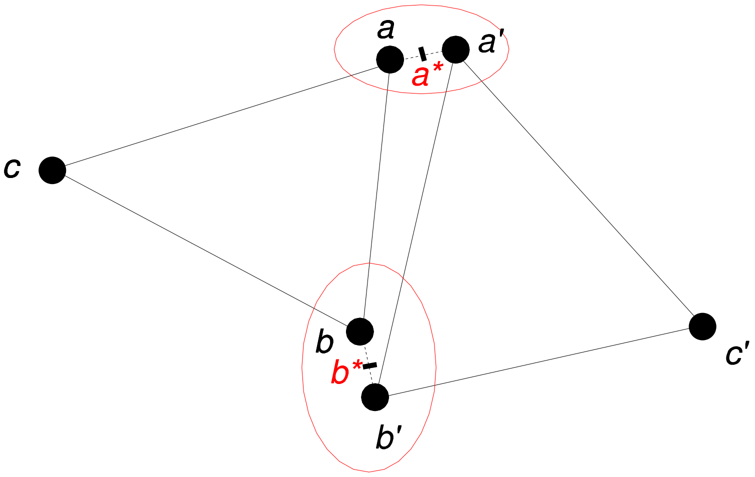


Figure 3.9: Example of triangles (a, b, c) and (a', b', c') which are quasi-adjacent at (a, a') and (b, b').

Each edge contains the angle between the 2 quasi-adjacent triangles. This angle can *a priori* range from $-\pi$ to π modulo 2π. It is defined using the pairs of points that are responsible for the quasi-adjacence of the triangles. Let (a, b, c) and (a', b', c') be 2 triangles that are quasi-adjacent at (a, a') and

(b, b'). The mean positions $a^\star$ et $b^\star$ are defined as follows:

$$\begin{aligned} a^\star &= \frac{1}{2}(a + a') \\ b^\star &= \frac{1}{2}(b + b') \end{aligned}$$

Let w be a shortcut for $\overrightarrow{a^\star b^\star}$. w defines an oriented plane $\mathcal{P}$. Let v_1 and v_2 be the vectors that define the oriented planes that are respectively given by (a, b, c) and (a', b', c'). Let us call $\tilde{v_1}$ and $\tilde{v_2}$ their projected image according to w. The θ angle between the 2 quasi-adjacent triangles (a, b, c) and (a', b', d) is then defined as the angle between vectors $\tilde{v_1}$ and $\tilde{v_2}$ in the oriented plane $\mathcal{P}$. If the order of a and b and of a' and b' is conserved such as in (b, c, a) or (c', a', b'), then the angle between the quasi-adjacent triangles is the same. But if there is an inversion in either of the triangles as in (b, a, c), then the angle will be $\theta + \pi$.

> *The version 4.4 of SuMo does not set constraints on the value of this angle to allow the creation of an edge in the graph.*

3.2.4 Data storage

The time which is required for the generation of the graphs of triplets of chemical groups from a 3D structure of proteins is generally a few seconds. If the average time is 5 seconds and that we wish to compare a given 3D site against each of the 20,000 proteins of the PDB, it would take more than 24 hours just to generate the graphs. In practice, graphs are pregenerated and this kind of process takes less than one hour to return the full comparison results.

3.2.4.1 Format

Graphs of triplets—plus a few textual information coming from the original PDB files or site annotations—are stored into files under a form which is easily readable by the `sumo` program.

This conversion from and to a binary file are performed with the functions of the Marshal module of Objective Caml (section 2.3.3 page 30).

3.2.4.2 Compression

The large volume of the database can be reduced if the data may be compressed at low computational cost. Compressed data will save some disk space, but also some time when they have to be transfered through a network, e.g. when stored on a remote file system using NFS.

Direct use of a compression tool such as `gzip` does not result in a satisfying compression rate over data in the Marshal format. A large part of the data is indeed composed of 64 bits IEEE 754 floating point numbers. 51 of the 64 bits are used for the mantissa, which results in relative precision of 2^{-52} which is approximately 2.10^{-16}. But the coordinates which are given in PDB files cannot have more than 7 decimal digits which gives a relative precision of 5.10^{-8} in the best case. This relative precision can be written as $2^{-24,3}$ which means that 24 bits are enough to represent all the significant information since 24 bits result in a precision of 2^{-25}. Therefore, the floating point numbers may be rounded so that only the 24 first bits are used. The other 27 bits of the mantissa are set to 0. In the Marshal format, these bits will still appear as 27 consecutive 0 bits. This kind of data will be efficiently compressed by `gzip` and save 45% of space compared to the compression without rounding.

> *This compression scheme is sensitive to large translations of the data. Thus it is usable only for a maximum of 7 significant digits. This is the case in all PDB files at least because of the syntaxic constraints, but it can become wrong with other file formats. For instance 1,000,003.27 would be rounded to 1,000,003.*

For example, the size of an array of 10,000 floats that are randomly chosen between 0 and 10 is given in the following table:

Taille	No compression		Compression by `gzip`	
	Not rounded	Rounded	Not rounded	Rounded
Bytes	80,025	80,025	76,270	42,134
Ratio	100%	100%	95%	53%

3.2.5 Core comparison

The comparison of 3D structures of macromolecules consists in identifying similar regions in 2 graphs of triplet of chemical groups.

3.2.5.1 Similar triplets

The first step in comparing graphs of triplets is the search for pairs of similar triplets. Several criteria are used successively in order to test whether a pair is acceptable or not. Every criterion must be met. As an optimization, tests are performed in a specific order, which the order in which the following paragraphs appear.

Preliminary note If several permutations must be taken into account because the triplets that are being compared have 2 or 3 identical chemical groups, then each permutation of the first triplet must be considered against one of the permutations of the second triplet. For instance, if we have to compare the following triplets

$$\left(x_1^{\texttt{(acyl)}}, x_2^{\texttt{(acyl)}}, x_3^{\texttt{(hydroxyl)}}\right) \text{ et } \left(y_1^{\texttt{(acyl)}}, y_2^{\texttt{(acyl)}}, y_3^{\texttt{(hydroxyl)}}\right)$$

then it will be necessary to test the other possible permutation of the first triplet as in

$$\left(x_2^{\texttt{(acyl)}}, x_1^{\texttt{(acyl)}}, x_3^{\texttt{(hydroxyl)}}\right) \text{ and } \left(y_1^{\texttt{(acyl)}}, y_2^{\texttt{(acyl)}}, y_3^{\texttt{(hydroxyl)}}\right)$$

Nevertheless, the steps that do not try to match the vertices of the triangles do not have to test all possible permutations.

Triplets of the same type Only triplets of the same type can be considered as similar. Let us recall that the types of the triplets are ordered according to the type of chemical groups.

Let us suppose that there are approximately 1000 types of possible triplets, and that these triplets have all the same frequency in each structure of macromolecule. The number of comparisons of triplets that should be performed naively when comparing two structures of size n_1 and n_2 is $n_1.n_2$. However a significant amount of time can be saved if for both structures the triplets are grouped by type and then compared only against the matching group in the other structure. The number of comparisons that is performed is then approximately $1000 \cdot \frac{n_1}{1000} \cdot \frac{n_1}{1000}$ when n_1 and n_2 are large which is simplified into $\frac{n_1.n_2}{1000}$ instead of $n_1.n_2$ in the naive version.

Length of the edges The length of the edges of the matched triangles must be close. The relative deformation from one edge to another must be below a given threshold. For any pair of matched edges of length d_1 and d_2, the *relative deformation* is defined by the following δ function:

$$\delta(d_1, d_2) = \frac{|d_1 - d_2|}{\frac{1}{2}(d_1 + d_2)}$$

> *The δ function is the same as the one used in the general definition of the relative deformation (definition 3.12 page 88), where the distance of reference is implicitely set to the mean of d_1 and d_2.*

The maximal value for the relative deformation which is accepted is 0.25 in SuMo version 4.4.

Placement of the plane The triple scalar product which has been precomputed and which gives a notion of how the plane of the triangle is oriented against the rest of the macromolecule is compared. The maximum deviation is 50 $Å^3$ in SuMo 4.4.

Burial The burial which is estimated by the atomic density is compared. The maximal deviation of this property is 0.08 $g.mol^{-1}.Å^{-3}$ in SuMo 4.4.

Orientation of the chemical groups The orientation of the chemical groups is given through their geometric variants. We saw that the points and the vectors that belong to the geometric variants have been expressed in a triplet-centered system of coordinates. Thus, the superposition of the triangles is already performed.

Among the geometric variants that are currently provided in the implementation of SuMo, only unit vectors define the orientation of the chemical groups. These vectors are ready to be compared. The angle between these vectors must be below a given threshold. Each threshold is given when the types of chemical groups are defined. The full definition of the types of chemical groups is given in the appendix page 145.

Comparison of local shape This operation consists in comparing the shape of the environement from the positions of the surrounding atoms which coordinates are expressed in the triplet-centered system. The heuristics for shape comparison is described in details section 3.6.2 page 79. This the most CPU-consuming operation that is involved in the comparison of triplets. This is also the last one and will be performed only if all the other tests have been successful.

Like in the previous tests which require the triangle vertices to be matched, all the permutations of one of the triplets must be considered.

The minimal score is 0.65 on a scale that ranges from 0 to 1.

3.2.5.2 Connection of pairs

The next step of the comparison is the creation of what we will call a comparison graph. A graph is built in which the vertices are the matched triplets that have been identified at the previous step. Two conditions are required to connect two vertices of the comparison graph: double neighboring and conservation of the angle between triplets.

Double neighboring The pairs (x_1, y_1) and (x_2, y_2) of similar triplets can be connected if x_1 and x_2 are connected in their original graph of triplets, as well as y_1 and y_2 in their respective graph of triplets.

x_1 and x_2 might not be distinct (idem for y_1 and y_2).

Angle between triplets Let (x_1, y_1) and (x_2, y_2) be two pairs of similar triplets that satisfy the previous condition of double neighboring. Edge x_1x_2 in the first graph of triplets and edge y_1y_2 in the other graph of triplets must have a similar angle. Let these angles be denoted by θ_1 and θ_2. Let us recall that these angles are comprised between $-\pi$ and π. The difference $|\theta_1 - \theta_2|$ between these angles modulo 2π must be lower than a given λ threshold.

In SuMo version 4.4, the λ threshold is 40°.

3.2.5.3 Isolation of independent subgraphs

The next step of the comparison consists in extracting the independent subgraphs from the comparison graph.

Obtaining independent subgraphs The independent subgraphs from the comparison graph are extracted by using the algorithm 1 page 45.

Nature of the result The result is a set of independent subgraphs. Each subgraph is a list of matched triplets of chemical groups. The pairs of triplets are converted into pairs of chemical groups: at this point the notion of triplet disappears. Finally, the result is a non-redundant list of pairs of chemical groups. However, each chemical group can possibly belong to several pairs of the same list such as in

$$\{(g_1, g_1'), (g_2, g_2'), (g_3, g_1'), (g_4, g_4')\}$$

where g_1' appears twice.

3.2.5.4 Filtering

The lists of matched chemical groups are then filtered. This step is a way to satisfy some conditions that cannot be satisfied earlier. For example such a condition could depend on the whole list of matched elements, such as the RMSD or the volume as defined in section 3.6.2.3 page 83.

Contrary to the versions 1 and 2 of SuMo, versions 3 and 4 do not modify the lists of matched elements. Some lists are kept, others are discarded but none is modified anymore. SuMo indeed considers that the comparison heuristics is natural since triplets of chemical groups model microsites of relative rigidity while larger sites do not have to be superposable. Each triplet models a potential anchor for a rigid piece of a flexible ligand.

Notion of functional flexibility We will call *functional flexibility* the diversity of sites that bind a same given ligand by using the same kind of interactions. The sites themselves may not be flexible and may not be able to bind the same conformation of the ligand.

Thus the functional flexibility of some ligand binding sites implies the non-superposability of these sites at the points that are responsible for the interaction with the ligand. For this reason, since SuMo version 4 the RMSD is not used as a criterion for estimating the level of similarity between 3D sites.

Deformation instead of superposition The notion of deformation replaces the one of superposition. The purpose of this notion is to give more penalty to the local deformations while accepting that small local deformations can contribute to the non-superposability of sites while having the same functional properties.

A sophisticated definition of deformation and its calculation is given section 3.6.3 page 85. It takes into account the deformation of distances between elementary objects, but also the changes in their orientation by taking into account their possible symmetry.

The global estimation of deformation of a list of matched chemical groups considers at a larger scale some criteria that were already used when comparing triplets. These criteria are the relative deformation of the length of the triangle edges and the similarity in the orientation of the geometric variants after superposition.

The acceptable threshold after the global computation of the deformation must be stricter than the threshold that was used earlier while comparing triplets. This allows to constrain the mean deformation while allowing locally a few larger deformations.

The maximal deformation which is allowed in SuMo 4.4 is 0.15.

3.3 Multiple comparison

A system for performing multiple comparison of 3D structures or substructures has been designed and implemented in SuMo. However it does not depend exclusively on SuMo and could be used on the output of any tool that identifies lists of matched objects that are localized in space such as chemical groups.

3.3.1 General principle

The system for multiple comparison that has been designed does not perform structural alignment. It should be viewed as a system that searches for families of sites that are shared between different 3D structures.

The first step when comparing n 3D structures consists in obtaining the lists of matched chemical groups between all of these structures by performing the $n.(n-1)/2$ possible pairwise comparisons.

Then in each structure, the sites which are significantly overlapping are grouped together.

The final step of the comparison consists in setting up the families of sites that are shared by the different structures.

3.3.2 The steps of the multiple comparison

Figure 3.10 page 68 shows the different steps of the multiple comparison by representing the different kinds of objects that are used:

- 3D structure
- chemical groups
- matched chemical groups
- sites
- matched sites
- characteristic sites
- matched characteristic sites
- families of characteristic sites

3.3.2.1 Obtaining matched chemical groups

The comparison of each pair of 3D structures results in lists of matched elements. Each of these lists is associated with a specific color on figure 3.10. Each line on figure 3.10B indicates a matched pair of chemical groups.

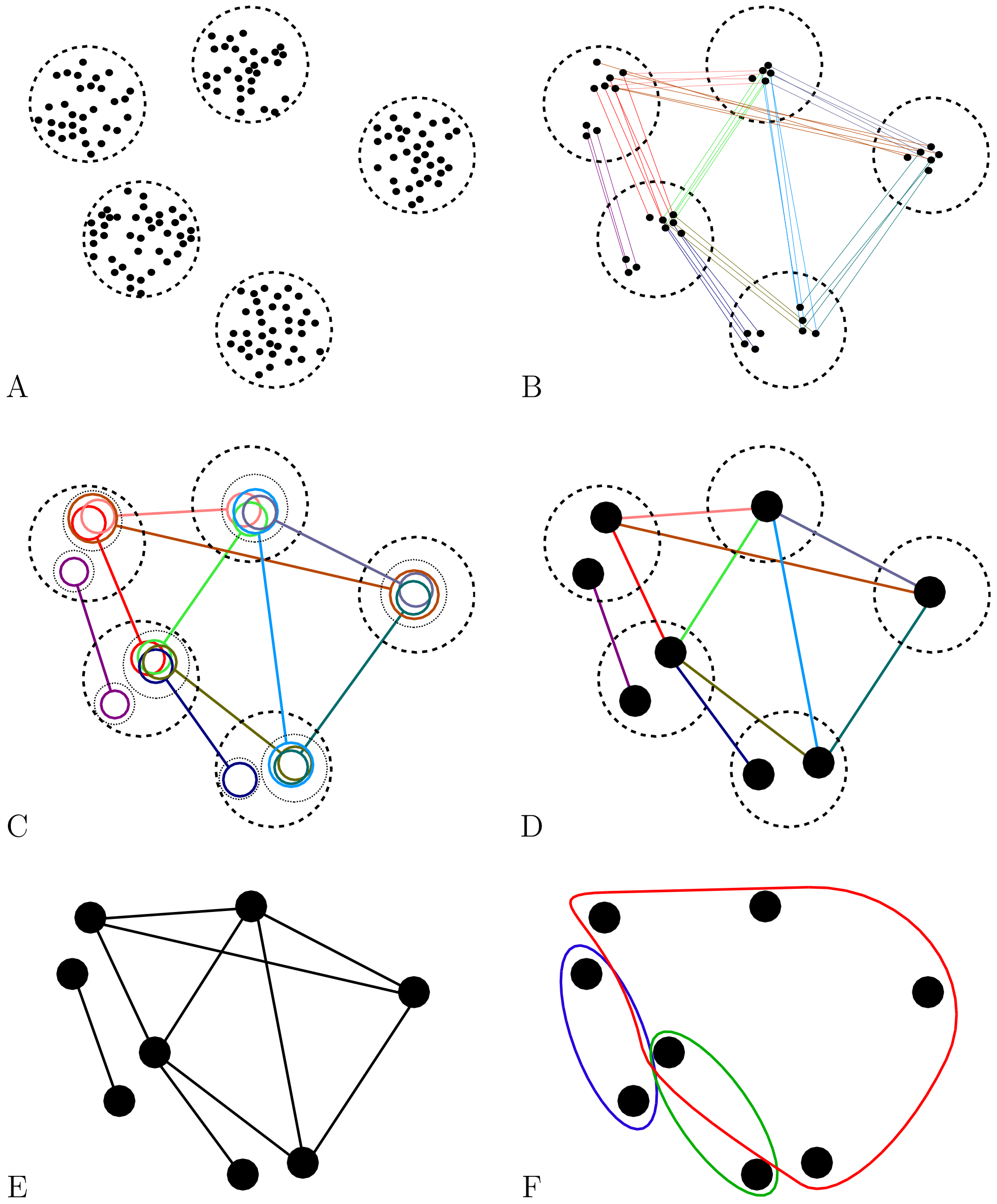


Figure 3.10: Steps of the multiple comparison of structures of macromolecules. **A**: 5 3D structures represented by chemical groups. **B**: search for matched chemical groups between all pairs structures. **C**: notion of site and of matched sites. **D**: grouping sites that are significantly overlapping into characteristic sites. **E**: graph of characteristic sites. **F**: 3 families of characteristic sites.

3.3.2.2 Obtaining sites

Each list L of matched elements will result in a pair (A, B) of sites defined by:

$$\left\{ \begin{array}{lcl} A & = & \{a|(a,b) \in L\} \\ B & = & \{b|(a,b) \in L\} \end{array} \right.$$

We will say that sites A and B are matched.

3.3.2.3 Grouping significantly overlapping sites

For each 3D structure, several sites have been identified by comparison with other structures. For a given structure, all the sites will be grouped if they are considered as enough overlapping. One given site may be put into different groups. A group of sites is called *characteristic site.*

Overlap between 2 sites For two sites A and B, a score is calculated:

$$\phi(A, B) = |A \cap B| - k.\min(|A|, |B|)$$

where k is a constant that we will call *overlap factor.* k is a number between 0 and 1. A and B are said to be overlapping when $\phi(A, B) \geq 0$.

The overlap factor gives the minimum overlap between two sets relatively to the smallest of these sets. The value of this k factor is set to $\frac{2}{3} + \epsilon$ in the current version of SuMo, which indicates that strictly more than two third of the chemical groups of the smallest set must also belong to the other site.

Overlap between n sites n sites are said to be overlapping if and only if all the sites are pairwise overlapping.

The search for all the sets of overlapping sites within a given 3D structure can be solved by searching all the cliques in the graph such that:

- the vertices represent the sites,
- the edges represent the pairwise overlaps between sites.

Only maximal cliques are used to form what we will call characteristic sites. A definition of this classic NP-complete problem is given page 92.

Characteristic sites A characteristic site has been defined as a group of n sites. Each chemical group i is associated with the number of sites m_i to which it belongs and that were used to generate the characteristic site. If the m_i/n ratio is close to 1 then the chemical group can be considered as mandatory for a given biological function, and optional if it is low.

3.3.2.4 Matching characteristic sites

Characteristic sites from different 3D structures are considered as matched when they have at least one pair of matched sites, as illustrated on subfigures 3.10C and 3.10D page 68.

3.3.2.5 Graph of characteristic sites

A graph is built in which the vertices represent characteristics sites and the edges represent the matched characteristic sites. Such a graph is shown on subfigure 3.10E.

3.3.2.6 Families of characteristic sites

Families of characteristic sites are extracted from the previously defined graph. These families may be overlapping. The search for families is based on the search of almost complete subgraphs that will be called f-cliques and that are described in details section 3.6.4 page 92. The f function which is used in SuMo version 4.4 is the following:

$$f : n \rightarrow \left\lfloor 0.24 \cdot \left(n - \frac{1}{2}\right)\right\rfloor$$

The following table shows the first values that are taken by f:

n	$f(n)$
$1, 2, 3, 4$	0
$5, 6, 7, 8$	1
$9, 10, 11, 12$	2

This approach allows the generation of groups of characteristic sites that have similarities which are detected by SuMo almost everywhere. The result is illustrated by subfigure 3.10F.

3.4 Databases

In early 2003, the PDB has approximately 20,000 entries and most of them are structures containing mostly proteins. The size of such a database in the non-compressed official PDB format requires a space of 11 gigabytes (GB). Although the hardware for storing such data is today quite cheap, other problems arise.

When the whole contents of a remote database is scanned by a local process, all the database will transfered through the network. Two kinds of efforts may help to solve such problems:

1. buy hardware so that the database is mirrored on the client host or so that the network between the server and the client is very fast,
2. reduce the size of the database by compressing its information.

A big difficulty is how to build a database from the PDB for being scanned efficiently by SuMo. Our purpose is to remove redundancies that are at the same time structural and functional repeats and that are obvious to the user, while keeping everything else. It is important to remove or hide any repetitive result when it is possible so that interesting results are better highlighted. Several selection heuristics have been created for this purpose.

Two databases are generated and used by SuMo. The first one is the database of potential target structures. The other one is the database of known functional sites, which currently only contains the ligand binding sites.

3.4.1 Potential target structures

The database of potential target structures is a representation of all the proteins from the PDB in the SuMo format after some redundancies have been removed.

3.4.1.1 Removal of redundancies

We assume that the high number of structures which are available for proteins of very similar folds or sequences is not a redundancy, and very often it is exactly the opposite.

Thus every structure from the PDB constitutes an entry in the database of potential target structures. Nevertheless, the structure of many proteins is given as a multimer which may or may not have a biological relevance. Only at this level some information will be considered as redundant and removed. This should be done carefully; it is described in the following paragraphs.

Preserving the environment If, for instance, a homodimeric structure of a protein has been identified and that we want to keep only one of the monomers in the final representation, then the properties of the environment around each chemical group must be the same as if nothing had been removed. In particular this concerns the following parameters:

- local atomic density,
- orientation against the macromolecule,
- position of the surrounding atoms,

- free or bonded hydrogen donors or acceptors.

These parameters are preserved when using the `-select` option when the graph of triplets is built from the initial structural data. A somewhat opposite behavior is triggered when using the `-restrict` option. In this case, it is more or less the same as working on a truncated PDB file. The `-restrict` option will not be used here and in general should be used with care. Other options of the `read` function of the SuMo language are presented section 3.7.1 page 96.

Preserving boundary regions Functional sites may be the result of the multimerization of proteins and extend over the boundary between 2 identical subunits. In order to fix this problem, not only one monomer is selected but also a shell around the monomer. This shell has a thickness of 6 Å in the version 4.4 of SuMo. This radius is used for the selection using the `around` keyword which precise meaning is given with the description of the selection language, section 3.7.1.3 page 101. For example, the selection of the A monomer of an AB dimer will be performed with `Pdb_chain "A" or 6.0 around (Pdb_chain "A")`.

3.4.1.2 Identification of redundant chains

Currently the only criterion which is used for the removal of redundant chains is the sequence of the polymers which is given in the PDB file. One of the problems that occur is that some monomers—usually amino acids—are not resolved in one or several monomers and break the apparent sequence of the polymer. This problem can be worked around by considering that two sequences which are very similar can be considered as a redundancy instead of requiring a perfect identity. Another problem is that some PDB files use the same chain identifier for different molecules such as a subunit and its ligands or even more exotic naming schemes.

Two chains will be considered as equivalent according to their sequence which is determined after the numbering of the monomers in the PDB file. Monomers that do not belong to the 20 classic amino acids are ignored. Sequences shorter than 10 monomers are ignored too. Two sequences s_1 and s_2 must share at least 90% of their sequence. For better results as well as for efficiency purposes, it means that at least 90% of the subsequences of length 4 from s_1 must exist in s_2, and reciprocally.

A standalone program has been written for the sole purpose of identifying the significant chains in a PDB file and the equivalence between these chains according to the previously described criteria. The program's name is `seqinfo`. It is used directly for generating the chains that will be selected for the database.

3.4.1.3 Final size of the database

The size of the compressed database of target structures is about 2 times the size of the non-compressed PDB.

3.4.2 Ligand binding sites

A database of ligand binding sites is generated and used by SuMo.

3.4.2.1 Selection

The selection of ligand binding sites is performed by identifying the chemical groups that have at least one target location which is located at a distance of less than 4 Å of one the non-hydrogen atoms of the ligand. This is implemented and accessible with `around` construct in the language for selecting chemical groups (described section 3.7.1.3 page 101). The definition of ligands is given section 3.2.1.1 page 47.

3.4.2.2 Removal of the redundancies

The removal of the redundancies for this database is based on the same principle as the one which is used for the database of target structures. For instance, in the case of an AB homodimer of proteins, the selection of the binding site of molecule 3 will use the following code:

```
((4.0 around Molecule_index 3) and (not Molecule_index 3))
and
(Pdb_chain "A" or 6.0 around (Pdb_chain "A"))
```

3.4.2.3 Removal of tiny sites

The sites which are too small to be returned as results of a comparison by SuMo are not put into the database.

3.4.2.4 Sequence of types of triplets

The ordered list of all types of triplets of a given ligand binding site is recorded into a separate file.

This is an optimization that speeds up the comparison against other small sites. If two sites that are being compared do not share a single triplet of the same type, then it is not necessary to load the full representation of the sites since the result will be empty. This optimization is used in the systematic comparison of all sites against themselves which is described section 3.5.4 page 76.

3.4.2.5 Size of the database

The database of ligand binding sites contains 11,000 sites and concerns 1,800 ligands. These sites were extracted with version 4.4 of SuMo from the PDB with about 20,000 entries.

This database uses 400 megabytes (MB) of disk space.

3.5 Predictive annotation

A system of automated prediction of ligand binding sites has been developed. It is based on a post-processing of the results obtained after scanning the database of ligand binding sites. Although the raw results are already a prediction, the system which is introduced here automatically performs a summary under the form of a list of annotated chemical groups from the structure of interest.

This system is based on two essential concepts: family of ligands and specificity of detection.

3.5.1 Families of ligands

Let B be the set of known ligands, for instance those found in the PDB. A *family of ligands* is a bipartition (I, E) of B where I defines the ligands that actually belong to the family and E are the ligands that explicitely do not belong to the family.

> *When the PDB is updated and introduces new definitions of ligands, the defined families must be updated.*

Let S denote the function that returns all the ligand binding sites for a given set of ligands in a given database. A *family of ligand binding sites* is defined by the image $(S(I), S(E))$ of a family of ligands (I, E).

The families of ligands are generated according to the will of the local administrator of SuMo and are stored manually in a the file system which is associated with the database. Every ligand identifier l automatically defines a singleton family $(\{l\}, B - \{l\})$.

3.5.2 Apparent specificity

Let f denote a function that returns lists of matched chemical groups between some chemical groups from an arbitrary structure and a set S of n ligand binding sites:

$$f(X, S) = \{C_1, C_2, \ldots, C_i, \ldots, C_n\}$$

where X is the query, i.e. the set of chemical groups that are used for scanning the S database of sites and C_i is the result of the comparison with site number i from S. C_i is a set of lists of matched chemical groups:

$$C_i = \{L_1, L_2, \ldots, L_{k_i}\}$$

Each C_i comparison result is then merged into one single site that gathers all matched chemical groups that are found in L_1 to L_{k_i}. The resulting subset of X is a site that will be denoted by M_i.

We can now define the function m_S that for a given chemical group g from X returns the number of times when this chemical group is present in sites from M_1 to M_n:

$$m_S(g) = \sum_{i=1}^{n} \begin{cases} 1 & \text{if } g \in M_i \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

The *selectivity* of chemical group g for a set of sites S is defined by the following ϕ_S function:

$$\begin{aligned} \phi_S(g) &= \frac{m_S(g)}{|S|} \\ &= \frac{m_S(g)}{n} \end{aligned} \tag{3.2}$$

The *apparent specificity* of a chemical group g which belongs to a given arbitrary set X for a family of sites given by $(S(I), S(E))$ is defined by following ψ function:

$$\psi(g, I, E) = \frac{\phi_{S(I)}(g)}{\phi_{S(E)}(g)}$$

In practice, function f is the comparison heuristics of SuMo. It must be noticed that the specificity at g depends on X since X often represents a 3D substructure and not a full structure. For example, if X is a selection of chemical groups and some of them belong to a given ligand binding site but this ligand binding site is not entirely included in X, then it will be difficult to predict a relevant specificity of the chemical groups for the given ligand.

In order to assess the reliability of the apparent specificity, il it important to associate it with the number of sites and the number of chemical groups that are used in the computation. Thus we consider that $m_{S(I)}(g)$ must be equal or greater than 10 if we want the apparent specificity to be usable for chemical group g and the given (I, E) family.

3.5.3 Application: prediction and annotation

Scanning the whole database of ligand binding sites with a query X may result in some large data which is not so easy to analyze. Instead, for each chemical group of X, the apparent specificity for each of the families of ligands can be computed and displayed when it is considered as reliable as defined previously. This guarantees that SuMo detects a similarity with a site inside of a matched family in at least 10 cases.

An apparent specificity which is close to 1 would indicate that SuMo is very bad at telling if a site belongs to a given family or not.

A higher specificity, e.g. 10, shows that the given chemical group is 10 times more often detected as a member of the family than an outsider.

Clever definitions of families of ligands can be designed according to the chemical structure of ligands and with some minimal knowledge about the comparison heuristics. Such definitions can lead to higher specificities than by considering only the trivial predefined singleton families.

3.5.4 Application: self validation

In order to evaluate the quality of SuMo for predicting functional sites, it is possible to test it on known functional sites.

Thus, each ligand binding site of the SuMo database can be used as a query and compared to every known site. For each family that contains this ligand, the apparent specificity of each of its chemical group will be computed, when possible.

3.5.4.1 Specificity at the level of the functional site

We will now define the apparent specificity of a functional site X instead of a specificity for each of its chemical groups. Let us define the following Φ_S function, analog to the selectivity function ϕ that has been defined by the expression 3.2 page 75:

$$\Phi_S(X) = \frac{\sum_{g \in X} m_S(g)}{\sum_{g \in X} |S|}$$

where S is the database of sites that is scanned and m_S is the function which is defined by expression 3.1 page 75. The apparent specificity of site X for the family of ligands (I, E) is given by the following Ψ function:

$$\Psi(X, I, E) = \frac{\Phi_I(X)}{\Phi_E(X)}$$

3.5.4.2 Computation

Let us consider a database of ligand binding sites that contains n sites. The computation of $\Psi(X, I, E)$ for every site X of the database and all families (I, E), $n.(n-1)/2$ pairwise comparisons of sites are required. If the results of each comparison cannot be stored until the end of the process—which happens in practice—then $n.(n-1)$ comparisons are required, every comparison being performed twice.

In order to speed up the comparisons, an optimization has been implemented: the second site is fully loaded only if it has at least one triplet of the same type of those found in the first site (the query). This requires only to load a much smaller and simpler file. In practice, 3 out of 4 comparisons are skipped and the computation time is reduced by a factor 4.

For SuMo version 4.4 and the 11,000 sites that were considered in March 2003, the time which is required on a machine with 2 Intel Pentium 800MHz processors is 10 days. If the size of the sites—which is directly related to the number of triplets—is multiplied by a factor k close to 1, the required time is approximately multiplied by a factor k^2.

3.6 Details on heuristics

When designing a heuristics of good quality, it is essential to not split in time or space the different steps of the design, which are:

1. formulating the biological problem,
2. modeling the problem,
3. defining an algorithm.

The quality of each of these steps is estimated by (1) the relevance of the analysis or the prediction being carried out, (2) the quality of the approximation induced by the modeling and (3) how long it takes to compute a solution for an instance of the problem. Each of these criteria is essential: a solution must bring useful hints to the biologists and this solution must be obtained within a reasonable amount of time. The nature of the model that has been chosen will determine to which extent the chosen approach is reusable. Working on a very simplified model will in some cases result in very fast computations without even writing a dedicated program, but most of the time it will not return any valuable information concerning the original problem. An opposite approach is to work on a complex model that will make it difficult to reuse the program for other kinds of problems and will require a heavy investment for writing algorithms and programs.

Very strong communication is thus required in the team that develops the different steps. In practice, it is much more efficient if only one person is in charge of the three steps rather than one biologist, one mathematician and one computer scientist.

The current section presents several heuristics that were initially developed for the comparison of structures of biological macromolecules. However these problems and their solutions are independent from the notion of molecule. The problems that deal with points in 3 dimensions may be most of the time extended to any euclidean space.

3.6.1 Atomic density

Density is defined as an amount of some material in a given volume. In continuous medium, a local notion of density can be defined as the limit of the ratio between the amount of material and the volume around the point of interest when this volume tends to 0, i.e. $\frac{dm}{dx.dy.dz}$. But in a discrete medium such as a macromolecule which is modeled by point atoms, this does not work. In this case, an interpolation will be performed so that a continuous density function could be defined.

A *local density* is defined by a centered weight function ϕ that tends to 0 when the distance to the origin increases:

$$\phi : \mathbf{R}^3 \rightarrow \mathbf{R}^+$$

For any point P_i of coordinates (x_i, y_i, z_i), the weight function is expressed as:

$$w_i : (x, y, z) \mapsto \phi(x - x_i, y - y_i, z - z_i)$$

ϕ must be spheric, i.e.

$$x^2 + y^2 + z^2 = x'^2 + y'^2 + z'^2 \Rightarrow \phi(x, y, z) = \phi(x', y', z')$$

And a reasonable definition of ϕ also requires that ϕ decreases when the distance to the origin increases:

$$x^2 + y^2 + z^2 < x'^2 + y'^2 + z'^2 \Rightarrow \phi(x, y, z) \geq \phi(x', y', z')$$

Let **P** denote a discrete set of points that are associated with a mass given by function m. The density which is associated with P_i is formulated as:

$$\text{densité}(P_i) = \frac{\frac{\sum_{P_j \in \mathbf{P}} w_i(P_j).m(P_j)}{\sum_{P_j \in \mathbf{P}} w_i(P_j)}}{\int\int\int_{\mathbf{R}^3} w_i(x,y,z).dx.dy.dz} \tag{3.3}$$

The units of distance and mass should keep a physical meaning and the density should not depend on ϕ in a hypothetical homogenous medium. For these reasons, ϕ must be normalized, i.e.

$$\int\int\int_{\mathbf{R}^3} \phi(x,y,z).dx.dy.dz = 1$$

In order to make the density function continuous, ϕ must be continuous.

The ϕ function that was chosen—mainly for efficiency reasons—is the following:

$$\phi(x,y,z) = \begin{cases} 4\left(1 - \frac{r}{r_{\max}}\right) & \text{if } r \leq r_{\max} \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

where r is the norm of (x,y,z), i.e. $\sqrt{x^2+y^2+z^2}$.

The main advantage of this function is that the points that are located in radii larger than $r_{\max}$ can be ignored in the computations. In particular in the case of macromolecules, the cost of the density computation at a given point does not depend on the size of the system. However, other kinds of functions could have been chosen. Figure 3.6.1 page 80 shows the density as computed for different values of $r_{\max}$ in a system which is composed of 6 points forming a regular hexagon.

3.6.2 Comparison of local shape

3.6.2.1 Formulation of the problem

Given two superposed objects, can we say that their shape is similar? This problem arises when several pairs of chemical groups have been matched and the 2 molecules have been rigidly superposed according to the matched groups. If we wish to compare the environment around the chemical groups of interest, we need to set up a heuristics of shape comparison that is not restricted to the region of interest. The data that define the shape of molecules are the positions of the atoms, and possibly some information regarding their radius. A molecule which is modeled by spherical atoms can be viewed as the union of a finite set of spheres. The difference in the shape of the superposed submolecules M_1 and M_2 can be estimated through the volume of

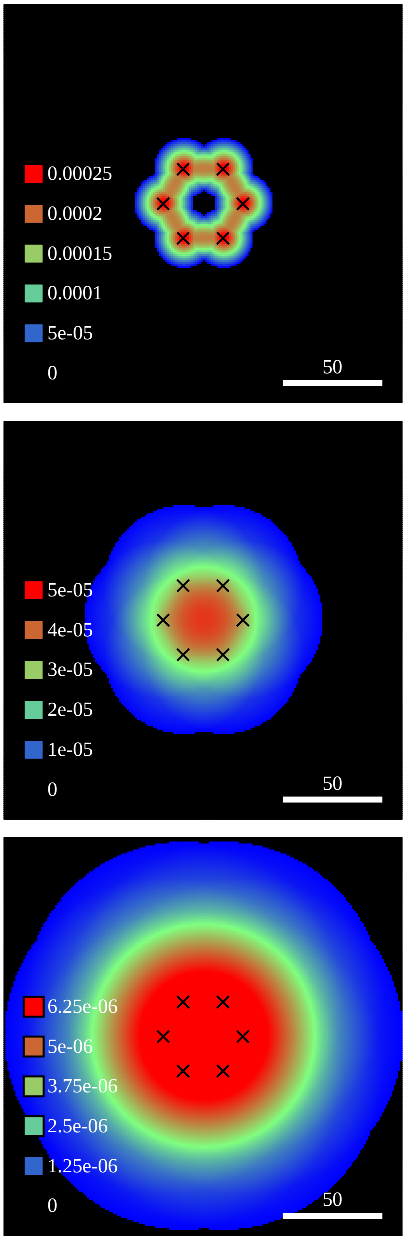


Figure 3.11: Density of a set of points as calculated by function 3.3 page 79 using the weight function 3.4 page 79 with different values of $r_{\max}$, respectively 15, 40 and 80. The points are represented by crosses and are all located in the plane of the sheet. Black regions are those where the density is zero and are continuous with the blue regions.

the non-intersection of M_1 and M_2, i.e. $(M_1 \cup M_2) - (M_1 \cap M_2)$. The calculation of the volumes of M_1, M_2 and $\text{excl}(M_1, M_2)$ is not trivial and forces us to discriminate the atoms that are inside the environment from the other atoms. Such a discrimination would make the function of shape comparison discontinuous. This problem is illustrated in 2 dimensions by figure 3.6.2.1 page 82.

The criteria for a good heuristics of local shape comparison are the following:

- continuity regarding any point of the system;
- computation in $O(n)$ where n is the size of the environment that is considered for the shape comparison.

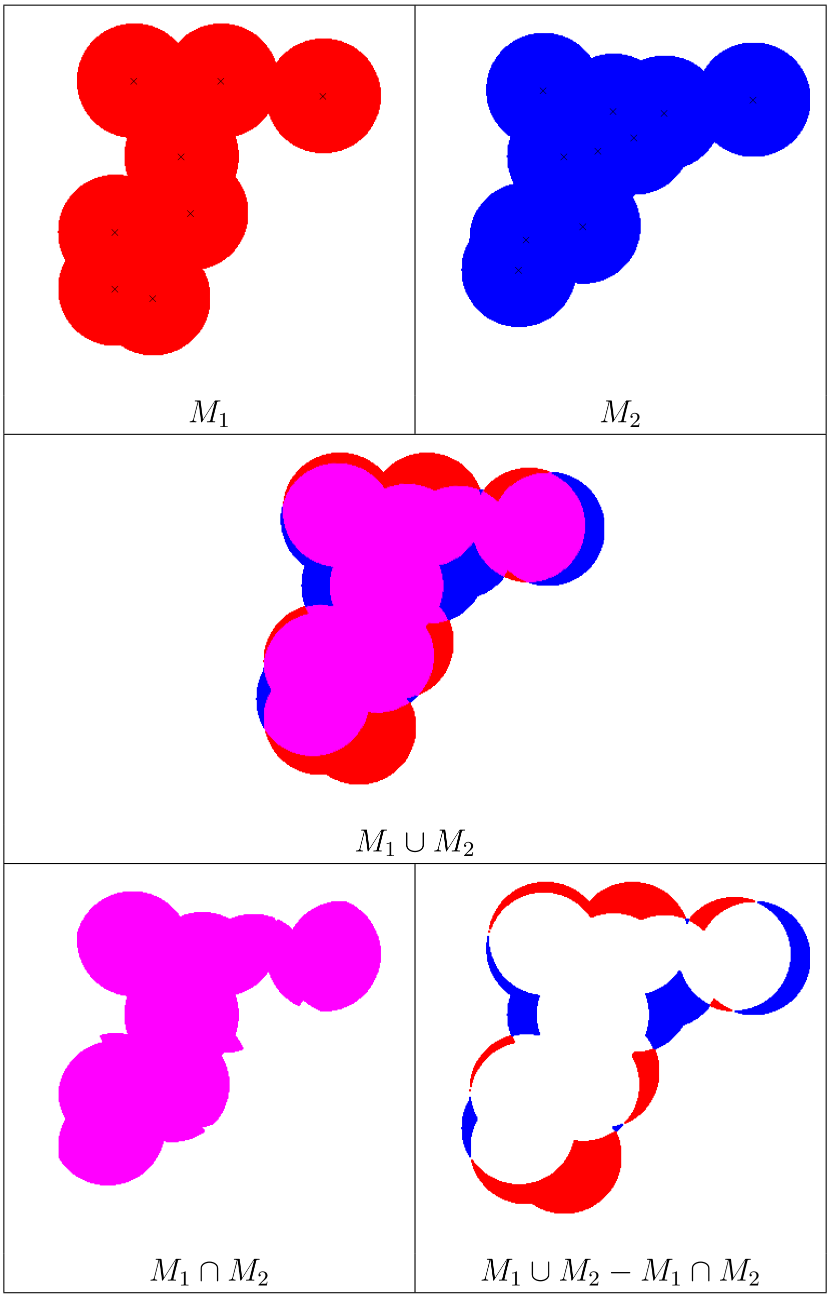
Any element that has an attribute that defines its position will be called a point. For example, it is possible that 2 points have the same coordinates but at the same time are different because of their other properties. In the following, we will call a volume any function that verifies the following conditions for any sets of points A and B:

1. $\text{volume}(A) \geq 0$
2. $\text{volume}(A \cup B) \leq \text{volume}(A) + \text{volume}(B)$
3. for any transformation f that keeps the distances between elements of A:
   $\text{volume}(A) = \text{volume}(f(A))$
4. for any points (p, q):
   $\lim_{\|p-q\| \to 0} \text{volume}(A \cup \{p\}) = \text{volume}(A \cup \{q\})$
5. for any points (p, q):
   $\lim_{\|p-q\| \to +\infty} \text{volume}(\{p, q\}) = \text{volume}(\{p\}) + \text{volume}(\{q\})$

3.6.2.2 General scoring function

Let M_1 and M_2 denote 2 sets of points that were superposed. We would like to know if their shape is similar. The comparison function is denoted by $\text{comparison}_{\text{shape}}$ and was defined as follows:

$$\text{comparison}_{\text{shape}}(M_1, M_2) = \frac{\text{volume}(M_1) + \text{volume}(M_2) - \text{volume}(M_1 \cup M_2)}{\text{volume}(M_1 \cup M_2)} \tag{3.5}$$



M_1

M_2

$M_1 \cup M_2$

$M_1 \cap M_2$

$M_1 \cup M_2 - M_1 \cap M_2$

Figure 3.12: Union and intersection of sets of spheres M_1 and M_2.

Definition 3.5 page 81 is inspired by the expression of the ratio between the intersection and the union of solids, here referred as E_1 and E_2:

$$\frac{\text{intersection}}{\text{union}} = \frac{|E_1 \cap E_2|}{|E_1 \cup E_2|} = \frac{|E_1| + |E_2| - |E_1 \cup E_2|}{|E_1 \cup E_2|} \tag{3.6}$$

The volume of the intersection of solids quantifies the overlap and is maximal when the 2 solids are perfectly superposed. Nevertheless, when working with discrete sets of points that model abstract entities, the intersection of these points does not have the meaning we want. However, the volume of the union of these points has the expected properties, i.e. a minimal value when the sets are perfectly superposed and a maximal value when the distances between the sets are infinite. Thus when M_1 and M_2 are perfectly superposed, the score has the maximal value of 1.

3.6.2.3 Volume function

Each point p_i is associated with a position and a weight w_i. The volume function that was chosen is the following:

$$\text{volume}(\{p_1, \ldots, p_n\}) = \sum_{i=1}^{n} m_i \tag{3.7}$$

where the m_i verify the following system of equations:

$$\forall i \in [\![1, n]\!] \quad m_i = w_i \cdot \frac{m_i}{\sum_{j=1}^{n} f(d_{i,j}).m_j} \tag{3.8}$$

where $d_{i,j}$ is the distance between points p_i and p_j, and f is a continuous, decreasing function such that $f(0) = 1$ and $\lim_{d \to +\infty} f(d) = 0$. Such a function will be called *function of influence* since it quantifies the influence of one point onto another point and this influence decreases when the distance between the points increases. The following function of influence was chosen:

$$f(d) = 2^{-\left(\frac{d}{\delta}\right)^s} \tag{3.9}$$

2 parameters are involved in this definition, δ and s. In the context of the comparison of the shape of macromolecules, the values that were chosen are the following:

- $\delta = 2\text{Å}$
- $s = 2$

System 3.8 page 83 may be rewritten as:

$$\forall i \in [\![1, n]\!] \quad w_i = \sum_{j=1}^{n} f(d_{i,j}).m_j \tag{3.10}$$

Under this form, it appears that the problem consists in solving a linear system of n equations of n variables denoted by m_j. However we will see in the next paragraph that this is not exactly the approach that has been retained for computing the volumes.

3.6.2.4 Computation of the volume

We just saw that the volume of n weighted points can be computed as the solution of a linear system of n equations and n variables. But the cost of the resolution of such a system is $O(n^{2+c})$ where c is a positive constant which is lower than 1 that depends on the algorithm that is used. The initial form of the problem (equation 3.8 page 83) was actually designed so that an approximate but fast computation of the solutions is possible, by repeated iterations.

The heuristics is basically the following: point p_i is under the influence of all the other points of the system, the closer points being the most influent. This influence is characterized by the decreasing of the mass m_i. Weight w_i is the *characteristic mass* of point p_i, i.e. the mass of this point when it is isolated. If 2 points p_i and p_j both of weight 1 have the same coordinates and are far from all other points of the system, it is just as if there was only one of them. Therefore p_i and p_j will both have a mass of 0.5. If all masses m_i are initialized with their maximal value w_i as a first guess, we will apply the following procedure several times in order to make the values m_i converge to their correct value. This procedure is directly derived from expression 3.8 page 83:

$$m_i^{(t+1)} \leftarrow w_i \cdot \frac{m_i^{(t)}}{\sum_{j=1}^{n} f(d_{i,j}).m_j^{(t)}} \tag{3.11}$$

When using the function of influence 3.9 page 83, many coefficients $f(d_{i,j})$ are very low compared to 1, which is the value of each coefficient in the diagonal, i.e. $f(i, i)$. For this computation, we can choose to ignore the points that are located at a distance of more than 3δ, since in this case their mutual influence is:

$$\begin{aligned} f(3\delta) &= f(6\mathring{A}) \\ &\simeq 2.10^{-3} \end{aligned}$$

For the iterative computation of w_i, only the points which are within a radius of 6Å around p_i will be taken into account.

When the density of points is limited, which is the case with atoms or groups of atoms, the number of neighbor points that are taken into account for computing one w_i is limited. The size of formula 3.11 page 84 as used in the approximate computation is therefore not $O(n)$ anymore but $O(1)$.

The number of cycles that musts be performed depends on the required precision. This precision must be in agreement with the approximation that was made when ignoring the small coefficients. Moreover, it would not be reasonable to expect a better precision than the experimental uncertainties on the position of the atoms. In practice the first 3 digits are always correct when 10 cycles have been applied. The approximate computation is therefore perfectly suitable given the expectable accuracy. The time for computing the volume of n points is $O(n)$.

3.6.3 Estimation of deformation

3.6.3.1 Nature of the problems

Let us consider two sets of objects, $A = \{a_1, a_2, \ldots\}$ and $B = \{b_1, b_2, \ldots\}$. These objects are identified by their location in space. We identified a list L of n pairs of objects from $A \times B$ that are considered as equivalent. This is a binary relation between A and B that we will call *list of matched elements*. L is of the following form:

$$L = \{(a_{i_1}, b_{j_1}), (a_{i_2}, b_{j_2}), \ldots, (a_{i_n}, b_{j_n})\}$$

where each element a_{i_k} or b_{j_l} can possibly exist in different pairs from L.

If every pair (a, b) in L represents an equivalence for the realization of a local property, then a given element a will possibly be involved in different local properties.

Problem 1 *What is the quality of the matches given by L?*

The problem 1 page 85 is meaningless if the notion of quality of matches is not well-defined. For instance, a criterion that is frequently used in the field of structural biology is the RMSD (see definition 1 page 28). In the case of the identification of functional 3D sites in macromolecules, and in particular sites that bind flexible ligands, it has been considered in section 3.2.5.4 page 66 that it is preferable to adopt a criterion that deals with local deformations rather than global ones as it is the case with the RMSD.

3.6.3.2 Deciding what is local

The solution that was adopted is based on the deformation of the local environment of each pair (a_{i_k}, b_{j_l}). The definition of what is local is related to a notion of critical distance, below which the structures are considered as rigid and above which they are considered as virtually flexible. The term "virtually flexible" is used here with the meaning of functional flexibility as in definition 3.2.5.4 page 66.

In the case of 3D structures of molecules, the critical distance will be between the size of the largest rigid chemical groups and the smallest variable distances.

We consider as essential the double aromatic rings such as those found in tryptophan or puric bases. On the other hand, the flexibility of common ligands can be observed on molecular assemblies of the form A–B–C–D, where a free rotation is possible around the B–C axis which results in variations in the distance between atoms A and D. In the case of a carbon backbone $C_{(A)}$–CH_2–CH_2–$C_{(D)}$, the distance between A and D commonly fluctuates between 3 Å and 3.9 Å. The size of a double aromatic ring is 5-6 Å. It is thus reasonable to assign a value between 3 and 6 Å to the critical distance.

3.6.3.3 Solution

When we want to define a quantitative notion of local deformation in dimension 3, we have the possibility to define it from the deformation of tetrahedrons of small size by considering the variation of solid angles and distances that define these tetrahedrons. This has the property of differentiating the stereoisomers, as opposed to the study of only triangles and segments.

In spite of this, the deformation of segments has been chosen as a base for the estimation of local deformation because:

- in practice, in SuMo, the stereoisomers are removed before the estimation of deformation,
- the implementation is much easier and the computation much lighter.

In SuMo, stereoisomers are avoided thanks to the oriented angle between two adjacent triangles. For more details, see section 3.2.5.2 page 65.

Case of point objects Here we define the deformation for point objects only. This definition will be extended in the next sections in order to handle solids that may have some symmetries, as it is the case for the chemical groups in SuMo (section 3.2.1 page 44).

The principle of the estimation of deformation is the following: for each pair (a_i, b_j) of matched objects, the variation of distances between this pair of points and all other pairs are taken into account. This variation is called deviation and its definition is the following:

Definition 2 (Deviation between 2 pairs of points) *Let* $P = (a_i, b_j)$ *and* $Q = (a_k, b_l)$ *denote two pairs of points belonging to a list* L *of matched points, the deviation between* P *and* Q *is the difference between the euclidean distances* $a_i a_k$ *and* $b_j b_k$*:*

$$\begin{aligned} \text{déviation}(P, Q) &= \Big| \, \|a_i - a_k\| - \|b_j - b_k\| \, \Big| \\ &= |a_i a_k - b_j b_k| \end{aligned}$$

We also define the distance between 2 pairs of points:

Definition 3 (Distance between 2 pairs de points) *Given two pairs of points* $P = (a_i, b_j)$ *and* $Q = (a_k, b_l)$ *belonging to a list* L *of matched points. The distance between* P *and* Q *is defined as the arithmetic mean of the euclidean distances* $a_i a_k$ *and* $b_j b_k$*:*

$$\begin{aligned} \text{distance}(P, Q) &= \frac{1}{2} \left(\|a_i - a_k\| + \|b_j - b_k\| \right) \\ &= \frac{1}{2} \left(a_i a_k + b_j b_k \right) \end{aligned}$$

The estimation of the deformation of a list L of matched elements depends on two essential factors:

1. The *coefficient of importance* is a weight coefficient. It gives the importance of the variation being measured.

2. The *distance of reference* allows the expression of the deformation in a relative manner and depending on the distance being considered.

These two factors are described in details later. They depend only on the *distance between 2 pairs of points.*

Coefficient of importance We will consider that the deformation of a distance has a higher functional impact when the distance in question gets smaller. From a given d_{max} distance, the importance of the conservation of distances if considered as null. The following function is used:

$$\text{importance}(P, Q) = w_0 - \frac{w_0}{d_{\text{max}}} \cdot \text{distance}(P, Q)$$

where w_0 is a constant. In the implementation, d_{max} plays the role of a *cutoff*. We can give it a distance of 10 Å for example.

Relative deformation We will consider that the deformation must be expressed relatively to a characteristic distance of reference, that we will also call distance of reference. Here, the critical distance d_c as introduced section 3.6.3.2 page 86 is important. For the small distances, the distance of reference will indeed be considered as constant, while for larger distances the distance of reference will be the distance itself. The following function was chosen:

$$d_{\text{ref}}(P, Q) = \begin{cases} \text{distance}(P, Q) & \text{if distance}(P, Q) > d_c \\ d_c & \text{otherwise} \end{cases}$$

The deformation of distance $d(P, Q)$ is defined as the ratio δ between the measured deviation and the distance of reference:

$$\delta(P, Q) = \frac{\text{deviation}(P, Q)}{d_{\text{ref}}(\text{distance}(P, Q))} \tag{3.12}$$

δ can be considered as a deviation which is expressed relatively to a characteristic distance of reference.

Expressing function d_{ref} with two phases allows constant penalties for deviations of small distances and relative penalties for deviations of large distances. A reasonable value for d_c is 3 Å.

General formulation The deformation of a list of matched elements $L = \{(a_{i_1}, b_{j_1}), (a_{i_2}, b_{j_2}), \ldots, (a_{i_n}, b_{j_n})\}$ is defined by the following function:

$$\text{deformation}(L) = \frac{\sum_{P \in L} \sum_{Q \in L - \{P\}} \text{importance}(P, Q).\delta(P, Q)}{\sum_{P \in L} \sum_{Q \in L - \{P\}} \text{importance}(P, Q)}$$

It is just a weigthed mean of the deviations divided by the characteristic distances of reference.

Extension to multi-point objects without symmetry

Purpose of the extension Local deformation as defined previously gives a distance between two sets of points that are extracted from a list of matched elements. Here we will extend our definition of deformation so that it can take into account not only translations but also rotations of basic objects when these are not only defined by points but also contain some orientation data.

Definition of the objects being handled First, we consider objects without internal symmetry, that are modeled by what we call finite pseudo-solids, defined as follows:

Definition 4 (Pseudo-Solid) *We call pseudo-solid the model* (p, S) *of an object where* p *is its position and* S *is a set of arbitrary points.* p *is called the position of the pseudo-solid and the points from* S *are called the vertices of the pseudo-solid.*

Definition 5 (Finite Pseudo-Solid) *We call finite pseudo-solid any pseudo-solid* (p, S) *that has a finite number* $|S|$ *of vertices.*

Redefinition of deformation Let L denote a list of matched finite pseudo-solids. 2 matched pseudo-solids must have the same number of vertices. The computation of the coefficients of importance and the characteristic distance of reference use the distance between the positions, exactly like in the simple definition. But instead of considering the deviations of the distances between the positions of the pseudo-solid, we consider the deviation of the distances between every position of pseudo-solid and each vertex belonging to another pseudo-solid.

Adaptation for symmetric objects

Problem When the model for the objects have an internal symmetry, i.e. some orientations of these objects are equivalent from a functional point of view, it is important to take that into account.

For instance, the carboxylate group of an aspartate has 2 oxygen atoms which are a priori equivalent and denoted by `OG1` and `OG2`. We can model this object by the mean position of the oxygens and 2 additional vertices for each oxigen. Now we have to find a solution that allows us to consider that atoms `OG1` and `OG2` are functionally interchangeable.

Symmetric pseudo-solids

Definition 6 (Symmetric pseudo-solid) *A symmetric pseudo-solid is a pair* $(p, \{S_1, \ldots, S_k\})$ *where* p *is a point called its position and* S_1 *to* S_k *are distinct sets of points of same cardinality that are called symmetric variants. The elements of a symmetric variant are called vertices.*

The different symmetric variants of a given pseudo-solid model the transformations of this object that are functionally invariant. These transformations do not need to be isometric.

Extension of the definition of deformation Let L denote a list of matched symmetric pseudo-solids. In each pair of matched pseudo-solids, the number of symmetric variants may differ, but these variants must all have the same number of vertices.

Figure 3.13 page 91 illustrates the final definition of deformation that was chosen. In this example, 3 objects are matched and one of them (position (A_1, A_2)) is modeled by two symmetric variants that are composed of 2 points, while the two other objects have no symmetry and their only vertex is the position of the object.

The proposed definition is based on the choice of the best pair of symmetric variants between 2 matched objects. This brings us to define the notion of *local deformation.*

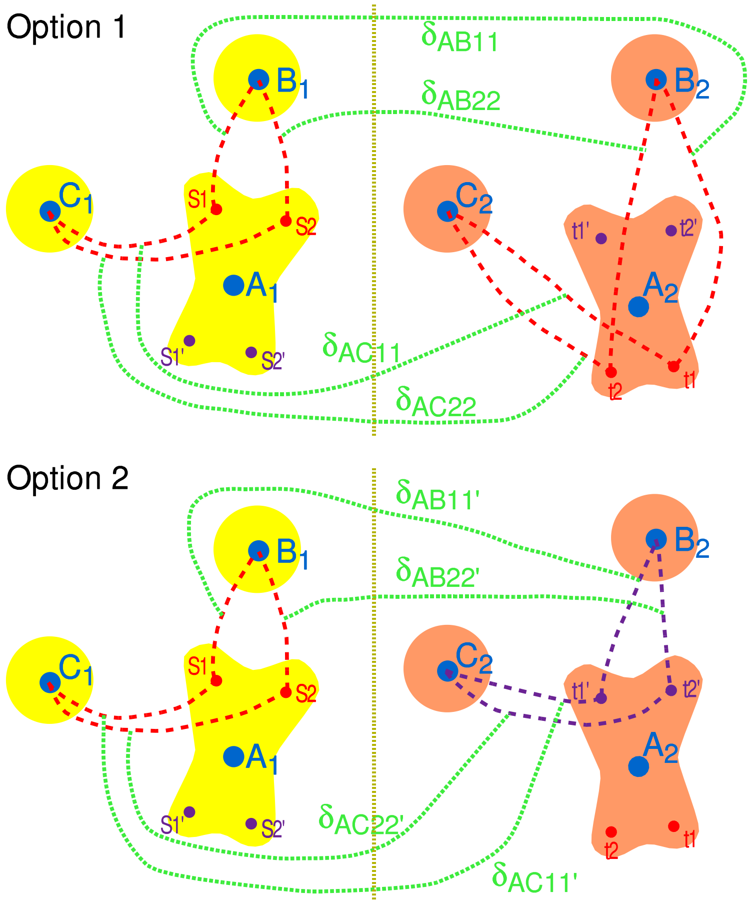
Definition 7 (Local deformation) *Let us consider a pair* $P_i = (X, Y)$ *of matched objects that are modeled by finite symmetric pseudo-solids. This pair is taken from a list of matched objects* $L = \{P_1, \ldots, P_i, \ldots P_n\}$*. We will call* A_1 *and* A_2 *their positions and denote by indexed letters* S *and* T *their symmetric variants:*

$$\begin{aligned} X &= (A_1, \{S_1, S_2, \ldots, S_{k_1}\}) \\ Y &= (A_2, \{T_1, T_2, \ldots, T_{k_2}\}) \end{aligned}$$

Let r *denote the number of points in the symmetric variants of* X *or* Y*, i.e.* $|S_1|$*. The local deformation at* (X, Y) *is the lowest sum of the deviations between the pair* (S_l, T_m) *of symmetric variants and the positions of all the other objects of the system:*

$$\text{local deformation}(L, (X, Y)) \cdot \text{local weight}(L, (X, Y)) = \\ \min_{(1 \le l \le k_1, 1 \le m \le k_2)} \left(\sum_{1 \le a \le r} \sum_{((p,V),(q,W)) \in L - \{P_i\}} w(\alpha).\delta(\beta) \right)$$

where w *is the function of importance that gives the coefficient of importance as defined page 87;* α *denotes* $((A_1, A_2), (p, q))$*;* β *denotes* $((S_l[a], T_m[a]), (p, q))$.


Option 1
δAB11
δAB22
B1
B2
C1
S1
S2
A1
C2
t1'
t2'
A2
S1'
S2'
δAC11
t2
t1
δAC22
Option 2
δAB11'
B1
B2
δAB22'
C1
S1
S2
C2
t1'
t2'
A1
A2
S1'
S2'
δAC22'
t2
t1
δAC11'

$$
\begin{aligned}
L = \{ \quad & \big((A_1, \{\{s_1, s_2\}, \{s_1', s_2'\}\}), (A_2, \{\{t_1, t_2\}, \{t_1', t_2'\}\}) \big), \\
& \big((B_1, \{\{B_1\}\}), (B_2, \{\{B_2\}\}) \big), \\
& \big((C_1, \{\{C_1\}\}), (C_2, \{\{C_2\}\}) \big) \quad \}
\end{aligned}
$$

$$
\delta_{AB11} = \frac{|s_1B_1 - t_1B_2|}{d_{\text{ref}}\left(\frac{1}{2}(A_1B_1 + A_2B_2)\right)}
$$

$$
w_A.\delta_A = \min \begin{cases}
w_{AB}.\delta_{AB11} + w_{AB}.\delta_{AB22} + w_{AC}.\delta_{AC11} + w_{AC}.\delta_{AC22} & \text{Option 1} \\
w_{AB}.\delta_{AB11'} + w_{AB}.\delta_{AB22'} + w_{AC}.\delta_{AC11'} + w_{AC}.\delta_{AC22'} & \text{Option 2} \\
w_{AB}.\delta_{AB1'1} + w_{AB}.\delta_{AB2'2} + w_{AC}.\delta_{AC1'1} + w_{AC}.\delta_{AC2'2} & \text{Option 3} \\
w_{AB}.\delta_{AB1'1'} + w_{AB}.\delta_{AB2'2'} + w_{AC}.\delta_{AC1'1'} + w_{AC}.\delta_{AC2'2'} & \text{Option 4}
\end{cases}
$$

$$
\text{deformation}(L) = \frac{w_A.\delta_A + w_B.\delta_B + w_C.\delta_C}{w_A + w_B + w_C}
$$

Figure 3.13: Estimation of deformation from a list L of matched finite symmetric pseudo-solids. 2 options among the 4 possible are shown. The chosen option for the local deformation δ_A is the one that minimizes the deformation. Here, option 2 will probably be preferred to option 1.

The local weight is the sum of the coefficients of importance that are used in the sum, i.e.:

$$\begin{aligned}\text{local weight}(L,(X,Y)) &= \sum_{1\leq a\leq r}\ \sum_{((p,V),(q,W))\in L-\{P_i\}} w((A_1,A_2),(p,q))\\ &= r.\left(\sum_{((p,V),(q,W))\in L-\{P_i\}} w((A_1,A_2),(p,q))\right)\end{aligned}$$

The global deformation is simply the weighted mean of the local deformations:

Definition 8 (Deformation) *For a list L of matched elements, the deformation is defined as follows:*

$$\text{deformation}(L) = \frac{\sum_{P\in L}\text{local weight}(L,P)\cdot\text{local deformation}(L,P)}{\sum_{P\in L}\text{local weight}(L,P)}$$

Variant The arithmetic mean of fraction terms such as the basic relative deformations given by function δ does not return a global deformation that is expressed by a sum of deviations divided by a sum of distances of reference. In this case, it may be preferable to keep separate the denominator terms and the numerator terms and finally divide the mean numerator by the mean denominator. Thus we will replace the arithmetic mean of d_i/n_i terms with weight w_i by:

$$\frac{\sum_i w_i d_i}{\sum_i w_i n_i} \qquad \text{(new mean)}$$

instead of:

$$\frac{\sum_i w_i \frac{d_i}{n_i}}{\sum_i w_i} \qquad \text{(arithmetic mean)}$$

Thus, the mean itself is expressed under a fractional form and the mean of the means becomes equivalent to the mean of the initial elements.

This is the form that is currently used for the mean of the local deformations, while keeping the basic δ deformations under fractional form.

3.6.4 Incomplete cliques

3.6.4.1 Definitions

Definition 9 *Let us consider a graph $G = (V, E)$ and a subset V' of the vertices V of graph G. V' is a clique of G iff each pair of vertices of V' is an edge, i.e. belongs to E.*

Definition 10 *Let us consider a graph $G = (V, E)$ and an increasing function f from $\mathbf{N}$ into $\mathbf{N}$. Let V' denote a subset of vertices from V. If each element of V' is at least connected to $|V'| - 1 - f(|V'|)$ other vertices of V', then we will say that V' is an f-clique of graph G.*

Definition 11 *We will call stable f-clique any f-clique whose elements can be ordered as $(v_1, v_2, \ldots, v_n)$ so that the following subsets are all f-cliques:*

$$\begin{array}{l} \{v_1\} \\ \{v_1, v_2\} \\ \{v_1, v_2, v_3\} \\ \vdots \\ \{v_1, v_2, v_3, \ldots, v_n\} \end{array}$$

Note that a clique is the special case of a (stable) f-clique when f is the null function.

Definition 12 *A maximal stable f-clique is a stable f-clique of a graph G that is not included in any other stable f-clique of G.*

Definition 13 *A maximum stable f-clique is a stable f-clique of maximum cardinality for a given graph.*

Both previous definitions are classic for cliques and are just a generalization for any value of f. Figure 3.14 page 94 shows different examples of graph of 5 vertices. For each of these graphs are given full lists of the maximal f-cliques, maximal stable f-cliques and maximal cliques with the following function f:

$$f : n \rightarrow \left\lfloor \frac{n-1}{2} \right\rfloor$$

Here we will solve the problem of finding all the maximal cliques of a given graph and its generalization to f-cliques stables. The problem of the maximum clique is known to be NP-complete for arbitrary graphs. Thus, returning a complete list of all the maximal cliques is at least as difficult. Moreover, we are interested in the more general case of f-cliques. Therefore no solution in polynomial time is known for this problem: the algorithms that are given here should be used on graphs that have a special structure or graphs of very small size.

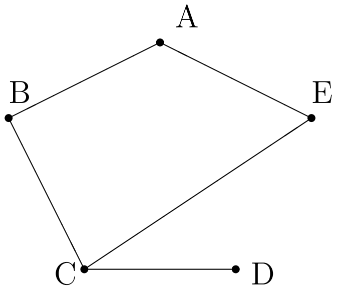
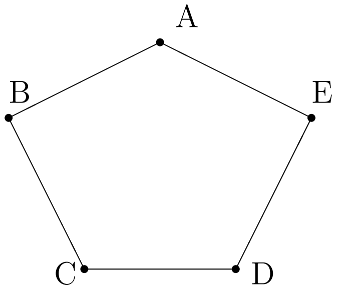
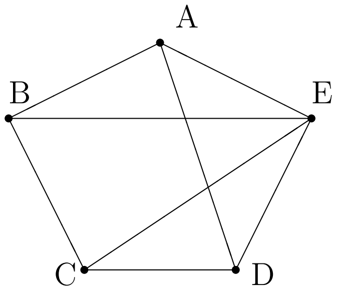
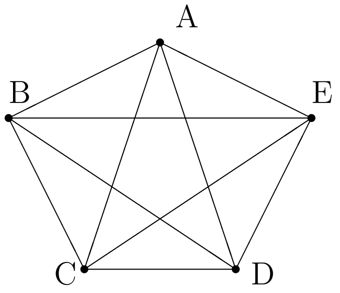
	maximal f-cliques	maximal stable f-cliques	maximal cliques
	ABCE, CD	ABC, ABE, ACE, BCD, BCE, CDE	AB, AE, BC, CD, CE
	ABCDE	ABC, BCD, CDE, DEA, EAB	AB, AE, BC, CD, DE
	ABCDE	ABCDE	ABE, ADE, BCE, CDE
	ABCDE	ABCDE	ABCDE

Figure 3.14: Maximal f-cliques, maximal stable f-cliques and maximal cliques in different examples of graphs. $f(n) = \left\lfloor \frac{n-1}{2} \right\rfloor$; according to f, each vertex of a given f-clique must be connected to at least 50 % of the other vertices of the f-clique.

3.6.4.2 Algorithms

The algorithm that has been designed for extracting the maximal stable f-cliques of a graph work by “seed extension”. The properties that follow must be matched or the algorithm will not be correct. They directly come from the definition of stable f-cliques.

Property 1 *Given a graph G, all stable f-cliques of size $n+1$ can be obtained from the set of stable f-cliques of size n, for $n \geq 1$.*

This property comes from the fact that any stable f-clique A of size $n+1$ can be written as $B \cup \{v\}$ where B is a stable f-clique of size n.

Property 2 *All stable f-cliques of size n ($n \geq 1$) can be obtained from the set of stable f-cliques of size 1, i.e. the individual vertices.*

Thus it is possible to define an algorithm that builds all the stable f-cliques of size n from the set of stable f-cliques of size $n-1$. Each stable f-clique of a graph $G = (V, E)$ is built in $O(|V|)$ from an f-clique of lower size. If m is the total number of stable f-cliques in graph G, the time for computing the maximal stable f-cliques with this algorithm is $O(m.\,|V|)$.

Algorithm 2 page 96 specifies the different steps of the extraction of the maximal stable f-cliques. The datatypes that are used for the manipulation of the different kinds of sets are not specified.

For example, a set of stable f-cliques can be implemented as a hash table in which the data are stable f-cliques and the keys are computed from the numerical identifiers of their vertices. For instance, the set of vertices 4, 23, and 6 will result in an array {4;6;23} from which the hash key will be computed. The standard library of Objective Caml provides a module dedicated to the manipulation of hash tables (Hashtbl). It uses a hashing function that works on any type of data. The tables grow as needed. More details on hash tables can found in book [46].

3.7 User interfaces

The levels of SuMo have been introduced section 3.1.1 page 34. The programmer’s level in not a user level. The middle level is not targeted to the average user but it is necessary to serve as a triple interface between the `sumo` program, the operating system and the users (human or not). The higher level is the regular user level and proposes an interface which is completely independent from the underlying operating system. It consists of a client-server interface which is based on the execution of programs (CGI) by an HTTP server and displays data mainly in the HTML format.

Algorithm 2 Extraction of the maximal f-cliques

```
Require: a graph $G = (V, E)$
$V = \{v_1, v_2, \ldots, v_n\}$
Cliques $\leftarrow \{\{v_1\}, \{v_2\}, \ldots, \{v_n\}\}$
Result $\leftarrow \varnothing$
while Cliques $\neq \varnothing$ do
    Extensions $\leftarrow \varnothing$
    for all Clique $\in$ Cliques
        for all Vertex $\in$ Clique
            for all Neighbor $\in$ Neighbors(Vertex)
                if Neighbor $\notin$ Clique then
                    if (Clique, Neighbor) $\notin$ Extensions then
                        Extensions $\leftarrow \{(\text{Clique}, \text{Neighbor})\} \cup$ Extensions
    New-cliques $\leftarrow \varnothing$
    for all (Clique, Vertex) $\in$ Extensions
        if Clique $\cup$ {Vertex} is still a $f$-clique then
            New-cliques $\leftarrow$ {Clique $\cup$ {Vertex}} $\cup$ New-cliques
            remove Clique from Cliques
    Result $\leftarrow$ Cliques $\cup$ Result
    Cliques $\leftarrow$ New-cliques
```

3.7.1 Native SuMo scripts

The SuMo language has to be used for writing commands for the `sumo` program. Its implementation allows to use it either interactively or from script files. In any case, the language is interpreted on-the-fly and most errors are detected only when the commands are executed. This important restriction is not so problematic since the interactive mode should only be used occasionally, for testing purposes. In practice, most of the commands are generated automatically by the higher level interfaces such as CGI programs of the web interface.

First we will present the SuMo language. Then the available functions are described in a specific section.

3.7.1.1 The SuMo language

Program A program is a mix of declarations and expressions, separated by semi-colons (`;`) when needed.

Expressions An expression may be a single value. In this case, this value is returned. If it is a sequence of values, the first one is supposed to be a function and is applied to the other values of the sequence.

Declarations After its evaluation, an expression `expr` can either be ignored:

```
expr;
```

or be given a name:

```
let ident = expr;
```

where `ident` is a sequence of characters beginning with a lowercase letter ('a' to 'z') and possibly followed by letters or digits ('a' to 'z', 'A' to 'Z', '0' to '9' and '_').

E.g.:

```
sumo> 0;
- : Int
sumo> let x = 0;
x : Int
```

Functions Only predefined functions may be used, but they are considered like any object of the language. In the following example, we first check the type of the value bound to `print`, then we bind it to the new identifier `output` and test it.

```
sumo> print;
- : Function
sumo> let output = print;
output : Function
sumo> output "hello";
hello
- : Unit
```

Built-in types All types are built-in. These are `Unit`, `Bool`, `Int`, `Float`, `String`, `Function`, `Message`, `PDB_file`, `DB`, `DB_entry`, `Molecular_graph`, `Comparison_result`, `Multi_comparison_result`, `'a Option`, `List`, `'a Disk_entry`. `'a` (alpha) may be any type as in Caml.

Constructors

Type constructors Type constructors are used to create new objects of a given type. All type constructors begin with an uppercase letter within 'A'-'Z'.

Type constructors are:

```
PDB_file string
DB string
Message string
DB_entry (string_db, string_id)
DB_entry (db, string_id)
```

where `string`, `string_db`, `string_id` are expressions of type `String` and `db` is an expression of type `DB`.

Options Options are arbitrary polymorphic constructors awaiting one argument. They begin with the minus ('-') character and are followed by lowercase characters.

E.g.:

```
sumo> print "Hello World!" -file "hello.txt";
- : Unit
```

produces the same effect as:

```
sumo> let fileoption = -file "hello.txt";
fileoption : String Option
sumo> print "Hello World!" fileoption;
- : Unit
```

Lists Lists are heterogeneous sets of elements. `[]` is the empty list, `["foo"; 12; 2.3]` is a list containing 3 elements of type `String`, `Int` and `Float`.

Comments Comments begin with `(*` and end up with a non-overlapping `*)`. Nested comments are accepted.

Keywords and special characters The following keywords are reserved by the language and may therefore not be used as identifiers:

```
= " ; , ( ) [ ] let true false (* *)
```

The following characters may be used to separate keywords or to indent the scripts: ' ' '\t' '\n' (ASCII decimal: 32, 9, 10).

3.7.1.2 The primitives of the SuMo language

Here we will briefly describe the role of the different functions that are available in SuMo version 4.4, without giving details about their expected arguments and options. For this, a detailed description is given by the interactive help of the program. This help is accessible through the `help` function. The following shows the effect of the `help ()` command at the beginning of an interactive session:

```
[pc-bioinfo1] ~/devel/sumo/src/sumo % ./sumo -i
        SuMo version 4.4-Boom

Reading PDB definitions from ‘/home/martin/.sumo/pdb_groups’... done
Reading SuMo definitions from ‘/home/martin/.sumo/sumo_groups’... done
sumo> help ();
HELP TOPICS: automatic_read compare exit help list_idents multi
             multiselect output_prediction output_std print print_custom
             print_idents read remove screen site_stat_analysis
             site_statistics store update_db_summaries
AVAILABLE FUNCTIONS: automatic_read compare exit help list_idents
                     multi multiselect output_prediction output_std print
                     print_idents read remove site_stat_analysis
                     site_statistics store update_db_summaries
Try ‘help "help";’ for details on this function.
- : Unit
sumo>
```

`help x` give information about topic `x` if it exists. Table 3.3 page 100 shows the role of the available functions.

Table 3.3: Functions of the SuMo language

Identifier	**Description**
`automatic_read`	Extraction and recording of the ligand binding sites
`compare`	Comparison of sets of preprocessed 3D structures
`exit`	Exits the program with 0 or with a specified error code
`help`	Interactive help
`list_idents`	Lists all identifiers
`multi`	Multiple comparison from results of pairwise comparisons
`multiselect`	Removal of families of characteristic sites that are considered as very close to a family of higher score
`output_prediction`	Computation and export of the prediction of functional sites for other applications
`output_std`	Export of the comparison results for other applications
`print`	Displays data
`print_idents`	Lists all identifiers and displays their values
`read`	Preprocessing of the file containing the structural data into data that are directly usable for comparisons
`remove`	Removes a file
`site_stat_analysis`	Regeneration of some statistic data that are recorded in the database

Identifier	Description
`site_statistics`	Comparison of all ligand bindings sites of the standard database against each other
`store`	Records preprocessed structural data
`update_db_summaries`	Updates some global information about the database

3.7.1.3 System for 3D selection

The system for selecting the chemical groups in a given structure consists in a specific language. A (usually short) program which is written in this language defines a predicate that filters chemical groups from a given 3D structure.

This language is used by the `-select` option of the `read` command of SuMo, and a similar language is used by the `-restrict` option.

This language is based on the combination of basic predicates with the classic boolean operators `and`, `or` and `not` or the `around` predicate that takes as arguments one predicate and some numeric parameters.

The basic predicates let us select a given chemical group according to the different kinds of data it carries or that are found in its environment.

The basic predicate constructors Some predicates are PDB-based and are relevant only on chemical groups that derive from PDB files. However, PDB is the only external format that is currently supported by SuMo.

The type of chemical group The `Group` constructor selects a given type of chemical groups:

E.g.: `Group "aromatic"`
`Group "pdb_atp"`

The label of the chemical group The `Label` constructor selects chemical groups with a specific label:

E.g.: `Label "backbone"`

The index of the molecule The number of the molecule that is assigned by SuMo can be selected, for example if we want to select a given ligand:

E.g.: `Molecule_index 3`

The PDB name of the monomer The `Pdb_group` constructor selects chemical groups that are associated with the given kind of PDB group such as an amino acid:

E.g.: `Pdb_group "CYS"`

The PDB index of the monomer The `Pdb_index` constructor selects the number of the PDB group of which the chemical groups depend:

E.g.: `Pdb_index 57`

A range of indices can also be specified:

E.g.: `Pdb_index 50 to 60`

The PDB name of the chain The PDB name of the current chain can be tested with the `Pdb_chain` constructor:

E.g.:
```
Pdb_chain "A"
Pdb_chain ""
```

The `around` operator The `around` operator has been designed for the selection of chemical groups that have at least one target location that is located at a given distance of the functional location of the chemical group being considered when this one validates a given predicate. The arguments are a range of distances and a predicate:

E.g.:
```
4.5 around Molecule_index 5
[1.,3.]  around Group "imidazole"
4 around Group "pdb_atp"
```

The range of distances can be given as only one upper bound, which means in this case that the lower bound is 0, or as an interval of the form $[d_1, d_2]$.

The classic boolean operators The boolean operators can be used to combine predicates. These operators are: `and`, `or` and `not`.

Priorities The priorities of the operators are the following:

`not` > `and` > `or` > `around`

Parentheses can be used to override these priorities.

Complex examples The selection of the regions that interact with at least two atoms of calcium such that the distance between these two atoms is at least 3 Å and at most 7 Åcan be performed with the following predicate:

```
(4 around (Group "pdb_ca" and [3.0,7.0] around Group "pdb_ca"))
```

In a similar way, we can select the sites that bind a molecule of ATP that does not interact with an Mg^{2+} ion:

```
(4 around (Group "pdb_atp" and not 4 around Group "pdb_mg"))
```

3.7.2 CGI/HTML interface

The regular interface to SuMo is the web-based interface. Queries may be submitted interactively in several steps or directly by sending a query in the SuMoQ format.

3.7.2.1 Interactive queries

An *interactive query* consists in several successive steps, from the input of the structural data until the detailed analysis of the results. The steps are the following:

1. Loading structural data from a PDB identifier or an uploaded file in the PDB format

2. Proposing combinations of predefined selections (chains or ligand binding sites)

3. Displaying the chemical groups that were finally selected while indicating those that will effectively be taken into account in the comparisons. Database is selected. Subset of the database may be chosen. Comparisons are launched.
4. Displaying the status of the job until it finishes.
5. Displaying the global results under different possible formats. Results may be saved or exported.
6. Displaying details on a selected site.

All of this is better illustrated by a visit into the SuMo server at:

```
http://sumo-pbil.ibcp.fr
```

Any comparison query that uses the SuMo web server includes a transparent or explicit formulation of the query in the SuMoQ format. In any case, the query in the SuMoQ is accessible and can be saved. It provides the following advantages:

- This file serves as a "materials and methods", i.e. it describes in a non ambiguous manner what has been submitted to the SuMo server.
- Storing only the query is much more compact than storing the results.
- As opposed to comparison results, a query remains valid when the version of SuMo changes. Keeping the query file in a safe and fast way to update the results.
- The automatically generated file can be used as a model for generating more specific queries that cannot be made in the interactive mode.

Next section describes the language for writing queries, SuMoQ.

A limitation of this system is that all the structures that are used must be available and identifiable by the web server. A temporary code is given to each structure which is uploaded by a user. This code may be used later as long as the data are kept by the server.

3.7.2.2 SuMoQ queries

Introduction The SuMoQ query language allows the formulation of complex queries in one step from the client's side. Here is an example of a simple query:

```
{
  email = "martin_jambon@emailuser.net";
  title = "My query for scanning ligand binding sites";
  {
    subtitle = "Lectin 2PEL";
    scan = {
      database = "ligands";
      pdb_id = "2PEL";
      selection = "Pdb_chain \"A\"";
    } /scan;
  } /
}
```

This query means that a comparison of chain A of PDB structure 2PEL against all ligand binding sites of the database must be performed. An email will be sent when the comparisons effectively start and when they finish; it gives the URL where the results can be browsed. Titles and subtitles are optional. If specified, the subtitle field is given in the `Subject` field of the emails.

Syntax The syntax of SuMoQ was initially designed for SuMo for the representation in a tree format of the structural data from the PDB. This syntax, like XML, allows the representation of many different kinds of data as long as they can be expressed in hierarchic format. This syntax is simpler and more readable than XML, and the basic values are typed. Here is a summary of the different features of this syntax:

- Each node has a given number of fields. Fields can either be anonymous or labeled and are associated with some data.
- One node cannot have different fields with the same label but may contain several anonymous fields.
- Data are either an atomic value or a node.
- Atomic data are typed. 5 types exist.
- Tags at the end of fields are optional but must be correct when specified.
- Identifiers can be used to avoid rewriting some data.

This syntax is described below. Terminal symbols are in lowercase, in bold face and colored. Non-terminal symbols are in uppercase. Regular expressions for characters (CHAR) and character strings (STRING) are not given.

Comments are opened with `(*` and closed with `*)` and can be nested. Spaces, tab stops, carriage returns and line feeds are ignored outside of CHAR and STRING.

```
TREE     ::= INT
           | FLOAT
           | BOOL
           | CHAR
           | STRING
           | { ((LABEL =)? TREE (/LABEL)?;)* }
           | IDENT
           | define IDENT = TREE
INT      ::= -?[0-9]+
FLOAT    ::= -?[0-9]*.[0-9]*
BOOL     ::= true
           | false
STRING   ::= ". . . "
CHAR     ::= '. . . '
IDENT    ::= [a-z][a-z,A-Z,_,0-9]*
LABEL    ::= IDENT
```

The main advantage of this format against XML is that the files are at the same time easy to read and easy to edit by hand since closing tags are optional. Anonymous fields allow the definition of sequences of objects without writing a pair of tags like `<item>` and `</item>` for each element of the sequence. For instance, a sequence of integers can be written as `{1; 83; 2; 122}` instead of `<list> <n>1</n> <n>83</n> <n>2</n> <n>122</n> </list>` in XML.

Any language which is based on this syntax can give any meaning to the fields.

Semantics of SuMoQ In SuMoQ, some fields are mandatory and some others are optional. Some fields such as `email` are mandatory only in a given context. The order of the fields never matters.

A SuMoQ query can contain several comparison requests. These requests are treated independently by the server in an undefined order.

The complete specification of the SuMoQ language is given by the figure 3.15 page 108. The email field must be specified is the query asks for scanning at least one whole database. If the server considers that the query is too heavy, it may refuse it. For example, the size of a substructure that can be used for scanning the database of target structures is limited to a certain number of triplets. This parameter depends on the configuration of the SuMo server.

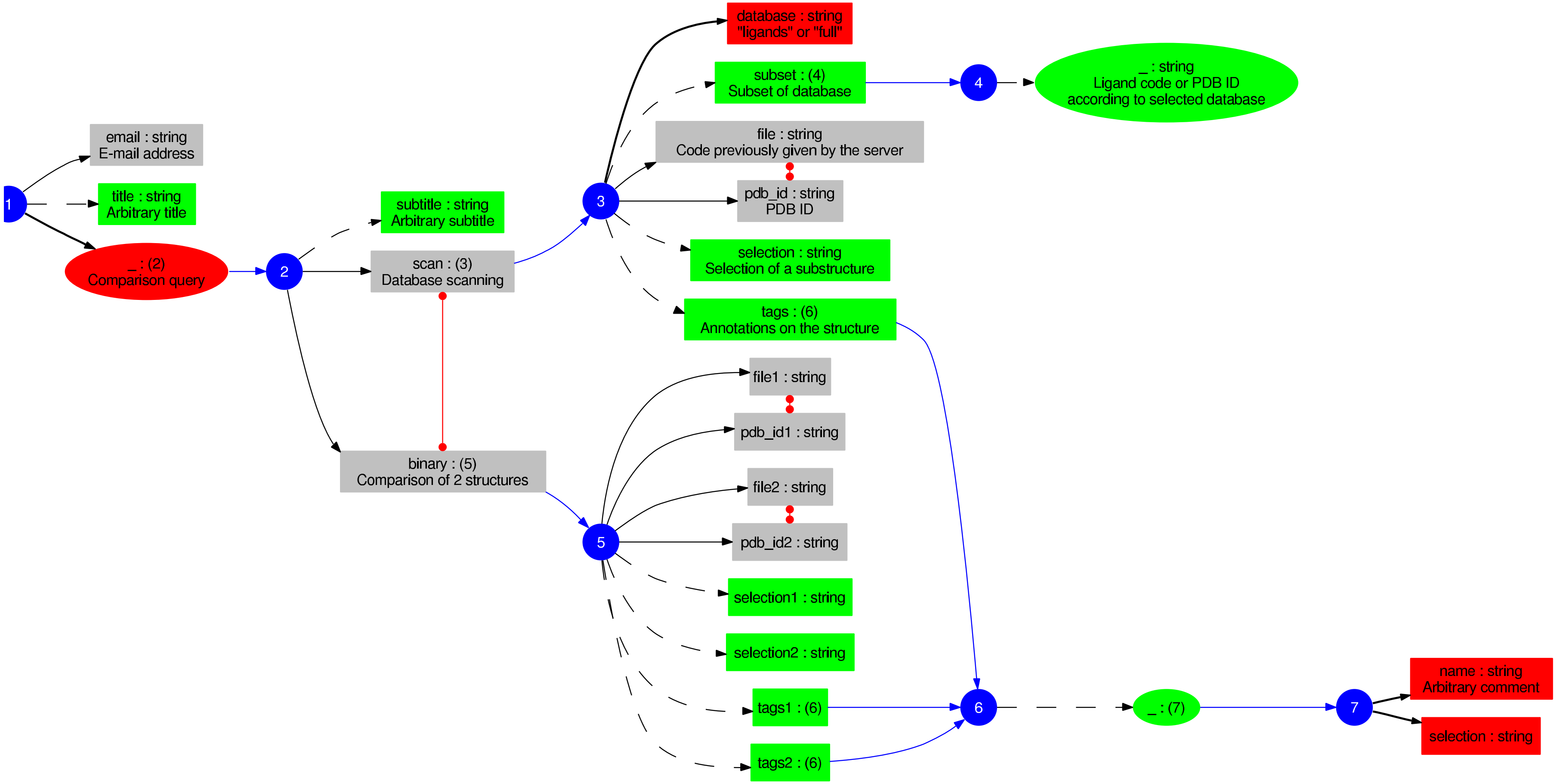

1
email : string
E-mail address
title : string
Arbitrary title
_ : (2)
Comparison query
2
subtitle : string
Arbitrary subtitle
scan : (3)
Database scanning
binary : (5)
Comparison of 2 structures
3
database : string
"ligands" or "full"
subset : (4)
Subset of database
file : string
Code previously given by the server
pdb_id : string
PDB ID
selection : string
Selection of a substructure
tags : (6)
Annotations on the structure
4
_ : string
Ligand code or PDB ID
according to selected database
5
file1 : string
pdb_id1 : string
file2 : string
pdb_id2 : string
selection1 : string
selection2 : string
tags1 : (6)
tags2 : (6)
6
_ : (7)
7
name : string
Arbitrary comment
selection : string

Figure 3.15: Specification of the structure of SuMoQ queries. Numbered circles indicate the different kinds of nodes. Type (1) is the root node. `field : type` indicates a field which has a `field` label and type `type`. _ is an anonymous field. Anonymous fields are drawn as ellipses. There can be an undefined number of anonymous fields. Mandatory fields are drawn in red, optional fields are in green and fields that are sometimes required are in grey. Unoriented red connectors indicate that exactly one of the connected fields is required.

Fields `selection`, `selection1` and `selection2` must be associated with a character string that defines a predicate for selecting chemical groups as described section 3.7.1.3 page 101.

Fields `tags`, `tags1` and `tags2` are annotations of arbirary regions that are selected with a given predicate and labeled with a free comment.

The comparison of 2 complete structures is possible by using the `binary` field. This kind of comparison is not proposed in the interactive mode since the interpretation of the results are usually difficult for a non-experienced user.

3.7.2.3 Display of the results

HTML The results of a comparison query are displayed as an HTML page. Each pair of matched sites that were detected by SuMo constitutes a line of a table that contains essential information such as the title and the short description that were found in the PDB files. The table is split into different pages when more than 50 lines must be displayed.

Each pair of matched sites can be expanded into an individual page with more details and projections of the matched sites. Matched amino acids are highlighted and both sites are oriented so that the first site is visible and the second site follows the orientation of the first site after superposition. The orientation of the first site is performed so that the mean of the vectors $\overrightarrow{CP}$ as defined section 3.2.1.2 page 49 is perpendicular to the screen and points toward the observer. The images are generated after the coordinates have been modified with MolScript [25] and Raster3D [2, 34, 33]. Scripts are also generated for 3D view under RasMol. Links to various external sites are also proposed.

Two other kinds of display are also proposed. The list of matched ligands, sorted by decreasing score, can be displayed. The prediction of ligand binding sites can be shown by the annotation of the chemical groups of the query structure and the use of the specificity function that is defined section 3.5 page 74.

Saving the results Either the results can be saved and later uploaded to the server or only the query identifier can be used to access the results during the period when they are kept by the server.

Format The data are saved in the format of Marshal, the serialization module of Objective Caml. These data do not contain any type information and there must be system that checks that they are safe for the server.

Cryptographic signature In order to avoid possible crashes due to incorrect data, it is necessary to check that the data have been generated by the SuMo web server.

This is achieved by adding a cryptographic signature at the beginning of the file. The protocol that has been designed for generating the signature is the following:

1. An MD5 digest of the file is computed.
2. The digest is encrypted.

The Digest module of Objective Caml is used for the generation of the MD5 digest. This signature is encrypted using the Cryptgps library. The key that is used for the encryption must be known only by the server. This key changes everytime the format of the results changes or more often for security reasons. Encrypting the signature instead of the whole file gives the possibility for any client program to read the data just by skipping the encrypted signature at the beginning of the file. However it is practically impossible to generate arbitrary data and the correct signature for these data at the same time.

Restriction to the best results The parameters of SuMo have been chosen so that the results are satisfying for the users in the largest number of cases as possible. But given the large volume of the results and that some users only want to keep the top scores, it is possible for them to download only the results whose score is above than a user-defined threshold. This functionality does not allow the regeneration of a predictive annotation of ligand binding sites using this new threshold.

Export The results can be exported into different formats that cannot be read back by the server.

SuMo version 4.5 proposes the following formats:

- XML format. It contains the same information than the data that are saved but in a portable, text format.
- Spreadsheet format. Lines and columns.
- Free text. Human readable text, not supposed to be read by a program.

XML format The XML format is generated automatically from the type information that is found the implementation files thanks to the IoXML [16] Camlp4 syntax extension.

Spreadsheet format A column-formatted file is generated from the comparison results. It provides more information than what is displayed on the web page, can be customized and can be loaded from a regular spreadsheet software.

The format can be modified by the user in different ways:

- Custom column separator.
- Ordered selection of the fields that must be exported.
- Optional header with the specification of the columns.
- Selection of some lines according to the contents of site annotations.

The character string that is used by default is a semicolon. It can be changed into anything else by filling out the ad hoc HTML form.

The number of possible fields is quite large. In order to reduce the size of the files, a user can specify only the columns that he wants by giving a format which is a string in which the fields are specified by `{`*id*`}` where *id* should be the name of a field. If *id* is not a valid field, it is substituted by a question mark.
E.g.

```
{molecule_identifier2} & {molecule_name2} & {sumo_score} \\
```

This format generates a LaTeX table with 3 columns. Lines will look like this:

```
1BOU & ATP & 2.69744 \\
```

The name of the different possible fields is given by default when no option is selected, since all fields are exported as well as the format string.

The annotations of sets of chemical groups may have been performed in the database and appear in the results. Let us recall that an annotation is a pair (E, S) where E is a set of chemical groups and S is a comment. Annotations are made automatically when the database is built or by the user when he defines queries in the SuMoQ format. Each chemical group may belong to an arbitrary number of annotations. The intersection between an annotated set of chemical groups and a given site is called *intersection of annotated region.* The intersection of annotated region of maximal size for a given site is called *maximal annotation.* The maximal size is in fact estimated using 3 different ways:

- volume of the SuMo chemical groups,
- sum of the coefficients of the chemical groups,
- number of chemical groups.

Some parameters of the maximal annotation of each site being considered are represented as specific fields. These fields contain the comment from the annotation and the size information. It is possible for the user to extract only the sites that match a specific annotation pattern by defining “Tag filter 1” (query) or “Tag filter 2” (target). Any site that does not match the given regular expression is ignored.

The format for these filters consists in a sequence of regular characters and special characters which are `*` for any sequence of characters, `?` for one or zero arbitrary character and `|` for separating patterns when either of the patterns must be matched.

E.g.: `sacch*binding|gluc*binding`

Free text format The only purpose of this format is to group all the results into a single, human-readable format without requiring any specific viewing tool. All matched chemical groups for all the matched sites are exported.

This type of file is generated directly by the `print` function of `sumo`.

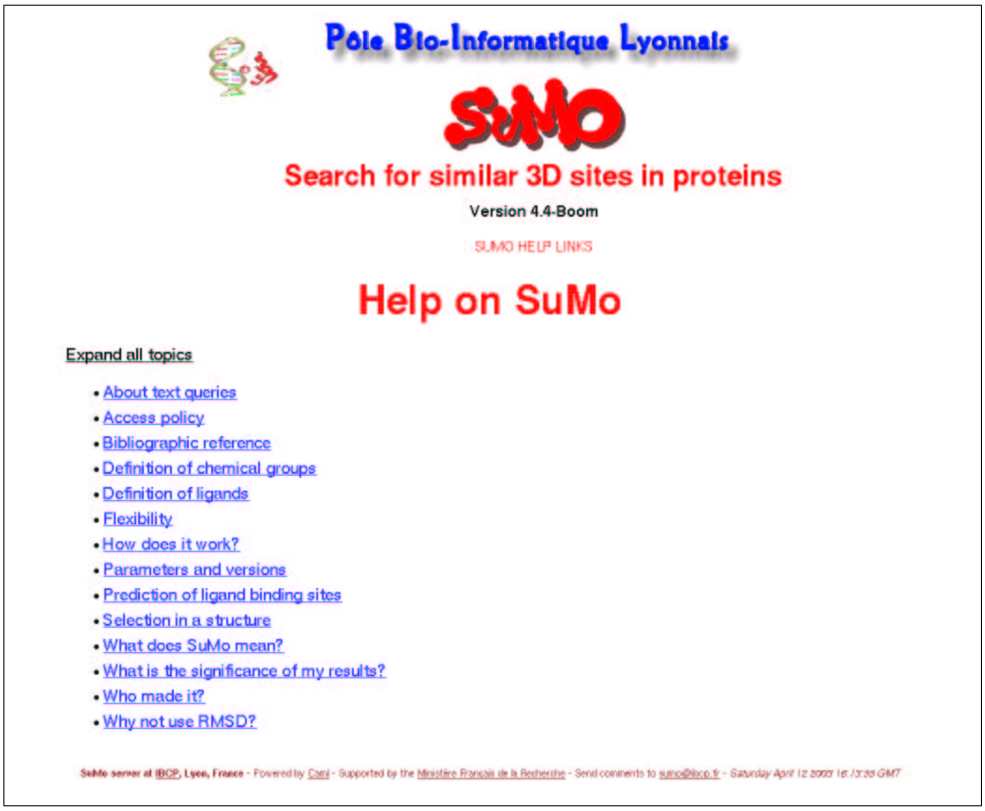
3.7.2.4 Online help

A system of online help has been designed. Its purpose is to answer most frequently asked questions. The table of contents is accessible from the HELP link of any SuMo web page.

Some terms that can be found in the different web pages of SuMo are not defined in situ for readability reasons but may be accessed directly by clicking the asterisk. All the help topics of SuMo version 4.5 are reproduced on page 155 (of the full paper version of this report).

3.7.2.5 Development of the system

In this section we will see some important points in the development of the web interface of SuMo.



Pôle Bio-Informatique Lyonnais

SuMo

Search for similar 3D sites in proteins

Version 4.4-Boom

SUMO HELP LINKS

Help on SuMo

Expand all topics

- About text queries
- Access policy
- Bibliographic reference
- Definition of chemical groups
- Definition of ligands
- Flexibility
- How does it work?
- Parameters and versions
- Prediction of ligand binding sites
- Selection in a structure
- What does SuMo mean?
- What is the significance of my results?
- Who made it?
- Why not use RMSD?

SuMo server at IBCP, Lyon, France - Powered by Caml - Supported by the Ministère Français de la Recherche - Send comments to sumo@ibcp.fr - *Saturday April 12 2003 16:13:55 GMT*

Figure 3.16: Table of contents of SuMo's online help

Languages and external libraries Like the rest of the SuMo system, all programs for the web interface were written in Objective Caml. Table 3.4 page 114 shows the different external programs and libraries that are required for the compilation or the execution of the system. Common utilities such as GNU `make` or `bash` are not mentioned.

Table 3.4: External software that is used for the development of the SuMo web server

Name	**Nature**	**Function in SuMo**
Objective Caml	Compiler	Compilation of all programs
Camlp4	Preprocessor	Expansion of syntaxic extensions Printfer and IoXML.
OcamlMakefile [35]	Makefile	Generic Makefile for the compilation of complex OCaml projects.
IoXML	Library	Syntax extension of Caml for the generation of XML code from arbitrary data types.
Cryptgps	Library	Cryptography library used for the generation of encrypted signatures for data that are saved by the users.
Ocamlnet [45]	Library	Parsing of CGI parameters. Emails.
MolScript	Executable	First step in the dynamic generation of images of matched sites.

Table 3.4: (continues)

Name	Nature	Function in SuMo
Raster3D	Executable	Second step in the dynamic generation of images of matched sites.

Printfer syntax extension

Motivation In the Caml language as well as in C and many other programming languages, a function named `printf` can be used by giving a format string and a list of arguments of different types that will fill the gaps in the format string.

E.g.: `printf "We are in %s %i.\n" month year;`
`%s` will be replaced by the string which is bound to `month`, and `%i` will be replaced by the string representation of `year` in decimal notation.

When the length of the text increases as well as the number of gaps to be filled, it becomes difficult to insert new gaps and the correct argument at the right place.

E.g.:
```
printf "On %i-%02i-%02i at %i:%i..."
  year month day hour minutes;
```
is not convenient when we need to swap some arguments.

When writing a text producing application, it is important to keep the source code that generates the text as close as possible to the final result. Another obstacle of regular OCaml for this kind of application is that the characters " and \ must be escaped but may may occur frequently in a target language such as HTML.

Syntax We need a syntax that allows us to place the format specification and its argument at the place where it will appear in the text. Camlp4 is a tool that allows the definition of syntax extensions over the regular syntax of OCaml. A Camlp4 quotation has been defined. This extension has been called *Printfer*. It has been designed in particular for CGI programs that produce HTML code, but can serve as a replacement for the `printf` function in any context. Here is an example:

```
<< Le $%02i{jour}/$%02i{mois}/$%i{année}, je vous donne rendez-vous,
Monsieur $name{nom}, à $int{heures} heures $int{minutes}.
Je vous invite à consulter la <a href="$doc_url">documentation</a> au
préalable. >>
```

will be expanded into the following Caml code:

```
Pervasives.print_string "Le ";
Printf.printf "%02i" jour;
Pervasives.print_string "/";
Printf.printf "%02i" mois;
Pervasives.print_string "/";
Printf.printf "%i" année;
Pervasives.print_string ", je vous donne rendez-vous, \nMonsieur ";
print_name nom;
Pervasives.print_string ", \224 ";
print_int heures;
Pervasives.print_string " heures ";
print_int minutes;
Pervasives.print_string ".\nJe vous invite \224 consulter la <a href=\"";
Pervasives.print_string doc_url;
Pervasives.print_string "\">documentation</a> au\npr\233alable."
```

In this example, the only function that is not predefined is `print_name`.

We will not give a complete description of the syntax, but only the main features. These features are listed in table 3.5 page 116.

Construct	**Resulting Caml code**
`$`*id*	`Pervasives.print_string` *id*`;`
`${...}`	`Pervasives.print_string (...);`
`$`*id*`{`*arg1 arg2 ...*`};`	`print_`*id* *arg1 arg2 ...* `;`
`$%`*format*`{...}`	`Printf.printf "%`*format*`" (...);`

Table 3.5: Main constructs of the Printfer syntax extension

Variants of the previous constructs exist. If square brackets `[ ]` are used instead of curly ones `{ }`, then `output_`*id* `Pervasives.stdout` is used instead of `print_`*id*. Normally, any valid Caml expression can be placed between the brackets, except comments that are treated in a slightly simplified way. Using `$>` instead of `$` flushes the standard output before evaluating the expression. A `$` character can be produced with `$$`. Strings `<<` and `>>` can be produced with `\<<` and `\>>`. `\` can be produced with `\\`. Heading and trailing spaces are ignored.

The syntax extension that was developed is used for producing HTML code from all the CGI programs of the SuMo server.

3.8 Task management

We will see in this section how the SuMo system manages the different tasks: in which order and with which physical resources.

3.8.1 Job queue

A utility for the management of a queue on a given host was written. This program was called `jobqueue` and has the following properties:

- it does not require to start a background process when the machine is started,
- included in the SuMo distribution,
- priorities are based on the estimation of how much time each job will require.

3.8.1.1 Architecture

3 kinds of processes are involved in the task management:

1. Some processes record the queries, i.e. create a file that contains execution directives in the directory that was assigned to the queue. These processes can start the task manager if this process is not running yet.
2. A central background process or *daemon* is the task manager. Its role is to watch all the pending requests when a process signals him that something new happened, and starts the processes that execute the tasks.
3. The processes that start the tasks are created by the task manager. Their role is to start the jobs and catch their exit status. These processes can also send emails when the job is started and when it ends.

The interactions between these processes are illustrated on figure 3.17 page 118.

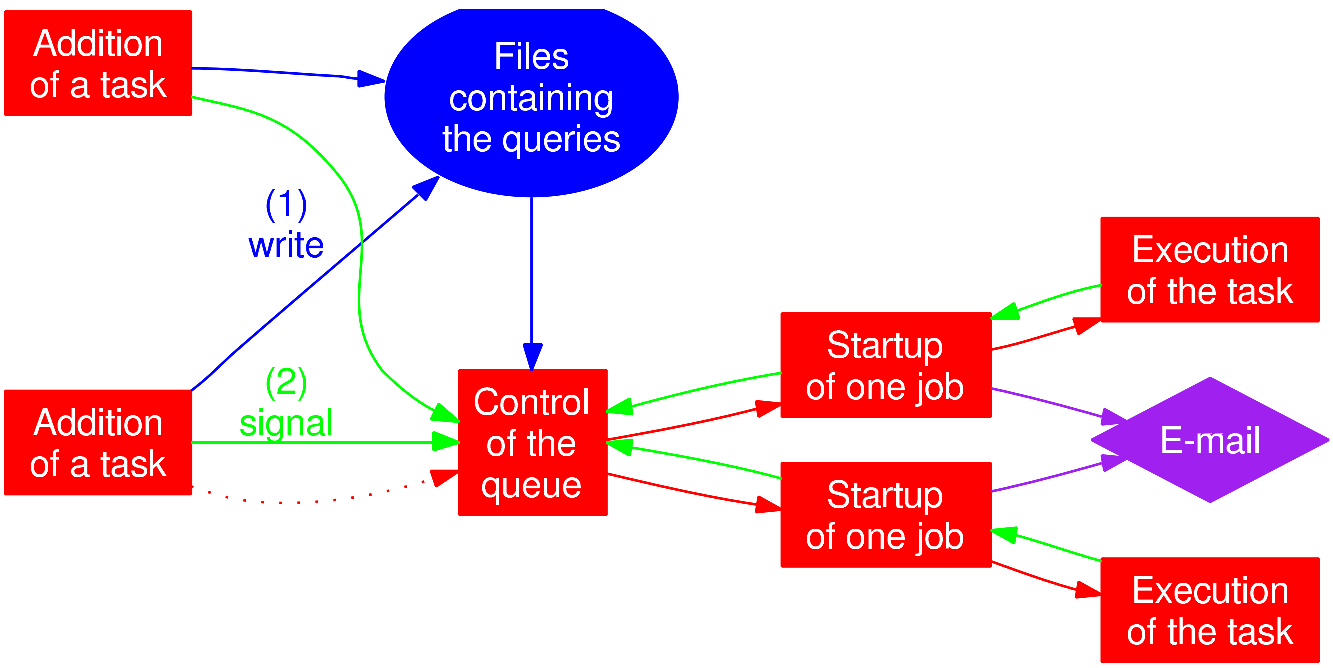

Addition of a task
Files containing the queries
(1) write
(2) signal
Addition of a task
Control of the queue
Startup of one job
Execution of the task
E-mail
Startup of one job
Execution of the task

Figure 3.17: `jobqueue` job management system. Red arrows represent the creation of a new process with the fork/exec mechanism. Green arrows represent interprocess signals.

3.8.1.2 Priority management

Property 3 *The job queue was designed so that on average, the waiting delay is proportional to the length of the job.*

In order to satisfy property 3 page 119, an estimation of the length of every task must be given when the request is submitted. The time unit does not matter as long as all the requests use the same unit for a given queue. Every time the task manager catches a signal, dynamic priorities are updated. These priorities depend on how long the requests have already been waited.

$$\text{priority} = \frac{\delta}{e}$$

where e is the estimated length of the task and δ is the waiting time. When a task can be started, the task manager will choose the task which has currently the highest priority.

It is possible to give a higher priority to one task by underestimating its length, and inversely give it a lower priority by overestimating their length.

This system has the advantage of not blocking small tasks when a long task has been submitted. In SuMo there is often a time factor of 1000 between tasks that are submitted.

3.8.1.3 Daemon startup

The task manager is started by the user of the job queue when a task is added for the first time. The maximum number of simultaneous tasks must be specified. The name of the queue is a directory in which the user has read and write access.

3.8.2 Parallel execution on a multi-processor machine

The comparison function of SuMo has been parallelized for taking profit from multi-processor computers. This parallelization does not require any specific library. It consists in dispatching the pairwise comparisons into small groups that will be handled by children processes. This processes write the results into a file when they are finished. The parent process harvests and merges the results once all the subtasks have been performed.

> *The function that parallelizes the job is polymorphic and could be reused for managing any kind of subtasks. The only condition is that the result must be convertible with the serialization functions of the Marshal of the standard library of Objective Caml.*

3.8.3 Distribution of the tasks over a cluster of computers

The `jobqueue` task management system works only on a single host, not on a *cluster* of hosts. The implementation of SuMo version 4.4 provides an interface to PBS system. PBS is currently used by the public SuMo server.

3.9 Frequently asked questions (FAQ)

3.9.1 About the method

Question 1 – Why use triplets of chemical groups?
Several considerations lead to use groups of three chemical groups rather than one, two, or more than three.

- In k dimensions, a set of k non aligned points defines a hyperplane of $\mathbf{R}^k$. A hyperplane has the advantage of being defined from a unique vector. In 3 dimensions, the hyperplane is a plane defined from 3 points.
- The use of triplets as vertices of the graph allows the addition of an angular information on each edge. This is important for detecting superposable substructures. But if individual chemical groups were used intead of triplets, it would not be possible to directly know if matched graphs A–B–C–D and A'–B'–C'–D' represent a good structural match or not. If the vertices were standing for more than 3 chemical groups, there would also be no notion of angle.
- There is no need to group the chemical groups by more than 3.
- There are approximately 1000 different types of triplets of chemical groups. This allows the use of a hashing technique while searching for matches.
- In 3 dimensions, 3 points are necessary and sufficient to anchor a ligand (locally) with no possible rotation. Therefore, it seems to be a not too restrictive idea to consider that any significant ligand requires at least 3 anchor points. Other functional properties than ligand binding are of secondary importance in SuMo.

Question 2 – Is the flexibility of the lateral chains taken into account?
Yes, if the fluctuations are reasonable. SuMo does not perform molecular

dynamics. If several conformations are known to be acceptable for a given molecule, each of them must be submitted independently to SuMo. Currently, only the first structure of a PDB file containing several structures is taken into account by SuMo.

Question 3 – Can we change any parameter?
One of the goals of SuMo is to be easy to use. Users of the web interface do not have the possibility of changing parameters for several reasons, including simplicity of interpretation. At the level of the command-line `sumo` tool, several parameters can be changed as well as the definition of the chemical groups. Changing the parameters however requires a deep knowledge of SuMo and almost nobody would be able to extract better results by simply changing a few numeric parameters.

Question 4 – Did you perform any statistical validation?
There cannot be any universal validation of a tool that solves a problem which is not expressed under mathematical terms. Of course some scoring functions may help users to sort the results that were obtained with SuMo. For example, a function that tries to estimate a specificity of a chemical group or a site for a given family of ligands has been defined (see section 3.5 page 74). Alternate approaches could simply ask users if they are satisfied with the results of SuMo.

Question 5 – Can SuMo consider as similar chemical groups of different types?
No. Each chemical group can only match a chemical group of the same type. Each type of chemical group should model one abstract property and a given set of atoms can be used to define several chemical groups at the same location but that model different properties. For instance, an imidazole cycle from a histidine carries some catalytical properties that are not found in other aromatic rings but at the same time can stack with other aromatic rings. These properties will be modeled with the `aromatic` and `imidazole` types of chemical groups that have the same physical location but different geometric representations (see section 3.2.1.2 page 48).

Question 6 – Are specific interactions such as aromatic/guanidinium stacking taken into account?
Yes. It is not necessary to define a specific type of chemical group for this kind of interaction since chemical groups `aromatic` and `guanidinium` are already defined with a geometry that is suitable for detecting these interactions. More generally, any interaction between chemical groups is taken into account by SuMo if the chemical groups are modeled correctly.

Question 7 – Can SuMo be used on homolgy-based models of protein structures?
Yes, but the relevance of the results will depend on the quality and the resolution of the model, exactly like for structures obtained by crystallography or NMR spectroscopy. In many cases, it is probably more relevant to work directly on the structures that were used to build the model instead of the model itself.

Question 8 – Do several models produced by molecular dynamics produce the same results with SuMo?
It depends on how the simulation disturbs the structure. SuMo is able to detect similarities between structures of identical proteins that were determined independently. If a dynamics does not change the conformation more than the accepted experimental errors, then the results of SuMo at different dates of the simulation will be very similar.

3.9.2 About the implementation

Question 9 – Isn't the choice of Caml as the main programming language a problem for industrialization?
Asking such a question before the development of SuMo is perfectly relevant and should be the case for every programming language. Before starting a project, one can have doubts about the quality of the compilers, the easiness of the language and the availability of some libraries. But asking this question when the project is almost finished and sold does not make much sense.

Question 10 – Isn't Caml slower than C++?
The reader can check (carefully) one of the following URLs to find some possible answers to this question:

```
http://shootout.alioth.debian.org/
http://www.bagley.org/~doug/shootout/
```

Another point is that the whole SuMo system (version 4.4, 30,000 lines) was designed, implemented and tested in three years by one person using Caml.

Chapter 4

Results of comparisons

Various tests have been performed with SuMo 4.4 and return promising results. However, the assessment of their biological relevance requires a certain investment. We will present here the main lines of a work that was performed in collaboration with Anne Imberty (CERMAV) and published [23]. Then, instead of presenting a small number of biological examples, we will show the results that were obtained after the comparison of all the ligand binding sites against each other.

4.1 Family of the legume lectins

Lectins are proteins that bind oligosaccharides reversibly [31, 32]. The family of the legume lectins has at the time of the study (September 2001) 106 members in the PDB and most of them (94) really function as a lectin at the expected site but some of them (12) do not have this property [32]. These are:

- 2 structures of the arcelin, a defence protein;
- 2 structures of α-amylases,
- 8 structures of demetallized lectins, i.e. deprived of the Ca^{2+} and Zn^{2+} ions that are required to stabilize the sugar binding site.

In order to test the quality of SuMo, the 3D site that binds the oligosaccharide in the typical structure 2PEL was delimited. It was defined by all the chemical groups that had at least one atom which was within 4Å around one of the atoms of the ligand. Here the ligand is a disaccharide, the lactose (LAT). This site is shown on figure 4.1 page 124. The search for similar members was performed among the 106 members of the family. Any matched site

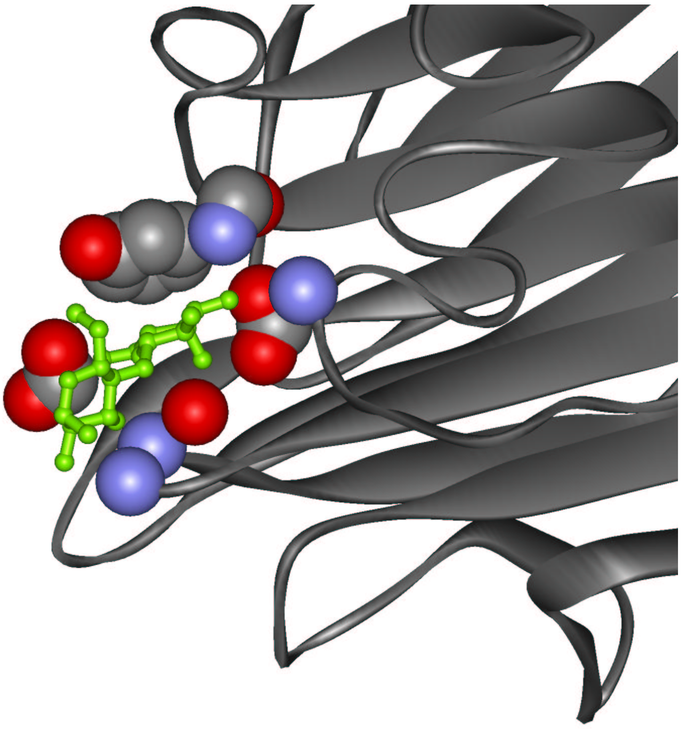


Figure 4.1: View of the site that was selected in PDB structure 2PEL and which was used to scan the legume lectin family. Spheres represent atoms that were used for building the selected chemical groups. The green ligand is a molecule of lactose

was considered as positive. The results of these comparisons is summarized on figure 4.2 page 125. Here is the numeric version of the results:

- 90 true positives,
- 12 true negatives,
- 0 false positive,
- 4 false negatives,

i.e. a success of 102/106 (96 %) if we consider the experimental data. 4 true lectins were not detected by SuMo; these results are shown and discussed in details in the initial publication about SuMo [23]. It must be noticed that at this time, SuMo version 2.0 (July 2001) was used. In this early version of SuMo, each chemical group was represented only as a single point without specific geometry. There was also no distinction between functional and physical locations, and free hydrogen bond donors or acceptors were not taken into account. Moreover, a final filter used the RMSD as a criterion for selecting the sites but this was removed from version 4.0.

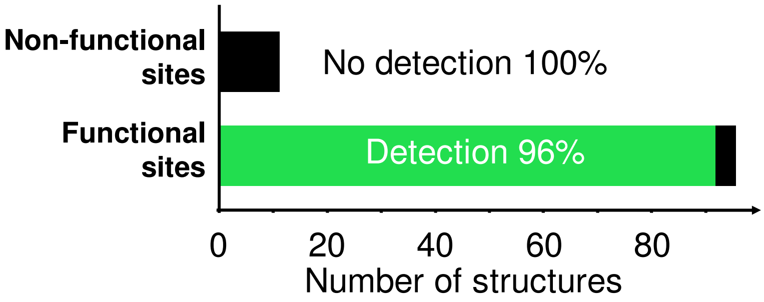


Figure 4.2: Results after scanning the legume lectin family with the known lectin site of structure 2PEL. Functional sites are those that bind an oligosaccharide, which may possibly be co-crystallized. Non-functional sites are all other sites and come from proteins that do not have a known lectin activity for the site of interest in the given experimental conditions.

4.2 Systematic comparison of sites

4.2.1 General data

The pairwise comparison of all ligand binding sites according to the protocol which is described section 3.5.4 page 76 was performed over the 11,292 sites that were available at the time of the computation. 1,836 different ligands were involved. We saw that the specificity function can be computed only when the site being considered matches at least 10 other sites of the same family. This drastically restricts the number of sites and of ligands that are considered. Thus 3,299 sites over 37 families of ligands were considered.

4.2.2 Definition of families of ligands

Only two families containing several ligands are currently defined. The first one is called *monosaccharide*, the second one *ATP-family*. The exact definition of these families is given figure 4.3 page 126. These definitions were made manually, according to the apparent similarity of the different ligands, with the help of the PDBsum [28, 27] and HIC-Up [24] web servers. A larger number of families could be defined, but it is difficult to perform either automatically or manually. Let us recall that families overlap and that any single ligand automatically defines a one-member family which has the same name as the ligand.

ATP-family: ATP, ATP-MG, ATP-MN, ADP, ADP-MG, ADP-MN, GTP, GTP-MG, GTP-MN, GDP, GDP-MG, GDP-MN, GDP-CA, ANP, ANP-MG, ANP-MN, GNP, GNP-MG, GNP-MN, GSP, GSP-MG, GSP-MN, 3AN, 3AT, ABP, ADI, ATR, DAD, DAT, DG3, DGT, DTP, GPX, M7G, MDG, MGP, MGT, SAP
monosaccharide: 2DG, 2FG, ALL, ARA, ARB, BMA, FCA, FCB, FRU, FUC, G2F, G6D, GAL, GLA, GLB, GLC, GLT, LXC, MAN, RAM, RIB, RNS, XUL, XYP, XYS

Figure 4.3: Definition of the two current families of ligands

4.2.3 Results

Table 4.1 page 127 lists the 37 families of ligands that contain at least one site for which the specificity can be estimated. The mean specificity of the sites of a family is a logarithmic mean, i.e.

$$\exp \frac{\sum_{i=1}^{n} \log x_i}{n}$$

This mean should be preferred to an arithmetic mean when the space between 2 values x_i and x_j is given by x_i/x_j rather than $x_i - x_j$. For instance, the logarithmic mean of $\{0.1; 0.01; 0.001\}$ is 0.01 while its arithmetic mean is 0.037. Unfortunately, perfect results—i.e. infinite specificities—cannot be taken into account for the computation of the mean.

> *The separate mean of denominators and numerators of the elements expressed as fractions (see page 92) could be a solution to this problem, but it would give too much weight to the most redundant sites. Weights should be assigned to each term so that each family has the same importance.*

Thus the mean specificity of a family of ligands in the table is a mean apparent specificity of the sites that bind ligands of the family in question. The apparent specificity of a site S is given by function Φ_S as defined page 76. The table should be interpreted as follows:

- The specificity of family of ligands must be interpreted as a ratio between the number of true positive chemical groups and the number of false positive chemical groups, while making as if there was as many sites inside of the family than outside.
- The mean specificity cannot take into account infinite specificities. The number of sites with an infinite specificity, if any, is given in the last column.

Table 4.1: Mean specificity of the ligand binding sites. N indicates the total number of sites for each family. N_{inf} indicates the number of sites for which the specificity is infinite and is thus not taken into account in the mean. These results were obtained in March 2003, with the version 4.4 of SuMo

Family of ligands	**Code**	**Specificity**	N	N_{inf}
HRACEN-4-ONE GUANOSINE DINUCLEOTIDE	PGD	13091.30	2	
HYPOXANTHINE	HPA	1897.69	16	
THYMIDINE-3',5'-DIPHOSPHATE	THP	1308.54	7	
S-ADENOSYL-L-HOMOCYSTEINE	SAH	1186.81	16	

Family of ligands	**Code**	**Specificity**	N	N_{inf}
CARBONATE ION	CO3	1067.26	36	
NICOTINAMIDE-ADENINE-DINUCLEOTIDE	NAD	707.13	63	
2'-DEOXYURIDINE 5'-MONOPHOSPHATE	UMP	524.69	37	
BIOTIN	BTN	473.34	19	
2,5-ANHYDROGLUCITOL-1,6-BIPHOSPHATE	AHG	434.05	15	
GUANOSINE-2'-MONOPHOSPHATE	2GP	422.93	35	
NADP NICOTINAMIDE-ADENINE-DINUCLEOTIDE PHOSPHATE	NAP	349.82	32	
MALTOSE	MAL	329.06	11	
monosaccharide		328.72	7	1
FRUCTOSE-6-PHOSPHATE	F6P	242.23	13	
PHOSPHATE ION	PO4	165.53	10	
PHOSPHOAMINOPHOSPHONIC ACID-GUANYLATE ESTER	GNP	145.83	7	
COPPER (II) ION	CU	140.44	108	2
NICKEL (II) ION	NI	114.74	16	2
PROTOPORPHYRIN IX CONTAINING FE	HEM	95.33	382	87
COBALT (II) ION	CO	69.62	3	1
ATP-family		63.89	93	24
FE (II) ION	FE2	63.18	23	
ADENOSINE-5'-DIPHOSPHATE	ADP	54.38	2	
GUANOSINE-5'-DIPHOSPHATE	GDP	49.65	7	
FLAVIN MONONUCLEOTIDE	FMN	38.96	89	43

Family of ligands	**Code**	**Specificity**	N	N_{inf}
MANGANESE (II) ION	MN	35.71	123	43
CALCIUM ION	CA	31.65	1293	92
FE (III) ION	FE	29.80	57	11
SULFATE ION	SO4	19.57	14	
ZINC ION	ZN	18.75	683	53
MAGNESIUM ION	MG	12.82	14	1
HEME D	DHE	7.94	13	10
FE2/S2 (INORGANIC) CLUSTER	FES	3.16	13	11
IRON/SULFUR CLUSTER	FS4	1	16	16
ACETATE ION	ACT	1	14	14
DIMETHYL SULFOXIDE	DMS	1	9	9
POTASSIUM ION	K	1	34	34
Mean		81.56		

4.2.4 Comments

These results could have been presented in many other ways, especially by finding a way for not removing the infinite specificities. However, the goal of this approach is simply to summarize the results of systematic comparisons, in order to give a general view of the quality of SuMo.

4.2.4.1 Representativity of the ligands

Since the apparent specificity of a site for a family of ligands is computed only when it matches at least 10 sites of the same family, many ligands are not taken into account in the table:

- either because these ligands were present less than 10 times in the PDB,
- or because these ligands come with less than 10 binding sites that use the same anchoring mode,

- or because SuMo is not able to detect some similarities in the family of sites.

Ligands that are not frequent but are present under many related forms in the PDB could be grouped into families. If the ligand binding sites of a given family are similar enough, then the problem of the minimum-10-matches can be solved and the apparent specificity would increase.

4.2.4.2 Problem of the infinite specificities

Certain families of sites have an infinite specificity, which is a very good sign but not taken into account in the mean. For example, the last 4 ligands of the table all have an infinite specificity, which means that the mean specificity is irrelevant for these ligands.

4.2.4.3 Problems related to the composition of the PDB

The PDB is frequently considered as redundant by people who are interested in the classification and in the evolution of the proteins. However we will consider that the structuralists that solved several structures of the same protein probably did not work in vain, and actually these 3D structures probably present crucial differences for the understanding of the function of the protein.

We will only notice that the PDB is not representative of the abundance of the proteins in given physiological conditions, and that it is essential to distinguish the notion of biochemical specificity from the apparent specificity as defined here as a way to evaluate the capabilities of SuMo.

Chapter 5

Discussion

The SuMo system is, with the PINTS server [44], the only one that allows requests for scanning a database of 3D ligand binding sites. SuMo, compared to PINTS, introduces several innovating concepts that were successfully implemented:

- no notion of main chain vs. lateral chains of amino acids anymore,
- free hydrogen bond donors and acceptors are taken into account while ignoring the ligands,
- attribution of specific geometric constructs and parameters for each type of chemical group,
- distinction between physical location, functional location and target locations for a given chemical group,
- concept of interaction between a very flexible ligand and a macromolecule of much slower dynamics and not disturbed by the presence of ligands, as opposed to the interaction between 2 macromolecules,
- use of a scoring function which is based on the rigid protein/flexible ligand model, instead of the classic RMSD,
- notion of importance of each chemical group and radius of influence.

However, SuMo does not claim that it provides a statistical validation of the results. No solution for giving a probabilistic significance to the results of SuMo has been found yet. Moreover, results from SuMo may be used to retrict significantly the search space, e.g. for identifying ligand binding sites in proteins, but are not a definitive answer. Such results can be complemented by appropriate simulations of molecular dynamics or ligand docking based on empirical physical models.

The development of the SuMo system started in January 2000, without the initial will of building such a complex and large system. The initial goal was simply to create a system that compares the surface properties of the proteins. This approach had to be independent from protein-specific notions such as the amino acid sequence and the fold of the main chain.

The first convincing results were obtained in March 2000, since the core comparison algorithm was used on graphs of triplets of chemical groups instead of graphs of single chemical groups. In this early version, the chemical groups were simply modeled by the mean position of some atoms from a given PDB group (amino acid). The algorithm was designed in order to identify similar regions of arbitrary size between structures of proteins. The comparison is symmetric since the beginning, i.e. there is no difference between the query structure and the target structure. Therefore the current search for ligand binding sites in a query structure does not try to match full sites with one part of the query but parts of the site with parts of the query.

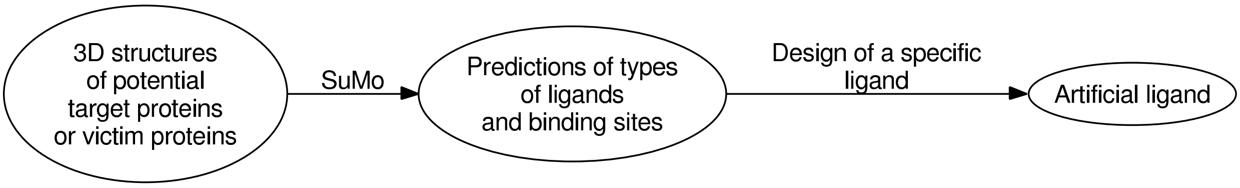
The purpose of the successive developments that followed the first interesting results was to make the tool easier to use and to improve the quality of the heuristics. SuMo is a non formal system for the analysis of experimental data: it selects some data so that they become visible for a human observer, by using a heuristics that tends to be close to the intuitions of the users. This is why the heuristics of SuMo is more and more complex. However, an approach such as SuMo over a purely human observation of data are:

1. reproducibility of the results,

2. speed.

SuMo predicts ligand binding sites only by comparing sites, but does not perform any simulation to check if the predicted ligands could effectively bind the predicted sites. This whole process belongs to the domain of *docking* simulations and structure-based *drug design*. SuMo only helps to select interesting target sites and fragments of potential ligands. The position of SuMo in a process of drug design is shown on figure 5.1 page 133.

In the following sections we will discuss the evolution of SuMo, i.e. its current capacities, the problems that arise and the nature of the obstacles to their resolution.



0]figures/drug-design/drug-design.ps

Figure 5.1: Position of SuMo in a process of drug design

5.1 Future of the software

The SuMo software is currently not distributed but can be used directly from the web site `http://sumo-pbil.ibcp.fr` for non commercial purposes. Access can be arbitrarily restricted by the administrators of SuMo and no guarantee is given to the users of SuMo.

For an intensive or commercial use of SuMo, users should make contact with the maintainers. A patent is being submitted by the CNRS. In April 2003, the CNRS, the MEDIT company and the inventors are negotiating the licensing conditions that should allow MEDIT to develop, use and distribute SuMo. MEDIT was founded in 2003 by François Delfaud and his associates.

Thus the development of SuMo should continue. We will now introduce the main constraints and difficulties for the maintenance and the future development of SuMo.

5.1.1 Usage requirements

The architecture of SuMo is essentially based on 3 levels (see section 3.1.1 page 34). The normal and convenient use of the software is done thanks to a HTTP server. It has the advantage of avoiding the installation of specific software on each client host. The counterpart of this system is that everyone has an unlimited access to the service except if a system of identification by IP addresses or passwords is set up.

The operating system that is currently used for running the SuMo server is GNU/Linux, but it should be portable to any POSIX-compliant system. A port of the `sumo` program to Cygwin, an emulation of Unix for Microsoft Windows was performed easily in 2001. Nevertheless, no other porting effort has been made since the web interface of SuMo makes it accessible from any kind of operating system.

5.1.2 Perennity

In order to be installed and run correctly, SuMo requires the following components:

- Objective Caml,
- a HTTP server,
- HTTP clients,
- a Unix-like operating system.

The operating systems is not a strong constraint since Objective Caml can run on most of the current systems, including Microsoft Windows and MacOS. Installing Objective Caml requires a C compiler. The different external libraries that are used by SuMo are written in Objective Caml and therefore can be adapted or fixed easily if necessary. All the components that are needed for the installation of SuMo are *open source*. Their simple use is granted for free to anyone except MolScript and Raster3D that are used to produce images.

5.1.3 Reusability

5.1.3.1 Heuristics

Several heuristics that were developed for SuMo are reusable. They are located in independent modules and are naturally polymorphic thanks to the type system of Caml.

However, most of the heuristics were developed precisely because they had to solve a specific problem which was strongly related to the specific conditions of SuMo. For instance, the heuristics of local shape comparison can be used for any kind of 3D object, but these objects must be first superposed according to certain pairs of points and the region of interest must be defined which makes the heuristics reusable in only specific cases. This heuristics relies on the calculation of a volume which is much more generic and could be used in a larger number of situations.

5.1.3.2 Languages

Specialized languages Most of the languages that were created for SuMo are not supposed to be reused. Their purpose is just to make easier and more natural some commands. For example, the language for defining chemical groups has a syntax which is totally dedicated to the definition of types of chemical groups so that the files are easy to read and to edit.

Generic languages Two generic languages have been developed and used in SuMo:

- the syntaxic extension Printfer makes it easy to produce HTML that contains a lot of dynamic data,
- the generic syntax for writing trees with typed leaves that was used for SuMoQ and for rewriting structure files.

These both tools are fully reusable in programs or systems that have nothing to do with biology or chemistry. They are also both only syntaxes that can be used over data that have their own semantics such as HTML or SuMoQ queries.

5.1.4 Future developments

The future development of the method and the software require a good knowledge of the different approaches that are used. While developing or modifying any piece of code, all these properties must be known:

- need for exact algorithms?
- cost of the algorithms
- relative cost compared to other computations that are performed in the same context
- how much is displayed
- memory usage
- disk usage
- delay for retrieving the data
- need for reusability?
- length of the code
- security level

Various developments of the method and of its implementation can be performed. These developments will be triggered by the demand of users and depending on the problems that are encountered. However, the core design of SuMo brings some constraints and problems that would be much easier solved with complementary approaches rather than by trying to make deep changes in SuMo.

5.2 Current limits of the system

The needs of SuMo in computational resources are moderate. A recent PC is enough for the most common tasks. Table 5.1 page 136 gives rough estimations of duration.

Type of comparisons	Duration
1 structure / 11,000 sites	5–30 min
1 site / 20,000 structures	30–90 min
11,000 sites / 11,000 sites	8 days

Table 5.1: Rough estimations of the length of the comparisons. Processor Intel Pentium III, 2 GHz.

The current implementation of SuMo does not require a large number of computers neither a huge storage space for someone who is interested in a specific site or a specific protein. More massive comparisons would however require more computers. It could be interesting to perform in advance the preditions of ligand binding sites for all the available 3D structures of proteins and store the results. Then these results would be accessible to any user without delay. If 20 min are required for each protein, 278 days would be required to precompute the results over the 20,000 structures of the PDB. If 10 CPUs were used at the same time, this could be completed in one month.

Thus the computation time is not a real limit. It is however important to have a good view over the limits of the quality of the results. SuMo is based on the following points:

1. representation of a 3D structure by a discrete set of objects, each of them representing a localized property,
2. negligible uncertainties on the positions of the objects (chemical groups) compared to the distances between these positions.

The first point imposes a constraint that we will call the *localization problem* while the latter imposes a constraint that we will call the *scale problem.*

5.2.1 The localization problem

3D structures of molecules are represented in SuMo by chemical groups. These chemical groups have a specific location in space and predefined shape and size. Only the chemical groups of the same type can be compared and possibly considered as equivalent.

However, some properties that are intuitively interesting such as hydrophobicity are not taken into account by SuMo. The arbitrary size and shape of regions that could be considered as hydrophobic makes it impossible to model them as SuMo chemical groups.

The solution that allows partially to work around this limitation is the shape comparison of the environment (see heuristics page 79). Currently no special property is projected on the atoms that are used to define the environment of a chemical group but such an extension is possible. It would require to assign to each atom of the environment a weight or characteristic mass (see page 84) which would be proportional to the property being mapped.

5.2.2 The scale problem

The scale problem is related to the variable uncertainty on the location of chemical groups of different types. The larger is the region that is modeled by a chemical group, the larger will be the uncertainty on the functional location. This is only a problem when the distances between chemical groups are in the same order of magnitude of this uncertainty.

It is however interesting to define types of chemical groups with different radii of influence. For instance:

	radius of influence	**absolute uncertainty**
H bond donor	small	small
phenyl	medium	medium

The notion of radius of influence is represented by the coefficient of the given type of chemical group, as defined page 48.

The scale problem occurs when chemical groups of different coefficients are grouped into triplets. A selection on the size of the edges of the physical triangles is performed while taking into account the coefficients (see page 56). The effect of this selection is to avoid to connect chemical groups that are too close or too distant according to their radii of influence. However, grouping in a same triplet chemical groups of very different radii of influence will lead to bad results since distances between chemical groups are compared. In this case, the results tend to be:

- too restrictive for large radii of influence,
- too permissive for small radii of influence.

The problem is that all the chemical groups must be connected together, either directly or not, so that they may all appear in the same list of matched elements.

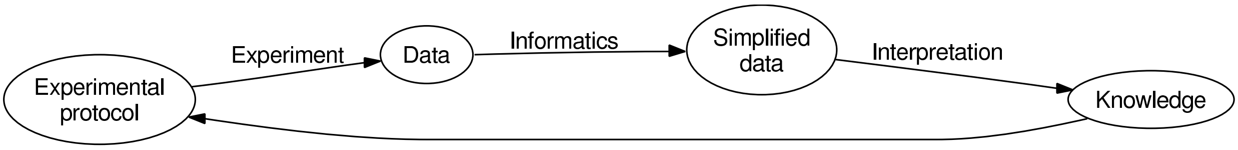


Figure 5.2: Informatics and experimental sciences

One possible solution is to accept in the same triplet only chemical groups that have more or less the same radii of influence or coefficients. For instance, if we consider chemical groups of types A, B and C and of radii a, b, and c so that a and c are incompatible, then the possible types of triplets would be restricted to the following list: (A, A, A), (A, A, B), (A, B, B), (B, B, B), (B, B, C), (B, C, C), (C, C, C), (B, B, C). If a given region of the molecule does not contain any chemical group of type B, then the groups of type A and C in this region will not be able to appear in the same matched sites.

5.3 Conclusion

Figure 5.2 page 138 is a diagram that shows the relationships between computer science and experimental sciences. Informatics applied to experimental sciences can convert experimental data which are difficult to read into a format which is easy to interpret for the researcher.

According to this diagram, SuMo processes 3D structures of proteins, which are difficult to analyze visually, into some result which is much more simple to analyze. We can consider that SuMo plays the role of an additional sensory faculty. It may see some information that the human cannot see with the usual representations of the 3D structures of molecules. This sensory faculty is purely objective since the results are perfectly reproducible. The final data are hence less complete than the initial data but allow much more sophisticated hypotheses to be made.

Chapter 6

Involvement in other projects

During my PhD, I was involved in other projects that lead to publications. I give here a short description of these projects.

6.1 Anti-apoptotic protein Nr-13

An analogy-based molecular modeling of protein Nr-13 was the carried out as an internship in Summer 1998. Nr-13 is an anti-apoptotic protein. After this study, several mutants have been made and tested by the team of Germain Gillet, IBCP. Protocols and results have been published [26].

6.2 Geno3D

The protocol of molecular modeling by analogy that was used for the prediction of the 3D structure of Nr-13 has been completely automated and made public through the Geno3D web server:

```
http://geno3d-pbil.ibcp.fr
```

This work was performed by Christophe Geourjon and was published [13]. Geno3D gets about 100 queries every day, and 15 to 20 of them successfully result in a 3D model.

Let us consider a protein sequence S for which we wish to predict the 3D structure. The principle of Geno3D is the following:

1. search for homologous 3D structures by comparison of amino acid sequences and prediction of secondary structures,

2. generation of geometric constraints on distances and angles that are supposed to be conserved between the known structures and S,
3. generation of a set of models using the geometric constraints and the simulation software CNS.

Chapter 7

Publications

7.1 Articles

- M. Jambon, A. Imberty, G. Deleage, and C. Geourjon. A new bioinformatic approach to detect common 3D sites in protein structures. *Proteins*, 52(2):137–45, 2003.

- C. Combet, M. Jambon, G. Deleage, and C. Geourjon. Geno3D: automatic comparative molecular modelling of protein. *Bioinformatics*, 18(1):213–4, 2002.

- P. Lalle, A. Aouacheria, A. Dumont-Miscopein, M. Jambon, S. Venet, H. Bobichon, P. Colas, G. Deleage, C. Geourjon, and G. Gillet. Evidence for crucial electrostatic interactions between Bcl-2 homology domains BH3 and BH4 in the anti-apoptotic Nr-13 protein. *Biochem J*, 368(Pt 1):213–21, 2002.

7.2 Patent

- M. Jambon, G. Deléage, and C. Geourjon. Process for identifying similar 3D substructures onto 3D structures and its applications. European patent EP1369807 2002-06-06, International patent WO03104388.

7.3 Oral communications

- M. Jambon. Communication (1 hour): Un système de prédiction de sites fonctionnels dans les structures 3d de protéines. Seminar for Hybrigenics, Paris, France, 12 June 2002.

- M. Jambon. Communication (20 min): Sumo, une méthode de comparaison des propriétés de surface des protéines. 5e journée de l'EDISS (Ecole doctorale), Lyon, France, 29 May 2001.
- M. Jambon. Communication (30 min): Sumo, une méthode de comparaison des propriétés de surface des protéines. Séminaire de ■ Bio-Informatique Structurale ■, Montpellier, France, 4 October 2001.
- M. Jambon, C. Combet, G. Deléage, and C. Geourjon. Communication (20 min): Sumo, une méthode de comparaison des propriétés de surface des protéines. 12e rencontres du GGMM (groupe de graphisme et modélisation moléculaire), Nîmes, France, 9-11 May 2001.
- M. Jambon, A. Imberty, G. Deléage, and C. Geourjon. Communication (20 min): Sumo: a software that detects 3d sites shared by protein structures. JOBIM 2002 (Journées ouvertes biologie informatique mathématiques), Saint Malo, France, 12 June 2002.

7.4 Posters

- C. Combet, M. Jambon, G. Deléage, and C. Geourjon. Poster: Geno3d un serveur web automatique de modélisation moléculaire. 12e rencontres du GGMM (groupe de graphisme et modélisation moléculaire), Nîmes, France, 9-11 May 2001.
- M. Jambon, G. Deléage, and C. Geourjon. Poster: Sumo : logiciel intégré de caractérisation de sites actifs de protéines au service du drug design. 9e carrefours de la fondation Rhône-Alpes Futur, Lyon, France, 17 November 2001.
- M. Jambon, M. Errami, Gilbert Deléage, and C. Geourjon. Poster: Development of the sumo method to detect 3d sites in proteins. 12e rencontres du GGMM (groupe de graphisme et modélisation moléculaire), Nîmes, France, 9-11 May 2001.
- M. Jambon, M. Errami, Gilbert Deléage, and C. Geourjon. Poster: Development of the sumo method to detect 3d sites in proteins. Summer school on protein folding, Cargèse, France, 20-26 May 2001.
- M. Jambon, M. Errami, Gilbert Deléage, and C. Geourjon. Poster: Development of the sumo method to detect 3d sites in proteins. 5e journée de l'EDISS (Ecole doctorale), Lyon, France, 29 May 2001.

- M. Jambon, M. Errami, Gilbert Deléage, and C. Geourjon. Poster: Development of the sumo method to detect 3d sites in proteins. JOBIM 2001 (Journées ouvertes biologie informatique mathématiques) Toulouse, France, 30 May-1 June 2001.

Appendix A

Definition of the chemical groups

```
(* $Id: amino_acids,v 1.32 2003/02/24 16:08:23 martin Exp $ *)

(* These are the current default definitions for chemical groups in
 * SuMo. The database must be recreated every time the definitions
 * are modified (addition/removal of groups, new identifiers,
 * new parameter, ...)
 *)

(* Recent changes:
   - version 4.4: many changes in parameters
   - version 4.3: new ‘target’ parameter for aromatic
   - version 4.2: important changes in the coefficients
*)

Export (acyl amide aromatic hydroxyl histidine guanidium
        thioether thiol
        delta_plus delta_minus
        (*proline glycine*)
        (*negative positive*));

Not_special ("ALA" "CYS" "ASP" "GLU" "PHE" "GLY" "HIS" "ILE" "LYS" "LEU"
             "MET" "ASN" "PRO" "GLN" "ARG" "SER" "THR" "VAL" "TRP" "TYR"
             "MSE");

Not_special ("HOH");


ala = <<"ALA">>;
arg = <<"ARG">>;
asn = <<"ASN">>;
asp = <<"ASP">>;
```

```
cys = <<"CYS">>;
gln = <<"GLN">>;
glu = <<"GLU">>;
gly = <<"GLY">>;
his = <<"HIS">>;
ile = <<"ILE">>;
leu = <<"LEU">>;
lys = <<"LYS">>;
met = <<"MET">>;
mse = <<"MSE">>;
phe = <<"PHE">>;
pro = <<"PRO">>;
ser = <<"SER">>;
thr = <<"THR">>;
trp = <<"TRP">>;
tyr = <<"TYR">>;
val = <<"VAL">>;

aa = <<"ALA" "CYS" "ASP" "GLU" "PHE" "GLY" "HIS" "ILE" "LYS" "LEU"
       "MET" "MSE" "ASN" "PRO" "GLN" "ARG" "SER" "THR" "VAL" "TRP" "TYR">>;
aaa = <<"ALA" "CYS" "ASP" "GLU" "PHE" "GLY" "HIS" "ILE" "LYS" "LEU"
        "MET" "MSE" "ASN" "GLN" "ARG" "SER" "THR" "VAL" "TRP" "TYR">>;


(***** H-bonding groups *****)
delta_plus {0.6} =

| Delta_plus [backbone] (aaa.<"N">,
                         aa[-1].<"C"> aaa.<"CA">,
                         aaa.<"N">,
                         target = 0.0,
                         functional_shift = 2.8, angle = 140.)

| Delta_plus_plan [d22]
                  (asn.<"ND">, (* position *)
                   asn.<"CG">, (* v_start *)
                   asn.<"ND">, (* v_stop *)
                   asn.<"CG">, (* plan1 *)
                   asn.<"ND">, (* plan2 *)
                   asn.<"OD">, (* plan3 *)
                   60,
                   target = 0.0,
                   functional_shift = 2.8, angle = 140.)

| Delta_plus_plan [d21]
                  (asn.<"ND">,
                   asn.<"CG">,
                   asn.<"ND">,
                   asn.<"CG">,
                 asn.<"ND">,
                 asn.<"OD">,
                 -60,
                 target = 0.0,
                 functional_shift = 2.8, angle = 140.)

| Delta_plus_plan [e22]
                (gln.<"NE">,
                 gln.<"CD">,
                 gln.<"NE">,
                 gln.<"CD">,
                 gln.<"NE">,
                 gln.<"OE">,
                 60,
                 target = 0.0,
                 functional_shift = 2.8, angle = 140.)

| Delta_plus_plan [e21]
                (gln.<"NE">,
                 gln.<"CD">,
                 gln.<"NE">,
                 gln.<"CD">,
                 gln.<"NE">,
                 gln.<"OE">,
                 -60,
                 target = 0.0,
                 functional_shift = 2.8, angle = 140.)

| Delta_plus [e]
           (arg.<"NE">,
            arg.<"CD"> arg.<"CZ">,
            arg.<"NE">,
            target = 0.0,
            functional_shift = 2.8, angle = 140.)

| Delta_plus_plan [h12]
                (arg.<"NH1">,
                 arg.<"CZ">,
                 arg.<"NH1">,
                 arg.<"CZ">,
                 arg.<"NH1">,
                 arg.<"NH2">,
                 60,
                 target = 0.0,
                 functional_shift = 2.8, angle = 140.)

| Delta_plus_plan [h11]
                (arg.<"NH1">,
                 arg.<"CZ">,
                  arg.<"NH1">,
                  arg.<"CZ">,
                  arg.<"NH1">,
                  arg.<"NH2">,
                  -60,
                  target = 0.0,
                  functional_shift = 2.8, angle = 140.)

| Delta_plus_plan [h21]
                 (arg.<"NH2">,
                  arg.<"CZ">,
                  arg.<"NH2">,
                  arg.<"CZ">,
                  arg.<"NH1">,
                  arg.<"NH2">,
                  60,
                  target = 0.0,
                  functional_shift = 2.8, angle = 140.)

| Delta_plus_plan [h22]
                 (arg.<"NH2">,
                  arg.<"CZ">,
                  arg.<"NH2">,
                  arg.<"CZ">,
                  arg.<"NH1">,
                  arg.<"NH2">,
                  -60,
                  target = 0.0,
                  functional_shift = 2.8, angle = 140.)

| Delta_plus (trp.<"NE1">,
             trp.<"CD1"> trp.<"CE2">,
             trp.<"NE1">,
             target = 0.0,
             functional_shift = 2.8, angle = 140.)

| Delta_plus [d1]
            (his.<"ND1">,
             his.<"CG"> his.<"CE1">,
             his.<"ND1">,
             target = 0.0,
             functional_shift = 2.8, angle = 140.)

| Delta_plus [e2]
            (his.<"NE2">,
             his.<"CD2"> his.<"CE1">,
             his.<"NE2">,
             target = 0.0,
             functional_shift = 2.8, angle = 140.)
```

```
| Delta_plus_multiple (lys.<"NZ">,
                       lys.<"CE">,
                       lys.<"NZ">,
                       3,
                       71,
                       target = 0.0,
                       functional_shift = 2.8, angle = 140.)

| Delta_plus_multiple (ser.<"OG">,
                       ser.<"CB">,
                       ser.<"OG">,
                       1,
                       71,
                       target = 0.0,
                       functional_shift = 2.8, angle = 140.)

| Delta_plus_multiple (thr.<"OG1">,
                       thr.<"CB">,
                       thr.<"OG1">,
                       1,
                       71,
                       target = 0.0,
                       functional_shift = 2.8, angle = 140.)

| Delta_plus_multiple (tyr.<"OH">,
                       tyr.<"CZ">,
                       tyr.<"OH">,
                       1,
                       71,
                       target = 0.0,
                       functional_shift = 2.8, angle = 140.)
;


delta_minus {0.6} =
| Delta_minus [backbone] (aa.<"O">,
                          aa.<"C">,
                          aa.<"O">,
                          1,
                          target = 1.)

| Delta_minus [d1] (asp.<"OD1">,
                    asp.<"CG">,
                    asp.<"OD1">,
                    1,
                    target = 1.)

| Delta_minus [d2] (asp.<"OD2">,
                    asp.<"CG">,
                    asp.<"OD2">,
                    1,
                    target = 1.)
| Delta_minus [e1] (glu.<"OE1">,
                    glu.<"CD">,
                    glu.<"OE1">,
                    1,
                    target = 1.)
| Delta_minus [e2] (glu.<"OE2">,
                    glu.<"CD">,
                    glu.<"OE2">,
                    1,
                    target = 1.)
| Delta_minus (asn.<"OD1">,
               asn.<"CG">,
               asn.<"OD1">,
               1,
               target = 1.)
| Delta_minus (gln.<"OE1">,
               gln.<"CD">,
               gln.<"OE1">,
               1,
               target = 1.)
| Delta_minus (ser.<"OG">,
               ser.<"CB">,
               ser.<"OG">,
               1,
               target = 1.)
| Delta_minus (thr.<"OG1">,
               thr.<"CB">,
               thr.<"OG1">,
               1,
               target = 1.)
| Delta_minus (tyr.<"OH">,
               tyr.<"CZ">,
               tyr.<"OH">,
               1,
               target = 1.)
;

Hbond (delta_plus, delta_minus);



(***** Other groups *****)

acyl {0.75} =
| Plan (asp.<"CG"> asp.<"OD1"> asp.<"OD2">,
        asp.<"CG">,
        asp.<"OD1"> asp.<"OD2">,
        asp.<"OD1">,
        angle1 = 60,
        angle2 = 90,
        target = 1.)
| Plan (glu.<"CD"> glu.<"OE1"> glu.<"OE2">,
        glu.<"CD">,
        glu.<"OE1"> glu.<"OE2">,
        glu.<"OE1">,
        angle1 = 60,
        angle2 = 90,
        target = 1.)
;

amide {0.75} =
| Chiral (asn.<"CG"> asn.<"OD1"> asn.<"ND2">,
          asn.<"CG">,
          asn.<"OD1"> asn.<"ND2">,
          asn.<"OD1">,
          angle1 = 60,
          angle2 = 90,
          target = 1.)
| Chiral (gln.<"CD"> gln.<"OE1"> gln.<"NE2">,
          gln.<"CD">,
          gln.<"OE1"> gln.<"NE2">,
          gln.<"OE1">,
          angle1 = 60,
          angle2 = 90,
          target = 1.)
;

positive {0.75} =
| Point (arg.<"NH1"> arg.<"NH2">)
| Point (lys.<"NZ">)
| Point (his.<"NE2">)
;

negative {0.75} =
| Point (asp.<"OD1"> asp.<"OD2">)
| Point (glu.<"OG1"> glu.<"OG2">)
;

aromatic {0.9} =
| Biplan (phe.<"CG"> phe.<"CD1"> phe.<"CD2">
           phe.<"CE1"> phe.<"CE2"> phe.<"CZ">,
          phe.<"CG">,
          phe.<"CE1">,
          phe.<"CE2">,
          angle = 45, (* was 60 *)
          target = 4.5)

| Biplan (tyr.<"CG"> tyr.<"CD1"> tyr.<"CD2">
           tyr.<"CE1"> tyr.<"CE2"> tyr.<"CZ">,
          tyr.<"CG">,
          tyr.<"CE1">,
          tyr.<"CE2">,
          angle = 45,
          target = 4.5)

| Biplan (his.<"CG"> his.<"ND1"> his.<"CE1"> his.<"CD2"> his.<"NE2">,
          his.<"CG">,
          his.<"ND1">,
          his.<"NE2">,
          angle = 45,
          target = 4.5)

| Biplan [penta]
         (trp.<"CG"> trp.<"CD1"> trp.<"NE1">
           trp.<"CD2"> trp.<"CE2">,
          trp.<"CG">,
          trp.<"CD1">,
          trp.<"CE2">,
          angle = 45,
          target = 4.5)

| Biplan [hexa]
         (trp.<"CD2"> trp.<"CE2"> trp.<"CZ2"> trp.<"CH2">
           trp.<"CE3"> trp.<"CZ3">,
          trp.<"CD2">,
          trp.<"CZ2">,
          trp.<"CZ3">,
          angle = 45,
          target = 4.5)
;

hydroxyl {0.65} =
| Polar (ser.<"OG">, ser.<"CB">, ser.<"OG">, angle = 120, target = 0.)
| Polar (thr.<"OG1">, thr.<"CB">, thr.<"OG1">, angle = 120, target = 0.)
| Polar (tyr.<"OH">, tyr.<"CZ">, tyr.<"OH">, angle = 120, target = 0.)
;

thiol {0.65} = Polar (cys.<"SG">,
               cys.<"CB">,
               cys.<"SG">,
               angle = 120,
               target = 0.)
;
```

```
histidine {0.9} = Chiral (his.<"CG"> his.<"ND1"> his.<"CE1"> his.<"CD2"> his.<"NE2">,
                his.<"CG">,
                his.<"CE1"> his.<"NE2">,
                his.<"CD2">,
                angle1 = 90,
                angle2 = 90,
                target = 0.)
;

proline {0.75} = Point (pro.<"CA"> pro.<"CB"> pro.<"CD"> pro.<"N">);

glycine {0.75} = Polar (gly.<"N">,
                 aa[-1].<"C"> gly.<"CA">,
                 gly.<"N">,
                 angle = 60,
                 target = 0.)
;

guanidium {0.85} = Biplan (arg.<"CZ">,
                    arg.<"NE">,
                    arg.<"NH1">,
                    arg.<"NH2">,
                    angle = 45,
                    target = 2.)
;

thioether {0.65} = Point (met.<"SD">)
                | Point (mse.<"SE">); (* selenium from seleno-methionine *)
```

Appendix B

SuMo server's help

(paper version only)

Appendix C

Copy of the publications

(paper version only)

Bibliography

[1] P.J. Artymiuk, A.R. Poirrette, H.M. Grindley, D.W. Rice, and P. Willett. A graph-theoretic approach to the identification of three-dimensional patterns of amino acid side-chains in protein structures. *J Mol Biol*, 243(2):327–44, 1994.

[2] D.J. Bacon and W.F. Anderson. A fast algorithm for rendering space-filling molecule pictures. *Journal of Molecular Graphics*, 6:219–220, 1988.

[3] D. Bagley. The great computer language shootout. `http://www.bagley.org/~doug/shootout`

[4] A. Bairoch. Prosite: a dictionary of sites and patterns in proteins. *Nucleic Acids Res*, 19 Suppl:2241–5, 1991. 0305-1048 Journal Article.

[5] W.C. Barker, J.S. Garavelli, H. Huang, P.B. McGarvey, B.C. Orcutt, G.Y. Srinivasarao, C. Xiao, L.S. Yeh, R.S. Ledley, J.F. Janda, F. Pfeiffer, H.W. Mewes, A. Tsugita, and C. Wu. The protein information resource (PIR). *Nucleic Acids Res*, 28(1):41–4, 2000.

[6] W.C. Barker, J.S. Garavelli, P.B. McGarvey, C.R. Marzec, B.C. Orcutt, G.Y. Srinivasarao, L.S. Yeh, R.S. Ledley, H.W. Mewes, F. Pfeiffer, A. Tsugita, and C. Wu. The PIR-International Protein Sequence Database. *Nucleic Acids Res*, 27(1):39–43, 1999.

[7] A. Bateman, E. Birney, L. Cerruti, R. Durbin, L. Etwiller, S.R. Eddy, S. Griffiths-Jones, K.L. Howe, M. Marshall, and E.L. Sonnhammer. The Pfam protein families database. *Nucleic Acids Res*, 30(1):276–80, 2002.

[8] H.M. Berman, T. Battistuz, T.N. Bhat, W.F. Bluhm, P.E. Bourne, K. Burkhardt, Z. Feng, G.L. Gilliland, L. Iype, S. Jain, P. Fagan, J. Marvin, D. Padilla, V. Ravichandran, B. Schneider, N. Thanki, H. Weissig, J.D. Westbrook, and C. Zardecki. The Protein Data Bank. *Acta Crystallogr D Biol Crystallogr*, 58(Pt 6 No 1):899–907, 2002.

[9] C. Blanchet. *Logiciel MPSA et ressources bioinformatiques client-serveur Web dédiés à l'analyse de séquences de protéine.* PhD thesis, Université Claude Bernard, Lyon 1, 1999.

[10] B. Boeckmann, A. Bairoch, R. Apweiler, M.C. Blatter, A. Estreicher, E. Gasteiger, M.J. Martin, K. Michoud, C. O'Donovan, I. Phan, S. Pilbout, and M. Schneider. The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003. *Nucleic Acids Res*, 31(1):365–70, 2003.

[11] E. Chailloux, P. Manoury, and B. Pagano. *Développement d'applications avec Objective Caml.* O'Reilly, 2000.

[12] C. Combet, C. Blanchet, C. Geourjon, and G. Deléage. Nps@: Network protein sequence analysis. *TIBS*, 25:147–150, 2000.

[13] C. Combet, M. Jambon, G. Deleage, and C. Geourjon. Geno3D: automatic comparative molecular modelling of protein. *Bioinformatics*, 18(1):213–4, 2002.

[14] F. Corpet. Multiple sequence alignment with hierarchical clustering. *Nucleic Acids Res*, 16(22):10881–90, 1988.

[15] D. de Rauglaudre. Camlp4: Pre-processor-pretty-printer for objective caml. `http://caml.inria.fr/camlp4/`

[16] D. de Rauglaudre. Ioxml. `http://cristal.inria.fr/~ddr/IoXML`

[17] Altschul S. F., Madden T. L., Schaffer A. A., Zhang J., Zhang Z., Miller W., and Lipman D. J. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic Acids Res*, 25(17):3389–402, 1997. 0305-1048 Journal Article Review Review, Tutorial.

[18] D. Fischer, O. Bachar, R. Nussinov, and H. Wolfson. An efficient automated computer vision based technique for detection of three dimensional structural motifs in proteins. *J Biomol Struct Dyn*, 9(4):769–89, 1992. 0739-1102 Journal Article.

[19] M. Gribskov, A. D. McLachlan, and D. Eisenberg. Profile analysis: detection of distantly related proteins. *Proc Natl Acad Sci U S A*, 84(13):4355–8, 1987. 0027-8424 Journal Article.

[20] L. Holm and C. Sander. The fssp database: fold classification based on structure-structure alignment of proteins. *Nucleic Acids Res*, 24(1):206–9, 1996. 0305-1048 Journal Article.

[21] L. Holm and C. Sander. Touring protein fold space with dali/fssp. *Nucleic Acids Res*, 26(1):316–9, 1998. 0305-1048 Journal Article.

[22] M. Jambon, G. Deléage, and C. Geourjon. Process for identifying similar 3D substructures onto 3D structures and its applications. Dépôt par le CNRS d'une demande de brevet européen numéro 02291407.1 en date du 6 juin 2002.

[23] M. Jambon, A. Imberty, G. Deleage, and C. Geourjon. A new bioinformatic approach to detect common 3D sites in protein structures. *Proteins*, 52(2):137–45, 2003.

[24] G.J. Kleywegt. Recognition of spatial motifs in protein structures. *J Mol Biol*, 285(4):1887–97, 1999.

[25] P.J. Kraulis. Molscript: A program to produce both detailed and schematic plots of protein structures. *Journal of Applied Crystallography*, 24:946–50, 1991.

[26] P. Lalle, A. Aouacheria, A. Dumont-Miscopein, M. Jambon, S. Venet, H. Bobichon, P. Colas, G. Deleage, C. Geourjon, and G. Gillet. Evidence for crucial electrostatic interactions between Bcl-2 homology domains BH3 and BH4 in the anti-apoptotic Nr-13 protein. *Biochem J*, 368(Pt 1):213–21, 2002.

[27] R.A. Laskowski. PDBsum: summaries and analyses of PDB structures. *Nucleic Acids Res*, 29(1):221–2, 2001.

[28] R.A. Laskowski, E.G. Hutchinson, A.D. Michie, A.C. Wallace, M.L. Jones, and J.M. Thornton. PDBsum: a Web-based database of summaries and analyses of all PDB structures. *Trends Biochem Sci*, 22(12):488–90, 1997.

[29] X. Leroy, J. Vouillon, D. Doligez, and coworkers. The objective caml system. software and documentation on the web, `http://caml.inria.fr/ocaml/`, 1996.

[30] S. L. Lin, R. Nussinov, D. Fischer, and H. J. Wolfson. Molecular surface representations by sparse critical points. *Proteins*, 18(1):94–101, 1994. 0887-3585 Journal Article.

[31] H. Lis and N. Sharon. Lectins: carbohydrate-specific proteins that mediate cellular recognition. *Chem. Rev.*, 98:637–674, 1998.

[32] R. Loris, T. Hamelryck, J. Bouckaert, and L. Wyns. Legume lectin structure. *Biochim. Biophys. Acta*, pages 9–36, 1998.

[33] E.A. Merritt and D.J. Bacon. Raster3d: Photorealistic molecular graphics. *Methods in Enzymology*, 277:505–24, 1997.

[34] E.A. Merritt and M.E.P. Murphy. Raster3d version 2.0: A program for photorealistic molecular graphics. *Acta Cryst.*, D50:869–73, 1994.

[35] M. Mottl. Ocamlmakefile: Automated compilation of complex ocaml-projects. `http://www.ai.univie.ac.at/~markus/home/ocaml_sources.html`

[36] C. Notredame, D.G. Higgins, and J. Heringa. T-Coffee: A novel method for fast and accurate multiple sequence alignment. *J Mol Biol*, 302(1):205–17, 2000.

[37] W. R. Pearson. Rapid and sensitive sequence comparison with fastp and fasta. *Methods Enzymol*, 183:63–98, 1990. 0076-6879 Journal Article.

[38] R. Preissner, A. Goede, and C. Frommel. Dictionary of interfaces in proteins (dip). data bank of complementary molecular surface patches. *J Mol Biol*, 280(3):535–50, 1998. 0022-2836 Journal Article.

[39] R. Preissner, A. Goede, and C. Frommel. Homonyms and synonyms in the dictionary of interfaces in proteins (dip). *Bioinformatics*, 15(10):832–6, 1999. 1367-4803 Journal Article.

[40] R. B. Russell. Detection of protein three-dimensional side-chain patterns: new examples of convergent evolution. *J Mol Biol*, 279(5):1211–27, 1998. 0022-2836 Journal Article.

[41] B. Sandak, R. Nussinov, and H. J. Wolfson. An automated computer vision and robotics-based technique for 3-d flexible biomolecular docking and matching. *Comput Appl Biosci*, 11(1):87–99, 1995. 0266-7061 Journal Article.

[42] I.N. Shindyalov and P.E. Bourne. Protein structure alignment by incremental combinatorial extension (CE) of the optimal path. *Protein Eng*, 11(9):739–47, 1998.

[43] T.F. Smith and M.S. Waterman. Identification of common molecular subsequences. *J Mol Biol*, 147(1):195–7, 1981.

[44] A. Stark, S. Sunyaev, and R.B. Russell. A model for statistical significance of local similarities in structure. *J Mol Biol*, 326(5):1307–16, 2003.

[45] G. Stolpmann. Ocamlnet. `http://sourceforge.net/projects/ocamlnet`

[46] Thomas Cormen, Charles Leiserson, and Ronald Rivest. *Introduction à l'algorithmique*. Dunod, 1994.

[47] J. D. Thompson, D. G. Higgins, and T. J. Gibson. Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res*, 22(22):4673–80, 1994. 0305-1048 Journal Article.

[48] A. Via, F. Ferre, B. Brannetti, and M. Helmer-Citterich. Protein surface similarities: a survey of methods to describe and compare protein surfaces. *Cell Mol Life Sci*, 57(13-14):1970–7, 2000. 1420-682x Journal Article Review Review, Tutorial.

[49] D. Voet and J. Voet. *Biochemistry*. Wiley and Sons, 1997. pages 389-400.

[50] A. C. Wallace, N. Borkakoti, and J. M. Thornton. Tess: a geometric hashing algorithm for deriving 3d coordinate templates for searching structural databases. application to enzyme active sites. *Protein Sci*, 6(11):2308–23, 1997. 0961-8368 Journal Article.

[51] A. C. Wallace, R. A. Laskowski, and J. M. Thornton. Derivation of 3d coordinate templates for searching structural databases: application to ser-his-asp catalytic triads in the serine proteinases and lipases. *Protein Sci*, 5(6):1001–13, 1996. 0961-8368 Journal Article.

[52] A. C. Wallace and J. M. Thornton. Procat, a database of 3d enzyme active site templates `http://biochem.ucl.ac.uk/bsm/procat/procat.html`, 1996.

[53] P. Weis and X. Leroy. *Le langage Caml*. Dunod, 1999.

[54] Xavier Leroy, Damien Doligez, Jacques Garrigue, Didier Rémy, and Jérôme Vouillon. *The Objective Caml system*. Institut National de Recherche en Informatique et en Automatique, release 3.06 edition.

[55] E.M. Zdobnov and R. Apweiler. InterProScan–an integration platform for the signature-recognition methods in InterPro. *Bioinformatics*, 17(9):847–8, 2001.

Index

ab initio, 25
algorithm, 77
α-amylases, 123
amide, 52
anchor, 66
annotation of chemical groups, 52
annotations of sets of chemical groups, 111
anti-apoptotic, 139
apparent specificity, 75
arbitrary annotations, 53
arcelin, 123
aromatic, 121
aspartate, 53
atomic density, 78–79
atomic environment, 49
ATP, 103, 126

biological problem, 77
boolean operators, 103
`build-sumo-db`, 41
burial of a chemical group, 49

C, 30, 115, 134
C++, 30, 122
calcium, 103
Caml, 114, 116, 122, 134
Camlp4, 30, 34, 110, 114, 115
carboxylate, 53
CERMAV, 123
CGI, 31, 95, 96, 114, 115, 117
chain, 52
characteristic distance
  in deformation, 88
characteristic mass, 84, 137
characteristic site, 69
chemical group, 44–55
chemical groups, 44
  defining, 53–55
clique, 92
f-clique, 93
f-cliques
  examples, 94
ClustalW, 26
cluster
  of PCs, 120
CNRS, 133
CNS, 140
`cns2pdb`, 41
coefficient
  of importance, 87
coefficient of importance, 87
comparison
  multiple, 68
competition, 23
compilation
  utilities, 41
compression, 61, 71
`configure`, 41
conformation, 23
constructors
  types of chemical groups, 51
CPU, 136
CPU time, 61
Cryptgps, 110, 114
cryptographic signature, 110
crystallography, 23

cutoff, 88
CVS, 41
Cygwin, 133

databases, 70–74
deformation, 66, **92**, 85–92
  parameters, 87
density, 80
Density, 78
developer, 34
deviation between 2 pairs of points, **87**
Digest, 110
distance between 2 pairs of points, **87**
distance of reference, 87
  in deformation, 88
distribution of the software, 133
docking, 132
docking, 25
double neighboring, 64
drug design, 132

email, 105
external software
  web server, 114

family of ligand binding sites, 74
family of ligands, 74
FAQ, 120
finite pseudo-solid, 89
flexibility, 120
flexible ligand, 66
format
  PDB, 41, 44, 52, 101
function of a protein, 23
function of influence, 83
functional flexibility, 66, 86
functional location, 49
functional sites, 23
functional triangles, 56

Geno3D, 139–140
geometric constraints, 140
geometric variant, 50
  examples, 50
GNU, 114, 133
GNU/Linux, *see* Linux
graph
  comparison, 64
Graphviz, 35
guanidinium, 121
`gzip`, 62

Hashtbl, 95
help
  interactive, 113
  online, 112
heuristic, 18
heuristics, 77
HIC-Up, 126
HTML, 31, 95, 109, 111, 116, 117, 135
  automated generation, 115
HTTP, 31, 95, 133, 134
hydrogen, 55
hydrogen bond
  detection, 46–47
  model, 46

imidazole, 121
independent subgraphs, 45
industrialization, 122
INRIA, 30
interactive query, 103
interfaces, 95
Internet, 19
intersection of annotated region, 111
IoXML, 110, 114
IP addresses, 133
isometric tranformation, 90

job queue, 119
jobqueue

  figure, 118
`jobqueue`, 40, 41, 117

label
  of chemical group, 53
labels, 53
lactose, 123
languages
  specialized, 134
layers
  SuMo, 34
Lectins, 123
Lex, 30
libraries
  web server, 114
ligand, 23, 47
  nomenclature, 48
linear system, 84
Linux, 133
list of matched elements, 85
list of matched pairs, 43
local center of mass, 49
local deformation, 90
local density, 78
localization problem, 136

MacOS, 30, 134
magnesium, 103
main chain, 53
Makefile, 114
`Makefile`, 41
Marshal, 30, 61, 62, 109, 119
maximal annotation, 111
maximal stable f-clique, 93
mean
  logarithmic, 126
  of fractions, 92
MEDIT, 133
meta-server, 39
Mg^{2+}, *see* magnesium
Microsoft Windows, 133, 134
modules
  Caml, 34, 35
molecular dynamics, 122
molecular modeling, 139
molecule, 53
molecules
  identification, 44
MolScript, 109, 114, 134
monosaccharide, 126
Multalin, 26
multi-processor, 119
multimer, 71

NFS, 62
niveaux
  SuMo, 34
NMR, 23
NP-complete, 69, 93
Nr-13, 139
Nucleotides, 48

Objective Caml, 30, 34, 61, 95, 109, 110, 114, 119, 134
OCaml, *see* Objective Caml
Ocamldot, 35
Ocamllex, 30, 34
OcamlMakefile, 114
Ocamlnet, 114
Ocamlyacc, 30, 34
oligosaccharides, 48, 123
open source, 134
operating system, 133
overlap
  between sites, 69
overlap factor, 69

parameter, 121
passwords, 133
patent, 133
PBS, 120
PDB, 24, 70, 74, 130, 136
PDB group, 52

`pdb2tree`, 41
`pdb_groups`, 41
PDBsum, 126
permutations
  triplets, 58, 59, 63, 64
phantom chemical groups, 55
physical location, 49
physical triangles, 56
PINTS, 28
point object, 86
point of view
  administrator, 40
  programmer, 42
  user, 39
polymorphic, 134
POSIX, 133
predicate, 101
prediction, 74
`printf`, 115, 116
Printfer, 41, 114–117, 135
  syntax, 116
priorities, *see* `jobqueue`
problem of the localization, 136
PROCAT, 28
process
  `jobqueue`, 117
productivity, 30
profile analysis, 26
Proscan, 26
Prosite, 26
pseudo-solid, 89
pseudo-solids
  definitions, 89

quality, 77
quasi-adjacence, 60
quasi-adjacent, 60

radius of influence, 48, 137
Raster3D, 109, 115, 134
redundant chains, 72
regular expression, 26
relative deformation, 63
  generalized, 88
relevance of the results, 122
RMSD, 66, 85

scalar triple product, 59
scale problem, 136, 137
segments
  deformation, 86
selection
  chemical groups, 101–103
selectivity, 75
sensory faculty, 138
`seqinfo`, 41
shape
  comparison, 79
shape comparison, 134
size of the software, 34
solid angles
  deformation, 86
source files, 41
specificity, 126
spreadsheet, 110
stable f-clique, 93
standard vectors, 58
statistical validation, 121
stereoisomers, 86
structural genomics, 19
`substitute`, 41
SuMo, 18
  architecture, 34
  comparisons, 43
  language, 36, 96–99
`sumo`, 34, 36, 37, 41, 96, 112
`sumo-check`, 41
`sumo-clean`, 41
`sumo-columns`, 41
`sumo-database`, 41
`sumo-extend`, 41
`sumo-focus`, 41

`sumo-help`, 41
`sumo-results`, 41
`sumo-run`, 37, 41
`sumo-select`, 41
`sumo-sign`, 41
`sumo-sort`, 41
`sumo-welcome`, 41
SuMoQ, 37, 104
  description, 104–109
  higher level, 37–38
  specification, 108
  syntax, 105
Swiss-Prot, 24
symmetric pseudo-solid, 90
symmetric variants, 90
syntax
  Printfer, 116

target locations, 49
target structures, 71
tetrahedrons
  deformation, 86
triplets, 55–59
  selection, 56
type constructors, 51
type of a triplet, 58
types of chemical groups, 48

uncertainty, 136, 137
Unix, 30, 133, 134

`VERSION`, 41

web interface, 133
well-known monomer, 47
Windows, 30, *see* Microsoft Windows

XML, 39, 44, 105, 106, 110, 114

Yacc, 30