Computational approaches toward protein design

Seydou Traore

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

l'Institut National des Sciences Appliquées de Toulouse (INSA de Toulouse)

Présentée et soutenue le *jeudi 23/10/2014* **par :**

SEYDOU TRAORÉ

Computational approaches toward Protein Design

JURY

CATHERINE ETCHEBEST	Professeur Université Diderot-Paris 7	Rapportrice
RAPHAËL GUEROIS	Directeur de Recherche CEA	Rapporteur
JUAN CORTÉS	Chargé de Recherche CNRS	Membre invité
MAGALI REMAUD-SIMEON	Professeur INSA	Examinatrice
THOMAS SCHIEX	Directeur de Recherche INRA	Membre invité
THOMAS SIMONSON	Directeur de Recherche CNRS	Examinateur
ISABELLE ANDRÉ	Directrice de Recherche CNRS	Co-directrice de Thèse
SOPHIE BARBE	Chargée de Recherche INRA	Co-directrice de Thèse

---

École doctorale et spécialité :

SEVAB : Ingénieries microbienne et enzymatique

Unité de Recherche :

Laboratoire d'Ingénierie des Systèmes Biologiques et des Procédés
INSA, UMR INRA 792/CNRS 5504, F-31400 Toulouse, France

Directeur(s) de Thèse :

Isabelle ANDRÉ et *Sophie* BARBE

Rapporteurs :

Catherine ETCHEBEST et *Raphaël* GUEROIS

Laboratoire d'Ingénierie des Systèmes Biologiques et des Procédés
INSA, UMR INRA 792/CNRS 5504, F-31400 Toulouse, France

Computational approaches toward Protein Design

Seydou TRAORÉ

2011-2014

PhD Supervisors: Isabelle ANDRÉ & Sophie BARBE
PhD School: Sciences Ecologiques, Vétérinaires, Agronomiques et Bioingénieries

A mes parents

Name: Seydou TRAORÉ

Thesis Title: Computational approaches toward Protein Design

Supervisors: Isabelle André & Sophie Barbe

SUMMARY:

Computational Protein Design (CPD) is a very young research field which aims at providing predictive tools to complement protein engineering. Indeed, in addition to the theoretical understanding of fundamental properties and function of proteins, protein engineering has important applications in a broad range of fields, including biomedical applications, biotechnology, nanobiotechnology and the design of green reagents. CPD seeks at accelerating the design of proteins with wanted properties by enabling the exploration of larger sequence space while limiting the financial and human costs at experimental level.

To succeed this endeavor, CPD requires three ingredients to be appropriately conceived: 1) a realistic modeling of the design system; 2) an accurate definition of objective functions for the target biochemical function or physico-chemical property; 3) and finally an efficient optimization framework to handle large combinatorial sizes.

In this thesis, we addressed CPD problems with a special focus on combinatorial optimization. In a first series of studies, we applied for the first time the Cost Function Network optimization framework to solve CPD problems and found that in comparison to other existing methods, it brings several orders of magnitude speedup on a wide range of real CPD instances that include the stability design of proteins, protein-protein and protein-ligand complexes. A tailored criterion to define the mutation space of residues was also introduced in order to constrain output sequences to those expected by natural evolution through the integration of some structural properties of amino acids in the protein environment. The developed methods were finally integrated into a CPD-dedicated software in order to facilitate its accessibility to the scientific community.

Keywords: Computational Protein Design, Structural Bioinformatics, Combinatorial Optimization

---

Laboratory: « Laboratoire d'Ingénierie des Systèmes Biologiques et des Procédés »
LISBP, INSA, CNRS, INRA - Toulouse

PhD School: "Sciences Ecologiques, Vétérinaires, Agronomiques et Bioingénieries" (SEVAB)

Nom : Seydou TRAORÉ

Titre de la thèse : Approches computationnelles pour le Design de Protéines

Supervisors : Isabelle André & Sophie Barbe

RESUME :

Le Design computationnel de protéines, en anglais « Computational Protein Design » (CPD), est un champ de recherche récent qui vise à fournir des outils de prédiction pour compléter l'ingénierie des protéines. En effet, outre la compréhension théorique des propriétés physico-chimiques fondamentales et fonctionnelles des protéines, l'ingénierie des protéines a d'importantes applications dans un large éventail de domaines, y compris dans la biomédecine, la biotechnologie, la nanobiotechnologie et la conception de composés respectueux de l'environnement. Le CPD cherche ainsi à accélérer le design de protéines dotées des propriétés désirées en permettant le traitement d'espaces de séquences de large taille tout en limitant les coûts financier et humain au niveau expérimental.

Pour atteindre cet objectif, le CPD requière trois ingrédients conçus de manière appropriée: 1) une modélisation réaliste du système à remodeler; 2) une définition précise des fonctions objectives permettant de caractériser la fonction biochimique ou la propriété physico-chimique cible; 3) et enfin des méthodes d'optimisation efficaces pour gérer de grandes tailles de combinatoire.

Dans cette thèse, nous avons abordé le CPD avec une attention particulière portée sur l'optimisation combinatoire. Dans une première série d'études, nous avons appliqué pour la première fois les méthodes d'optimisation de réseaux de fonctions de coût à la résolution de problèmes de CPD. Nous avons constaté qu'en comparaison des autres méthodes existantes, nos approches apportent une accélération du temps de calcul par plusieurs ordres de grandeur sur un large éventail de cas réels de CPD comprenant le design de la stabilité de protéines ainsi que de complexes protéine-protéine et protéine-ligand. Un critère pour définir l'espace de mutations des résidus a également été introduit afin de biaiser les séquences vers celles attendues par une évolution naturelle en prenant en compte des propriétés structurales des acides aminés. Les méthodes développées ont été intégrées dans un logiciel dédié au CPD afin de les rendre plus facilement accessibles à la communauté scientifique.

Mots clés:	Ingénierie computationnelle de protéines, Bioinformatique structurale, Optimisation Combinatoire
Laboratoire:	Laboratoire d'Ingénierie des Systèmes Biologiques et des Procédés LISBP, INSA, CNRS, INRA - Toulouse
Ecole doctorale:	Sciences Ecologiques, Vétérinaires, Agronomiques et Bioingénieries (SEVAB)

Scientific contributions

PUBLICATIONS

[1] *D. Allouche, J. Davies, S. de Givry, G. Katsirelos, T. Schiex,* ***S. Traoré****, I. André, S. Barbe, S. Prestwich, B. O'Sullivan*. Computational Protein Design as an Optimization Problem. Artificial Intelligence Journal, March 2014.

[2] ***Traoré S****, Allouche D, André I, de Givry S, Katsirelos G, Schiex T and Barbe S*. A new framework for computational protein design through Cost function optimization. Bioinformatics, 2013.

[3] ***Traoré S****, Allouche D, André I, de Givry S, Katsirelos G, Barbe S and Schiex T*. Computational Protein Design as a Cost Function Network Optimization Problem. CP2012 Proceedings. 18th International Conference on Principles and Practice of Constraint Programming, Québec, Canada Oct, 8-12, 2012– Proceeding.

ORAL AND POSTER COMMUNICATIONS

- **Traoré S,** Allouche D, André I, de Givry S, Katsirelos G, Schiex T and Barbe S. *A New Framework for Computational Protein Design through Cost Function Network Optimization.* JOBIM 2013, Toulouse, July 1-4, 2013.
- **Traoré S,** Allouche D, André I, de Givry S, Katsirelos G, Schiex T and Barbe S. *Cost Function Network Optimization based Framework for Computational Protein Design.* GGMM 2013, Saint-Pierre d'Oléron, May 12-23, 2013.
- **Traoré S**, Topham C, Barbe S, André I. *The contribution of structural computational biology to the study and the design of enzymes.* GenoToul2012, Toulouse, Nov. 23, 2012.
- **S Traoré,** D Allouche, I André, S de Givry, G Katsirelos, S Barbe, and T Schiex. *Computational Protein Design by weighted constraint satisfaction.* Journée de l'Ecole Doctorale SEVAB, Toulouse, Nov. 6, 2012.
- Allouche D, **Traoré S,** André I, de Givry S, Katsirelos G, Barbe S, Schiex T. *Computational Protein Design as a Cost Function Network Optimization Problem.* 18th International Conference on Principles and Practice of Constraint Programming, Québec, *Canada,* Oct 8-12, 2012 (CORE: A).
- Allouche D, **Traoré S,** André I, de Givry S, Katsirelos G, Barbe S, Schiex T. *Cost function network and Quadratic programming: Connections & preliminary results on Computational Protein Design.* Workshop on Nonlinear Optimization and Constraint Programming, Québec, *Canada*, Oct 8, 2012.
- **Traoré S**, Allouche D, André I, de Givry S, Katsirelos G, Schiex T and Barbe S. *Computational Protein Design by weighted constraint satisfaction.* EMBO Conference, Catalytic mechanisms by Biological Systems, Groningen, *The Netherlands*, Oct 7-10, 2012**.**

Remerciements – Financements de thèse et formations complémentaires

Mes remerciements à:

L'Institut National de la Recherche Agronomique (INRA) & La Région Midi-Pyrénées

Pour le financement de la thèse conjointement par l'INRA et la Région Midi-Pyrénées.

L'Ecole Internationale de Recherche Agreenium (EIR-A)

Pour les formations doctorales complémentaires dont j'ai pu bénéficié. De même j'ai effectué un séjour scientifique de 3 mois au Etats-Unis, financé par l'INRA dans le cadre de l'EIR-A.

Contents

Abbreviations **xv**

Introduction **xvii**

1 Computational Protein Design (CPD): Paradigms, Methods and Challenges **1**
1.1 The CPD framework 5
1.2 Theoretical advances and applied interest of CPD 24
1.3 Concluding remarks, new trends and challenges ahead 29
1.4 References 32

2 Cost Function Network optimization **39**
2.1 Cost Function Networks 43
2.2 Local Consistencies 44
2.3 Weighted Constraint Satisfaction Problem solvers – Complete search 50
2.4 Alternative and complementary strategies 51
2.5 References 54

3 Cost Function Network-based Framework for CPD **55**
3.1 CPD as a Cost Function Network optimization problem 59
3.2 CPD as an optimization problem 69
3.3 A new framework for CPD through Cost Function Network optimization 105
3.4 Fast search algorithms for CPD 133
3.5 References 150

4 Concluding Remarks & Outlook **151**
4.1 Outline of the main results and their implications 153
4.2 Addressing the many energetically equivalent conformation issues 154
4.3 Improving the CPD input model(s) 155
4.4 Toward Multiple Objective Optimization & experimental applications 155
4.5 References 158

5 Résumé Long en Français **159**
5.1 Introduction 161
5.2 Stratégies de Design Computationnel de Protéines 162
5.3 De nouvelles approches basées sur les réseaux de fonctions de coût 166
5.4 Conclusion et perspectives 169
5.5 Références 171

Remerciements **173**

Abbreviations

AC	Arc Consistency
BE	Bucket Elimination
BQO	Boolean Quadratic Optimization
BS	Beam Search
CASA	Coulomb Accessible Surface Area
CDD	Cyclic Coordinate Descent
CFN	Cost Function Network
CP	Constraint Programming
CPD	Computational Protein Design
CSP	Constraint Satisfaction Problem
DAC	Directional Arc Consistency
DEE	Dead-End Elimination
DEEPer	Dead-End Elimination with Perturbations
DFBB	Depth-First Branch and Bound
DVO	Dynamic Variable Ordering
EA	Evolutionary Algorithms
EAC	Existential Arc Consistency
EPT	Equivalence Preserving Transformations
FDAC	Full Directional Arc Consistency
FDPB	Finite-Difference Poisson Boltzmann
GA	Genetic Algorithm
GB	Generalized Born implicit solvent
GMEC	Global Minimum Energy Conformation
ILP	Integer Linear Programing
KIC	Kinematic Closure
LC	Local Consistency properties
LDS	Limited Discrepancy Search
LK	Lazaridis-Karplus model
MA	Memetic Algorithm
MAP	Maximum A Posteriori
MB	Mini-Bucket
MC	Monte Carlo
MCSA	Monte Carlo Simulated Annealing
MD	Molecular Dynamics
MM	Molecular Mechanics
MOO	Multiple Objective Optimization
MPLP	Message Passing Linear Programming
MRF	Markov Random Field
MaxSat	Maximum Satisfiability problem
NC	Node Consistency
OSAC	Optimal Soft Arc Consistency
PDB	Protein Data Bank
SAT	Satisfiability
SCP	Side Chain Positioning
SDP	Semi-Definite Programming

TK	Tanford-Kirkwood electrostatic model
TS	Tabu Search
UP	Unit Propagation
VAC	Virtual Arc Consistency
VE	Variable Elimination
WCSP	Weighted Constraint Satisfaction Problem
WPMS	Weighted Partial MaxSAT

Introduction

Over the years, research in biochemistry and biophysics has accumulated knowledge to understand fundamental properties of biomolecules and their functions. Among biomolecules of primary importance are proteins. They are involved in a wide range of biological functions such as structural properties, signaling and catalysis. Their shape and the relationships between their structure and function have been widely investigated, enabling a fairly decent physico-chemical description and understanding of their properties and behaviors. With the ever-increasing number of biochemically characterized proteins and the growing number of protein structures available in databases, the exploration of structure-function interrelationships has become more and more approachable. Computational biology has also provided essential information to bridge the gap between static structural data and the huge amount of biochemical and kinetic information. More particularly, computational methods have led to major advances in the prediction of protein structures from their amino acid sequences, implementing principles of protein folding and their energetics. Molecular modeling techniques have also been widely used to investigate conformational flexibility and its impact on biomolecular recognition and protein function.

At a more applied level, there has been an increasing interest for proteins of all kinds (enzymes, antibodies, lectins, structural proteins, ...) for different uses in medicine (cancer therapy, autoimmune diseases, vaccines, neurodegenerative diseases, …), biotechnology (biosensors, biocatalysts, genetic/metabolic circuits, artificial biosynthetic pathways …) and/or nanotechnologies (construction of nano-devices and molecular machines, ...). Although the properties of natural proteins can be directly exploited, new, designed proteins, with novel functions or improved activities, are of major interest in all these application areas. Hence, protein engineering has become a key technology that has proven to be efficient and useful to modulate both the structure and the function of polypeptides and generate proteins displaying desired properties.

To increase the chances of success to tailor the wanted proteins, while reducing the human and financial costs, computational methods are in high demand. They allow guiding the evolution of the proteins on regions of their sequence space of particular relevance to achieve the desired function and thus reduce the size of mutant libraries to build and screen experimentally. This integration of computational approaches into engineering strategies has become essential to accelerate the design of neo-proteins. Computational Protein Design (CPD) methods have been developed for over a decade. They may involve the remodeling of a known protein scaffold in order to modify the protein function/activity, or, the complete (*de novo*) design of new protein to fulfill a particular function. The primary objective of CPD is the identification of a set of sequences that could fold into a predefined 3D scaffold, what is considered to be the inverse problem of protein folding prediction. In addition, CPD also addresses the requirements in order to find amino acid sequences that will optimize a desired property (physico-chemical property and/or biological function).

In spite of the significant advances in the field, one of the main challenges remains the very-high dimensional search space, which is the composite space of all the possible amino-acid sequences and all the possible spatial conformations of the protein, what optimization algorithms have to handle. Addressing this issue requires fast and efficient algorithms for searching large spaces, the accuracy of which is a fundamental necessity for successful protein design.

My thesis fits into this context with the main objective being the development of novel computational methods dedicated to the efficient and accurate combinatorial optimization. In the **first chapter** of this thesis, we give an overview of the main underlying principles of Computational Protein Design, their current limitations and the challenges ahead, their application to tackle different cases of protein design and at last, we provide a comparative analysis of a selection of existing CPD softwares. In the **second chapter**, we introduce some basic definitions and formalisms of the Cost Function Network (CFN) framework that will provide fundamental knowledge to understand how we used state-of-the-art CFN methods in the **third Chapter** to solve CPD problems. In collaboration with specialists in the field of combinatorial optimization, we developed a CFN-based framework that also includes a physical modeling of the CPD problem. To the best of our knowledge, our studies constitute the first reported CFN-based CPD framework in the literature. Motivated by the results obtained, novel CFN-based methods were then derived and implemented into a CPD-dedicated software called *osprey* to facilitate its access to the scientific community.

Results obtained within the frame of this thesis are reported as a series of three research articles already published and one that will shortly be submitted for publication in an international journal.

1 Computational Protein Design (CPD): Paradigms, Methods and Challenges

Contents

1.1 The CPD framework 5
1.2 Theoretical advances and applied interest of CPD 24
1.3 Concluding remarks, new trends and challenges ahead 29
1.4 References 32

Over the past few decades, the Computational Protein Design (CPD) has been proven very useful with respect to the development of proteins for many different applications. Major successes were achieved both in terms of methodological developments as well as biotechnological and biomedical applications.

In this field, pioneering work aimed at redesigning the hydrophobic core of existing proteins using simple knowledge about the stability of proteins [1]–[6]. Remarkably, these early successes led to the development of new properties not yet observed in nature [2], including the creation of a novel protein fold from scratch [7] as well as the amelioration of pharmacological properties of protein drugs [8]. Another milestone in CPD was achieved with the redesign of protein-protein interfaces including systems of biomedical interest [9], [10]. Early studies in this area were rapidly followed by the successful engineering of metal-mediated protein interfaces and metalloenzymes [11], [12], which were chosen as initial models due to the well-established definition of the transition metal coordination. A more recent landmark in CPD was the *de novo* design of enzymes from scratch, displaying original catalytic activities for which there is no counterpart in nature [13]–[15].

Such accomplishments were rendered possible thanks to the more and more realistic formulation of the design problem, especially with respect to *i)* the selection of suitable protein scaffolds for the design needs, *ii)* the treatment of protein flexibility in the design, in particular emanating from amino acid side-chains and protein backbone, *iii)* the development of well-adapted energy functions fast enough to allow high-throughput exploration of the sequence-conformation space, but still accurate enough to model properly protein properties and discriminate between sub-optimal sequence-conformation models, *iv)* the development of objective functions to adequately describe the fitness of the physico-chemical property and/or biological function that is sought.

Finally, significant algorithmic advances have been made to efficiently explore the associated sequence-conformation space and accurately predict the optimal sequence-conformation models. However, current trends in realistic representation of the design model require the development of more efficient algorithms to handle the huge sequence-conformation combinatorial space.

1.1 The Computational Protein Design Framework

Given a protein 3D structure scaffold, the structure-based computational protein design aims to find an optimal or an ensemble of near-optimal sequences. It is thus viewed as an 'inverse folding problem'. In addition to this primary goal, the CPD may seek to satisfy some design specific constraints. Physico-chemical properties and biological functions can be formulated as constraints (or objectives to the design). Schematically put, the CPD problem is to find, from an element of the structure space (the *input scaffold*), elements of the sequence-conformation space that are optimal with respect to the desired biological fitness expressed as an *objective function*. Inherently, structure-based design also requires considering the conformations accessible to the amino acids. Therefore, the search space of CPD is defined by both the possible amino acids and their associated conformations leading then to a composite *sequence-conformation space* modeling.

The CPD problem is then formulated as an *optimization problem*. The search space is often discretized and the energetics of the design space is captured by a *pairwise energy function*. The accurate modeling of these two representations of the search space has its own challenges (which will be discussed thereafter in this chapter). The objective function(s), built on top of the energy function is sought to mathematically express the design needs and constraints. The resulting optimization problem has to be handled by fast and accurate algorithms. Both the size of the search space and the cost distribution of the objective function are limiting factors for the combinatorial optimization.

Methodologies involved in a generic CPD framework can be clustered into three main components (Fig 1-1):

The first one is the *modeling of the CPD problem* which includes the selection of : (1) a protein 3D structure template (i.e. a 3D scaffold) or the definition of a minimal catalytic site and its grafting into a suited 3D protein scaffold for the *de novo* design of enzyme active site; (2) a search space by defining the variable amino acids of the molecular system to design (namely the mutable amino acid residues and the flexible residues for which only the conformational variability is explored so as to readjust the protein conformation to mutations); (3) an objective function relevant to the design target.

The second component is the *optimization phase* which aims to find (the) solution(s) in accordance with the defined objective function, given an initial input model. Both deterministic and stochastic optimization approaches have been widely applied for the computational redesign of relevant biological systems.

The third phase is then the *analysis and ranking of the design results* which usually consists in performing further *in silico* simulations (such as Molecular Dynamics simulations, Free energy calculations,..), both more accurate and more costly in terms of CPU-time than CPD methods, on selected models resulting from the design with the aim of refining them with more realistic physical concepts.

Following the computational design procedure, experimental validation (including stability assessment, structural characterization, functional assays…) can then be performed on the most promising designed sequences. In many cases, an additional optimization of the designed enzymes by subsequent rounds of directed evolution can be conducted to randomly introduce additional mutations which can tune up the targeted property and catch up some imperfections of the CPD modeling [16].

Needless to say that improvement of design methodologies will only be rendered possible by the availability of feedbacks from *in silico* refinement and experimental evaluation. Both successes and failures provide crucial information to enhance reliability and accuracy of computational methods [17] .

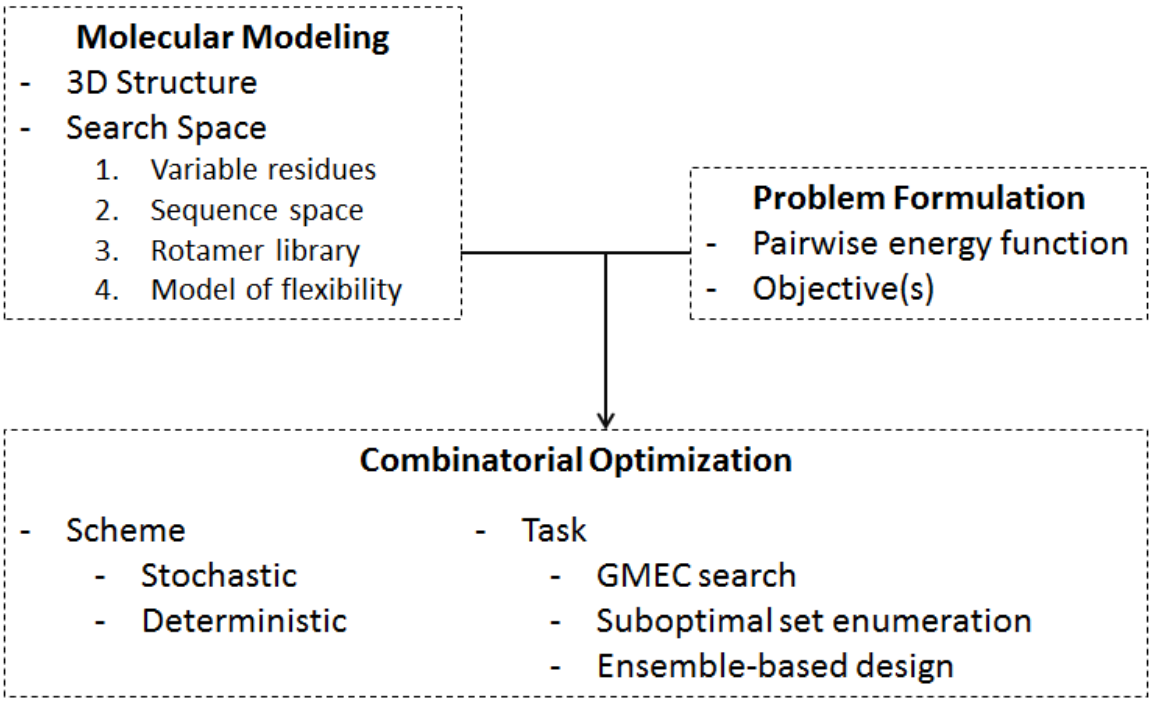


Fig 1-1 A generic CPD flowchart.

The CPD input model to the combinatorial optimization component is defined by the 3D structure template, the definition of the search space (the variable parts of the molecular system to design and their possible states), a pairwise-decomposable function to compute energy of inter and intra molecular interactions along with objective function to be optimized. The scheme of the optimization can be either stochastic of deterministic. Usual tasks to perform are: GMEC search, the enumeration of a suboptimal set of solutions and ensemble-based design. All elements are detailed thereafter in this section.

1.1.1 The input protein 3D structure template

The choice of the initial three-dimensional protein structure template is crucial to ensure its suitability for the design needs. The protein structure template is mainly selected amongst high resolution 3D structures determined either from X-ray crystallography or solution NMR studies (and deposited in the Protein Data Bank: http://www.rcsb.org). In some cases, when no structure is available, a 3D model can be built by comparative modeling using the existing 3D structures of

homologous proteins. The re-use of an existing structure has a practical advantage, since nature often produces proteins with vastly different functionalities using the same protein fold [18], [19].

In some cases, when *de novo* design of proteins from scratch is undertaken, it might require novel hypothetical scaffolds which can be built using databases of protein 3D fragments. However, this leads to a tremendous increase in the number of adjustable degrees of freedom (DOFs) for the polypeptide backbone, what results in a highly complex dimensional search space. When considering the *de novo* design of an enzyme active site, quantum chemical calculations are also used to build beforehand a minimal active site (theozyme) which will be grafted in a next stage onto the chosen protein scaffold [20].

1.1.2 The search space: degrees of freedom of the molecular system

The definition of the search space involves the selection of variable amino acid residues, as well as the way their conformational variability is taken into account. Whole or a subset of amino acid residues of the system can be considered as variable. Some variable residues are considered as *mutable* and their possible states are defined by two levels of degrees of freedom (DOFs): the amino acid types permitted at each position and their possible conformations. The selection of the mutable residues will pre-condition the chances of success of the design. Depending on the objective of the design, some residues might not be tolerant to mutations. For example, catalytic residues cannot be considered mutable if the enzyme catalytic activity needs to be maintained. To select the group of amino acid types permitted at each mutable position, CPD often takes into account the location of the mutable residues within the protein structure in order to preferentially introduce hydrophobic amino acids in the protein core and hydrophilic ones at the surface and maintain a stable protein [6], [21], [22]. In addition to mutable residues, a set of *flexible* amino acid residues (where solely change of conformation is allowed) is also selected in order to enable adaptation of the protein structure to mutations.

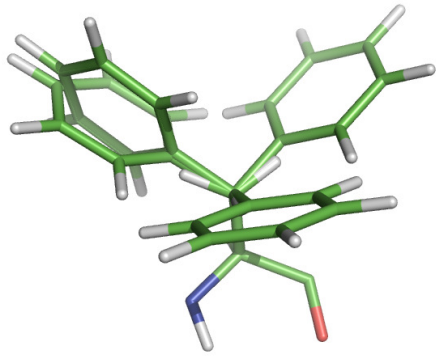


Fig 1-2 Amino Acid Rotamers (example of phenylalanine)

Amino acid residues have conformational DOFs, arising from the flexibility around their chemical bonds in a continuous angular space. The dihedral χ-angles represent the DOFs of the amino acid side chains and the dihedral angles ϕ and ψ from the main polypeptide chain, ω being mainly fixed. The real conformation space of a single amino acid can then be very large and ultimately, the size of the sequence-conformation space of a protein is continuous and huge. Consequently, for the sake of computational tractability, a first common approximation is to represent the side chain DOFs by a set of discrete conformations. Therefore, their continuous conformational space is approximated using discrete conformations defined by their inner dihedral angles which are called *rotamers*. These rotamers are low-energy side chain conformations derived from statistical analysis of high resolu-

tion crystal structures of the Protein Data Bank (PDB) which highlighted that the side chains of amino acids in protein structures avoid most of the theoretical conformational space and appear frequently as clusters in χ-angle space. An illustration for phenylalanine is given in Fig 1-2. In addition to the approximation made on the side chain conformational space, the classical modeling paradigm for CPD assumes a *fixed protein backbone* [23]. Therefore, the most basic CPD problem is viewed as a search for the optimal rotamers to fit on a given protein backbone. The search returning the optimal rotamers yields both side-chain conformations and the underlying designed sequence.

These fixed-backbone and discrete rotamer approximations have the advantage to dramatically reduce computation time. The fixed-backbone assumption also avoids reliance on energy potentials to discriminate between favorable and unfavorable backbone conformations. Despite these benefits, these conformational limitations have several consequences which limit the power of CPD approaches. Indeed, they artificially limit the natural degrees of freedom available to protein sequences folding into their stable conformations and thus may impair the CPD ability to reproduce native-like sequences [24]. In particular, numerous experiments have demonstrated that protein backbones adjust to sequence mutations [24]. Some side-chain conformations may be assigned with high energies and thus discarded during the search whereas slight backbone adjustments could have helped to correct such problems. Therefore, fixed backbone approaches may neglect a significant portion of sequence space which can lead to well-folded and functional proteins. Moreover, conformational changes of the protein backbone are crucial for the recognition and the interaction of proteins with their molecular partners such as a ligand, a cofactor, another protein… Therefore, further molecular flexibility has to be integrated in order to improve the accuracy of the predictions of the CPD approaches and extend the use of the latter to different design objectives. However, integration of more flexibility into CPD methods is a considerable challenge due to the huge increase in the combinatorial complexity and the needs for suitable optimization methods and accurate energy functions to discriminate effectively the multiple conformational states. In spite of these difficulties, several studies have tackled these challenges.

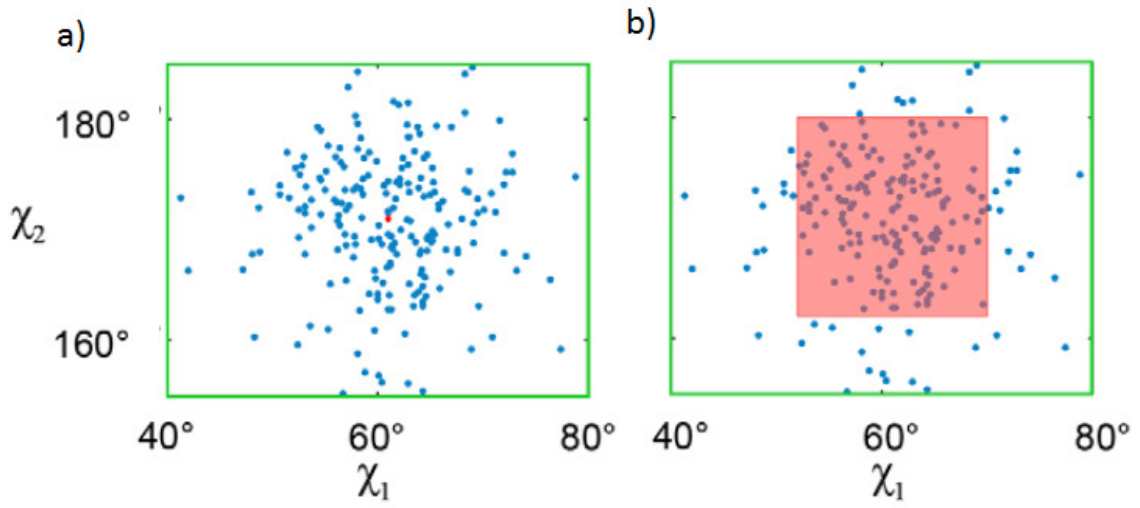


Fig 1-3 Distribution of isoleucine conformations in χ-space (adapted from [24]). The red region identifies the subspace covered by rigid rotamers (a) and continuous rotamers (b).

Recent works highlighted the gain in accuracy achieved by allowing *additional flexibility* resulting from the definition of a *continuous rotamer space* what reflects better the clusters of conformations observed for amino acids [24]. The case of isoleucine is exemplified in Fig 1-3. In particular, a study showed that the use of continuous rotamers in CPD allows identifying low energy sequences and ultimately more native-like sequences [24].

Various techniques have also been proposed to account for the flexibility of the backbone, both large-scale and local rearrangements. Mayo and coworkers incorporated backbone flexibility into the computational design of a novel protein topology, an α-helical tetramer with experimental validation, using well-characterized parameters of secondary structural elements [25]. However, this approach can only be applied if defined parametrical descriptions are available for the structural feature of interest. Random sampling of the backbone dihedral angles, previously described by Desjarlais and Handel [26] were also found effective in CPD experiments in combination with dihedral substitution using dihedral values extracted from the Protein Data Bank [7]. In a study using this later approach (conducted by Baker and coworkers), the CPD approach alternates between sequence optimization for a fixed backbone and backbone optimization for a fixed sequence. It has been successfully applied for the design of a novel protein fold [7]. Donald and coworkers also extended their fixed-backbone CPD approach to take into account the backbone flexibility by varying ϕ and ψ dihedral angles continuously within a voxel [27]. Other algorithms handle the backbone flexibility by setting upper and lower bounds on pairs of Cα-Cα distances and on the backbone dihedral angles [28].

The multi-copy backbone approach has been also employed in CPD. It uses several complete conformations of the backbone. These backbone conformations are used as template for multiple independent optimizations. Floudas and coworkers developed a method to calculate optimal sequences for a set of backbone templates derived from different structures of the same protein or extracted from molecular dynamics trajectories performed on the targeted protein [29]. They applied this approach to fully design the human β-definsin-2, a 41-residues peptide. The accuracy of the approach was assessed by comparison of the predicted sequences to the sequences of homologous proteins.

While above mentioned studies are based on conformational backbone sampling approaches developed in the field of molecular modeling, some recent solutions rely on robotics-inspired methods, namely inverse kinematics (or Kinematic Closure KIC) and Cyclic Coordinate Descent (CCD) algorithms [30]–[33]. These methods divide the protein backbone into small flexible fragments that are connected by fixed points called joints. Accessible conformations of the fragments are then searched while maintaining a constraint on joints in order to avoid chain breaks. The flexible backbone method based on CCD loop closure algorithm and fragment insertion method developed by the group of Baker optimizes backbone torsions until the break in the peptide chain resulting from the fragment insertion falls below a set threshold [33]. A CPD approach iterating sequence design and loop conformation optimization by this method has been successfully applied for the redesign of three loops in the protein tenascin. The designed variants were experimentally verified to form stable folded protein structures [34].

Following this success and in order to account for catalytic constraints involved in the design of enzyme's active site, the group of Baker proposed a method combining fragment insertion with CCD closure together with active site side-chains constraints. This procedure was used to produce a change in specificity of 10^6-fold in a human guanine deaminase for ammelide over guanine [35]. The crystallographic structure of the redesigned loop was found within 1 Å C_α rmsd of the proposed model.

While backbone flexibility methods described above explore the angular space of the backbone degree of freedom, another trend in CPD is to model the DOFs of the main chain by discrete local backbone states observed in 3D structures. Inspiration for such local perturbations came from observations that natural 3D structures usually undergo small characteristic backbone shift (3% of all residues) in order to accommodate changes in sequences [36]. These so-called backrub and shear motions (Fig 1-4) have been successfully applied to CPD [37]–[39]. The benchmark study on both GrsA-PheA and β1 protein domains [37] led to designed proteins with lower energies than those obtained from fixed-backbone approaches. Smith and Kortemme successfully applied their backrub-based method to model alternative side-chain conformations observed in high-resolution crystal structures and to improve the prediction of conformational adjustments to single point mutations over fixed backbone methods [38]. Following the same idea, Donald and coworkers have applied their backrub-based CPD method [37], combining sequence design and local backbone motion, to examine local backbone changes in both α-helix N-Cap and the glycine-aromatic coupling in antiparallel β-sheet [40]. The approach successfully reproduced *in silico* the N-Cap shift observed in natural protein structure for mutation of Ser/Thr to Asn/Asp. The computed shifts between mutants and wild-type for aromatic-glycine coupling in antiparallel β-sheet were also similar to those experimentally observed.

Although backrub approaches are efficient sampling methods to handle specific local backbone flexibility, the backrub is not a frequent type of motion (only 3% of all residues, mainly in extended structures [36]). Thus, it should be used in combination with other more general backbone flexibility modeling in order to better cover the natural backbone conformation space.

Another fundamental challenge in CPD is the need to accommodate the large translational, rotational and conformational *degrees of freedom of a ligand* within the already astronomically large sequence design calculations. When the ligand is a peptide, its conformational flexibility can be mostly described by discrete rotamer libraries, what is considered advantageous as it is consistent with the rotamer libraries used for sampling protein amino acid side chains. However, flexibility treatment of organic molecules, on the other hand, is often left to the user to pre-calculate an ensemble of low energy ligand conformers (rotameric combinations) with fixed geometry. Software solutions to construct continuous ligand rotamer libraries, better adapted to the search techniques used to explore protein residue flexibility, are generally lacking. Efforts have also been devoted to develop methods of *ligand placement* in the active site. These approaches are generally based on: (i) the use of stationary, rigid-body ligand poses in a large number of individual protein design calculations [42]; (ii) the generation of a discrete set of ligand poses filling the active site by rotation and translation [43]; (iii) the targeted placement of small molecule variations with reference to a contacting side chain [43]; (iv) iterative cycles of rigid ligand docking and sequence optimization [44].

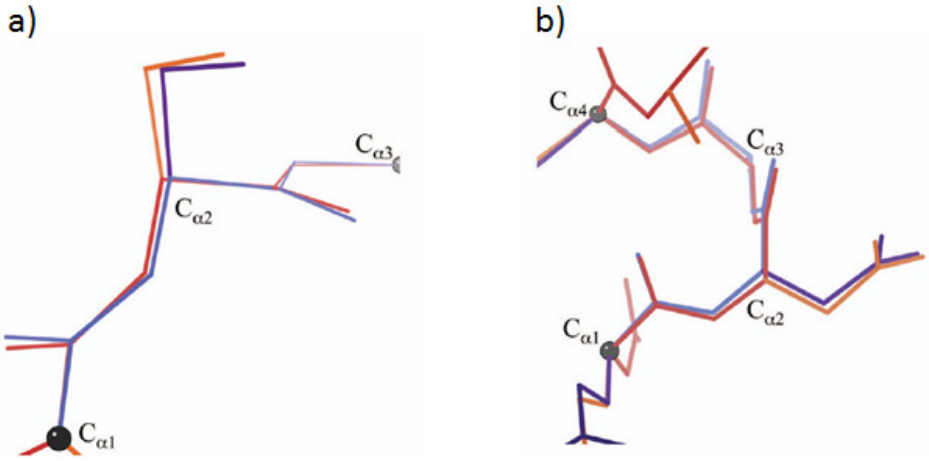


Fig 1-4 Two examples of local backbone motions (adapted from [39]): a) The Backrub motion involves a rotation of the central C_α with respect to the axis defined by its two neighboring C_α atoms. b) The Shear motion involves the displacement of a 3-peptide segment in a direction parallel to the axis defined by its external C_α atoms.

Recently, Donald's group extended its CPD approach to handle further protein main chain conformational changes [41]. They integrated within the same framework, approaches to handle multiple types of backbone freedoms denoted as perturbations. They include both backrub and shear motions, loop closure, secondary structure adjustments… Benchmark tests showed the ability of the framework to produce lower energy sequence-conformations than CPD studies which considered no or limited backbone flexibility.

1.1.3 The energy functions

After the brief introduction on the methods for modeling (macro)molecular flexibility in CPD, we will focus in this section on the most common all-atom energy functions used in CPD. All energy terms used in CPD are either inherently additive or generally approximated to be so. This greatly helps to speed up calculations during the energy computation. In addition, because the pairwise energy can be pre-computed and stored, the sampling of the same rotamer or rotamer pairs during the search does not require re-computing the same individual energy contributions. This pairwise representation also determines the type of applicable algorithms as deterministic optimization approaches (discussed in Section 1.1.5) require the matrix to be pairwise. The optimization will then read this matrix to find the optimal combination of rotamer assignments. Thus, given the pairwise formulation of the energetics, the total energy (E_{total}) of any given conformation has the following formulation in general:

$$E_{\text{total}} = E_c + \sum_i E(i_r) + \sum_{i,j} E(i_r, j_s) \qquad \textbf{(1-1)}$$

Where E_c is a constant energy contribution capturing interactions between fixed parts of the model, $E(i_r)$ depends on rotamer r at position i and $E(i_r, j_s)$ is the pairwise interaction energy between rotamer r at position i and rotamer s at position j.

Interactions between residues of the protein but also with their partners (ligand, cofactors or other proteins or nucleic acids, for example) are estimated by two types of empirical energy functions: *i) Physics-based energy functions* that are more precise but often heavy in CPU time and *ii) Statistical functions* that are often faster and derived from experimental structures (Knowledge-based). Hence, the type of commonly used energy expressions in CPD ranges from Molecular Mechanics (MM) force field energy functions to knowledge-based ones and also a combination of both.

The first ones (MM-based) are often altered in order to fulfill the requirements of CPD (fast and yet accurate energy terms in order to be applicable to the enormous sequence-conformation space). The individual energy terms include atomic *packing interactions* such as dihedral angles and reduced Lennard-Jones (and MM terms for bond stretching and bond-angle contributions are not always used in CPD experiments, because ideal geometry of these components is often used), *electrostatic interactions* between charges and *hydrogen bonds*. The *solvent* is represented by an implicit model, its explicit consideration being too expensive. Other ad-hoc terms are also used, for example to estimate the entropy of the side chains or the propensity of secondary structure elements. However, the latter one is generally accounted in other energy terms such as the electrostatic and Lennard-Jones, thus caution should be taken with the use of such additional terms.

Knowledge-based terms are used to model a broad range of energy contributions. Hence, both a backbone torsional angle and a rotamer statistical potential that accounts for the self-energy of a given rotamer have been used [45]. A pairwise electrostatic potential is also defined based on the probability of finding two particular amino acids at a given distance [45]. Knowledge-based terms can model complex effects difficult to be considered otherwise in CPD experiments, however, one difficulty in handling such terms lies in the introduction of additional non-overlapping energy terms during the development of energy functions.

A general energy function for CPD can be expressed in the form of a sum over individual empirical energy contributions E_k (either Physics-based or Knowledge-based) with a weighting factor w_k :

$$E = \sum_k w_k E_k = w_{vdw}E_{vdw} + w_{elec}E_{elec} + w_{solv}E_{solv} + \cdots \quad \textbf{(1-2)}$$

The weighting factors are usually optimized to reproduce some structural properties of naturally occurring proteins.

Finally, energy terms are not always straightforward to approximate both in an additive (interdependence) and in a pairwise manner. When designing an energy function for CPD, it is very difficult to avoid the overcounting of the same physical effect within the various terms [46]. Such artefacts can bias the objective function toward structural or functional properties not wanted or an unrealistic potential with respect to the target fitness property. This issue thus hardens the goal of bridging the gap existing between the fitness of computationally designed and naturally occurring proteins. Hence, for the CPD methodology to be accurate, both the selection of appropriate

energy terms and a correct balance between all these terms is to be accomplished [47]. This task is usually performed by optimizing the energy function using information derived from structural databases. One culprit with such knowledge-based optimization of energy functions is the assumption that native sequences are optimal with respect to the fitness function. Indeed, it is well established for example that native sequences are not optimized for stability [48] because they evolved in the complex cellular environment under multiple opposite constraints and functional requirements. Even more, such kind of energy function optimization generates weights that are dependent on the fitness function used for the energy function optimization itself and it thus limits its general applicability. An alternative approach is to train weights to predict the energy of mutations and pKa changes [49]. This method has the advantage of being defined as a change in free energy and not a fitness function. Hence, it has a wider applicability because of its independence with respect to any specific fitness function (such as stability or affinity) used in CPD experiments.

1.1.3.1 Lennard-Jones or *van der Waals* potential

A 12-6 Lennard-Jones (LJ) potential is usually used to account for the *vdW* contributions. Because of the usage of both the fixed backbone and discrete side chain rotamers, the LJ potential is not used as such in CPD in order to avoid artificial steric clashes. Although an extended rotamer library could be used to reduce this effect, it may not be always sufficient [26]. Hence, different weighting factors have been used on the repulsive (decreasing with R_{ij}^{-12}) and attractive (depends on R_{ij}^{-6}) terms in order to address these issues [45], [50]. A uniform scaling of the *vdW* radii in order to avoid artefacts of CPD approximations has also been used [51], [52]. However, these types of adjustments, while very simple to apply, result in a displacement of the well of the *vdW* potential. Thus, a linearized LJ more suitable for CPD approximations without displacing the well of the *vdW* potential was proposed [53].

1.1.3.2 Electrostatics and hydrogen bonds

The design of functional proteins is highly dependent on the modeling accuracy of the polar effect. Polar interactions such as charge-charge interactions (including salt-bridges) and hydrogen bonding are highly important for the secondary structures of proteins and they play a preeminent role in the specificity of intra-molecular and intermolecular interactions. Such interactions need to be precisely modeled if one wants to design realistic and functional proteins, enzymes and protein interfaces. However, accurate modeling of such effects in a CPD study yet remains an unsolved issue as it is strongly dependent on the local structural environment of interacting atoms which can be further modulated by the presence of the solvent.

A distance-dependent Coulombic term is often used to represent electrostatic interactions between two charged atoms. For hydrogen bonds, the simplest model is to account them within the electrostatic terms (implicit hydrogen bond). This formalism also takes into account the apolar part of hydrogen bonds in the computation of the *vdW* component. However, this simplification does not consider the correct orientation of hydrogen bonds. To address this drawback, an orientation-dependent hydrogen bond, based on geometric characteristics of hydrogen bonds observed in crystal structures, has been introduced by Baker and coworkers [54]. This model can be

viewed as an artificial covalent interaction. On a series of tests, their modeling led to more accurate results than a sheer Coulomb distance-dependent electrostatic model.

1.1.3.3 Solvent

The interactions of the solvent with macromolecules in the design system can be grouped into three classes: *vdW* (not directly modeled in CPD to the best of our knowledge), hydrophobic and polar effects. The explicit representation of such effects is highly computer demanding [55], [56]. Hence, such methods are very often avoided in protein design wherein large variations in sequence must be considered in addition to the enormous conformational space. Thus, *implicit solvent models* are often privileged. They allow calculating solvation energy terms without explicit representation of solvent atoms, which is often modeled as continuum media. Noteworthy, even the most accurate implicit solvent model, the so-called Finite-Difference Poisson Boltzmann (FDPB), is not tractable in CPD [57].

A simple model of solvation is the so-called Coulomb Accessible Surface Area (CASA) where the presence of the solvent is modeled by two terms [58].The first term accounts for the screening of electrostatic interactions between charged atoms by the solvent. The protein and the solvent are considered as having the same dielectric constant. The second term is a surface energy term that favors exposure of polar side-chains and burial of hydrophobic ones. It is proportional to an exposure preference that varies upon the atomic types (derived from experimental values) and the exposed area of the interacting atoms. It is not pairwise, as one atom can be in contact with two (or more) other atoms and its exposed surface depends on the entire structure. A pairwise approximation is thus needed [59], [60] in order to avoid overestimating this term. Of note, the most expensive part of this model it the computation of surface areas.

A more accurate implicit solvent model is the Generalized Born model **(**GB**)**. GB is a continuum electrostatic model in which, each atom is represented as a sphere with radius ρ_i (the burial of the atom in the protein) and has a charge q_i at its center and a dielectric constant of ε_p at its interior. The solvent is represented by a high dielectric constant ε_ω. Again, two terms are involved here. The first term of the GB energy represents the self-interaction of the atoms with the solvent, which is the interaction of an atom with the polarization it creates in the solvent. The second term is the GB pair interaction term. It describes the electrostatic interaction between two atoms through the polarization they create in the solvent environment. This term involves the charges of atoms and the so-called "*gb-function*" that depends on the entire structure coordinates (and thus, it is an inherently many-body function). Hence, it requires a pairwise approximation of atomic pair interaction terms for the calculations to be efficient. To this end, the GB model of Pokala and Handel which uses real atoms and backbone pseudo-atoms led to six order of magnitude speedup compared to the FDPB continuum model and successfully predicted the pK_a of more than 200 ionizable groups from fifteen different proteins [53]. A surface energy term was also taken into account within their GB model (GBSA). Similar continuum models such as Tanford-Kirkwood electrostatic interactions (TK) [61] harnessed by Havarnek and Harbury in order to make it less costly and the Lazaridis-Karplus model (LK) [62] are also in use. These last two examples have the advantage of not requiring the costly computation of surface area.

1.1.4 Fitness or objective functions

In addition to the accurate and efficient definition of the energy function at the atomic level, CPD optimization requires, for a given experiment, the definition of an objective that builds a mapping between the conformation space and the fitness landscape. Both physico-chemical properties (such as thermodynamic stability, foldability, and solubility) and functional properties (such as binding to a small molecule or another macromolecule) can be modeled as objective functions which will be optimized to get a stable and well-folded protein. Of note, during optimization, several objectives can also be combined in a single function (multiple objectives function) and negative design can be used to simultaneously optimize a fitness function that considers two or more competing states.

1.1.4.1 Designing for stability

The development of an objective function for protein stability requires knowledge about the unfolded state of the protein which unfortunately is not easily accessible. The thermodynamic stability is computed to approximate the folding free energy. The stability of a given sequence-conformation is the difference ΔG between the total free energy of the folded state and the total energy of the unfolded state. The energy of this reference state is thus as important as the absolute energy of the folded state itself ($\Delta G_{folded\ state}$). To generate models of the unfolded state, CPD methods usually use the common assumption that, in this state, the protein populates an ensemble of random extended structures. Hence, on average only very local interactions are accounted within this state. It is usually dependent on the amino acid composition.

A very simple and common model of the unfolded state is the extended tripeptide model where it is assumed that residues interact only with themselves and neighboring backbone and the surrounding solvent [5], [63]. Thus, the entire protein is modeled by a collection of n tripeptides of the form $Ala - X_i - Ala$ or $Gly - X_i - Gly$, where n is the size of the protein and X_i the amino acid at position i. From this modeling, the contribution of the tripeptide $Ala - X_i - Ala$, called its reference energy, only depends on the amino acid type at position i. The same force field for the calculation of the energy of the folded state is used to compute such contribution on a collection of tripeptides for every amino acid type and then it is averaged. The total energy of this state (ΔG_{ref}), which serves as a reference, is computed as a sum of the contribution of individual tripeptides.

Alternatively, the energy of such a state can also be calculated for each amino acid as a reference taking the best intra-rotamer energy of this given amino acid in the studied system and ignoring non-local interactions [64]. Another standard is to derive energy of the unfolded stated by maximizing the product (over amino acid type) of the Boltzmann factor of the self-energy of each amino acid type in a training data set of 3D protein structures [52]. Other methods such as that proposed by Serrano and coworkers implicitly account for the contribution of the reference state during the optimization of the energy function by fitting the weights of the objective function (free energy of unfolding in that study) to the changes in the stability of a mutant database [65]. The composition dependent reference state energy can also be computed from random sequence-

structure fragments (with 13-residue length) extracted from 3D structure datasets [53]. This fragment-based model of the unfolded state, first suggested by Rose and coworkers, was found to be more accurate than the tripeptide model in predicting the relative stabilities of mutations [66].

Finally, the objective function for optimizing the thermodynamic stability ($\Delta G_{folding}$) assumed to be proportional to the folding free energy is given by:

$$\Delta G_{folding} = \Delta G_{folded\ state} - \Delta G_{ref} \quad \textbf{(1-3)}$$

1.1.4.2 Designing for binding and multiple objective design: a challenging problem

The design for binding (enzyme active site and protein-protein interfaces) is a far more difficult task than altering the stability of a given protein. Since the evolution of new functions in nature might involve loss in stability [67], redesigning a binding interface requires a tight balance between preserving the stability of the protein and improving its binding or creating a new binding interface. Accurate modeling of enzyme's activity requires sophisticated calculations (such as quantum chemical calculations) that are however intractable for CPD problems due to the huge number of sequence models that need to be evaluated. This task is further hardened by the need sometimes to handle multiple substrates than can intervene in complex reactions. Furthermore, as exemplified by the limited number of successful *de novo* designs of enzymes in the literature, the concept of creating an enzyme able to catalyze a new chemical reaction is a very harsh problem, for which it is not even possible to ensure beforehand the availability of a suitable scaffold to design for the target reaction.

Also, for practical reason, the entropy of the active site is often neglected and allosteric effects that toggle the enzyme between active and inactive conformations are not always taken into account when designing an objective function. It has also recently been emphasized that the pre-organization of the active site reduces the entropic cost upon binding and importantly electrostatic pre-organization is a key to catalysis [68]. Thus, even the unbound state has to be somehow optimal in order to improve binding. Another modeling burden in engineering enzymes is the consideration of residues far away from the active site that still may have a critical long distance impact on the accommodation of active site residues. These mutations can compensate for the destabilizing effect of some mutations previously introduced to favor the binding [67]. Alternatively, such amino acid residues located for example at the protein surface can also evolve to optimize substrate recognition.

A simple scoring of the binding free energy is given by the following objective function (Equation (1-4)) that depends on the energy of the bound state (ΔG_{bound}) and the energy of the free partner, $\Delta G_{unbound}$ (corresponding to $\Delta G_{\text{protein}} + \Delta G_{\text{ligand}}$ for a protein-ligand complex) :

$$\Delta\Delta G_{binding} = \Delta G_{bound} - \Delta G_{unbound} \quad \textbf{(1-4)}$$

The amplitude of the binding energy is often small compared to the $\Delta G_{folding}$ because of a smaller number of residues involved in the binding interaction. Hence, favoring the binding requires finding a good balance between the contribution of $\Delta\Delta G_{binding}$ (contribution of residues at the binding interface) and the $\Delta G_{folding}$. A two-step approach to the design of binding proteins involves the generation of suboptimal sequences within a given threshold of thermodynamic stability loss and next, these sequences are ranked according to their binding. Such approach can involve calculation of an ensemble of structures for selected mutants in both bound and unbound states. An averaged binding energy for selected mutants which can be estimated either by conventional CPD approaches or during a refinement through MD simulations at equilibrium using an accurate continuum electrostatic model coupled with MM and surface area terms. Both approaches are based on the fundamental fact that binding is implemented by a *thermodynamic ensemble* of low energy conformations [69]. Similarly, the so-called K^* algorithm, is a recently introduced ranking procedure in the CPD framework to approximates the binding constant by means of conformational sampling of a Boltzmann ensemble [70], [71]. Of note, this stage is performed at fixe sequence.

Finally, following the fact that most mutations are destabilizing [16], [72], caution should be taken when designing for binding. Indeed, it has been shown that 70% of mutations are destabilizing [73]. The so-called *Pareto* sets are suitable to handle such consideration because a destabilizing mutation that does not confer a decrease in binding energy is not part of the *Pareto* optimal set [74], [75]. Thus, a sheer CPD approach to the design of function (i.e. mutations are allowed at all steps) is based on the building of multiple objective functions that can enumerate a set of non-dominated solutions (the *Pareto* optimal set) [74]. A multiple objective function combines a set of objectives into single objective function. Therefore optimization algorithms described hereafter should be applicable with such objective functions. The so-called Weighted Sum approach is the most common aggregate function used in CPD to formulate such multiple objectives problem [74]. Its formulation is given by:

$$score = (1 - w)\,\Delta G_{folding} + w\Delta G_{binding} \quad \textbf{(1-5)}$$

In this equation above, w controls both the weight of the contribution of the free energy of folding and the binding energy. In order to enumerate a *Pareto* optimal set, w was varied from 0.05 to 0.95 by steps of 0.05 in an earlier report [74].

1.1.5 Optimization algorithms: in the quest of a tradeoff between accuracy and speed

The objective function which is expressed in terms of the energies of rotamers and pairs of interactions is the target function to be minimized by the optimization algorithms. Several goals can be sought along this step such as the identification of the Global Minimum Energy Conformation (GMEC), the enumeration of a suboptimal solution set within a given interval of the GMEC, or the identification of a set of local minima, preferably including the GMEC.

From an algorithmic point of view, the CPD problem is NP-hard [76]. Even more, the side chain placement problem (included in the CPD problem) is NP-complete to approximate [77]. Because

of this and of the high dimensional nature of the structure-based computational design, metaheuristics, mainly stochastic methods have been extensively developed to solve practical CPD optimization problems. The first family of stochastic methods in the CPD field is often variants of the Monte Carlo (MC) or the Genetic Algorithm (GA). Briefly, these approaches are characterized by random local choices made during one cycle of the optimization process. Thus, two independent runs can give different results corresponding to distinct local minima. Hence, such approaches do not guarantee the identification of the GMEC for a single heuristic cycle. However, their reasonable CPU consumption allows performing thousands of runs in order to converge towards the GMEC. Thus, they are suitable to solving complex CPD problems. Another advantage of stochastic methods is their ability of approximating thermodynamic properties of the macromolecular system by conformational sampling at equilibrium at room temperature and pressure.

Wodak and coworkers performed a convergence analysis of stochastic methods [63]. From their discussion, a critical point appears to be the number of cycles. This issue is especially critical for the enumeration of a gap free list of solutions within a window of the GMEC. Hence, systematic analysis of the convergence of stochastic methods has to be performed to produce physically meaningful results. Albeit their advantages, the lack of provable guarantee with respect to the identification of the optimum of the objective function and ultimately their limited ability to precisely enumerate a suboptimal set of solutions in the vicinity of the GMEC makes their usage difficult in the context of the development of accurate scoring functions. Indeed, when discrepancies are found between experimental results and *in silico* experiments, errors due to insufficient sampling from stochastic methods (and other metaheuristics) cannot be distinguished from imperfections stemming from the modeling and the design of fitness scoring functions. Thus, if erroneous conclusions are made, the development of energy and objective functions may lead to over or underweighting energy terms or including additional terms not appropriate. In addition, because metaheuristics are incapable of recognizing the GMEC, even when they already found it, subsequent runs may continue enumerating solutions (with possible redundancy). This behavior tends to degrade the speed of metaheuristic methods for high dimensional space problems [78].

To alleviate these limitations and render more manageable iterative cycles between *in silico* optimizations, experimental testing and the conception of accurate fitness scoring and energy function, the development of complete exact deterministic methods is thus of uttermost interest. The most commonly used deterministic algorithm in CPD is the Dead-End Elimination (DEE), first introduced by Lasters and coworkers [79] for side chain placement problems. DEE being incomplete (i.e. it does not always converge to a single solution), it is often used in combination with the well-kwown exhaustive search algorithms A^* in order to extract the solution(s) from the remaining search space [63], [80]. As an alternative, Integer Linear Programing (ILP) approaches have also been applied to the CPD problem [81] with encouraging performance as well as dynamic programming [82]. Some CPD experiments have also been performed using methods based on the mean field theory [83]–[86]. However, while being deterministic in the sense that they reproduce the same results in two subsequent runs, the GMEC might not always be identified by mean field-based methods. Of note, mean field-based stochastic methods also exist [87].

Since DEE does not always converge to a single solution when it is faced with complex design problems, an enumeration algorithm such as A^* is then applied to extract the GMEC from the

remaining space. Two executions of these types of algorithm combinations give the same result and they provide the best mathematical solution (GMEC) when they converge. In addition, enumerating a gap free list of sequence-conformations within a user-specified energy window is straightforward with such method. However, since convergence is not ensured and A^* has an exponential time and space complexities, these methods can be extremely CPU-consuming. Therefore, they have often been limited to systems of small or medium sizes.

Thus, to provide provable accurate solutions to the CPD optimization problem, there is a crucial need to develop methods that give both the provable guarantee of complete exact deterministic approaches while being applicable to design systems of comparable size to those managed by metaheuristics. In subsection 1.1.5.1, stochastic approaches are briefly described with a focus on the MC and GA methods while subsection 1.1.5.2 reviews deterministic methods.

1.1.5.1 Stochastic approaches

Monte Carlo-based algorithms are the most commonly used heuristic algorithms in CPD [88]. In the field of structural bioinformatics, MC sampling has been first developed to sample protein conformations before its application to the CPD problem. The key feature is to rapidly generate local modifications of a starting structure and then accept or reject the new solution based on its evaluation using an objective function and an acceptation criterion. In an MC process, a Simulated Annealing metaheuristic is often applied (MCSA) in order to avoid local minima. The so-called Metropolis criterion constitutes such a metaheuristic.

The general shape of MC algorithms can be described as follows: a set of rotamers (of allowed amino acids) are first randomly picked for every variable position. Then, an iterative optimization procedure is applied in an attempt to converge to a minimum energy conformation [63], [89]. Each iteration cycle is called heuristic cycle and proceeds along the following manner: for each position (randomly selected), a rotamer that improves the value of the scoring function is repeatedly selected until the score no longer varies. In order to explore different local minima, an energy increase is accepted with a Boltzmann probability, thereby overcoming the energy barriers and local minima trapping. Thus, a simulated annealing can be coupled within the MC procedure by adding heating and cooling stages (MCSA) provided by the Metropolis criterion.

A second type of stochastic algorithm widely used in the CPD field is the Genetic Algorithm [90], [91]. It simulates natural selection on a set of structures by performing three evolutionary processes: mutation, selection and crossover (recombination). While MC works on one sequence-conformation at a time and tries to improve its score, genetic algorithms work on a population of sequence-conformations and attempt to improve the fitness of the population by applying mutations followed by several cycles of selection to the whole population.

GA runs as follows [78], [90]: a population of M structures is generated beforehand and define the parent structures for the next evolutionary process. Parent structures are mutated according to a given probability distribution associated to rotamers and a set of best mutants is selected for recombination. Next, a selection tournament is then applied to the generated population of se-

quence-structures: S mutated sequences are picked at random. The best element form S is selected to build one instance of the next generation of sequence-conformations population. This selection step is repeated M times in order to produce the whole population of the next generation.

The whole procedure is then repeated until population equilibrium is reached. As in an MCSA procedure, a heating and cooling stage can be simulated at each step by varying the size of S, thus tailoring the pressure of selection.

Finally, in order to further overcome local minima and improve the convergence of stochastic methods, hybrid methods which combine deterministic methods and stochastic ones such as the FASTER algorithm have also been developed and applied to CPD with significant improvement in the runtimes with respect to the identification of solutions which were nearly identical to the true GMEC [92], [93] .

1.1.5.2 Deterministic approaches

The Dead-End Elimination

The Dead-End Elimination (DEE) algorithm prunes rotamers and rotamer pairs that cannot exist within the Global Minimum Energy Conformation (GMEC) [79]. Similar criteria also exist for discarding pairs of rotamers or combinations of rotamers of higher orders. These elimination criteria are applied until convergence or a predefined number of steps are reached. DEE guarantees that the GMEC is found if the optimization converges but it does not guarantee the convergence.

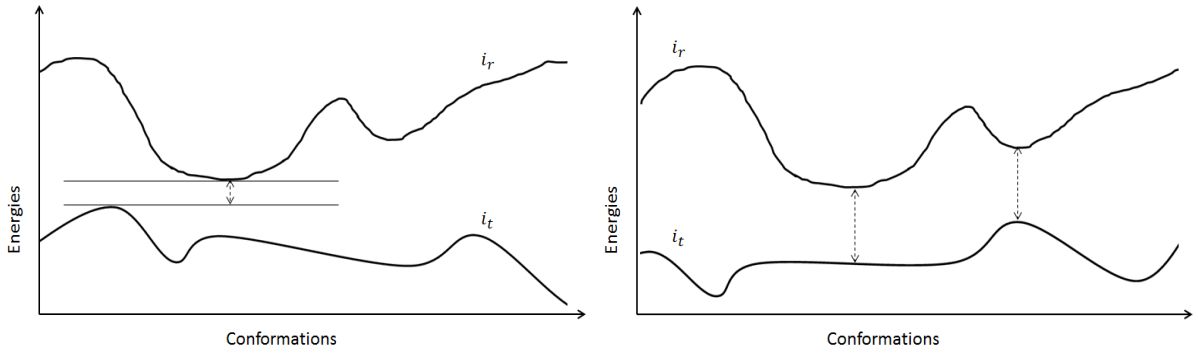
More formally, the original DEE single elimination criterion eliminates a rotamer i_r at position i if an alternate rotamer i_t exists for which the worst energy contribution is smaller than the best energy contribution of i_r. The original simple DEE criterion is given by the following equation:

$$E(i_r) + \sum_{j \neq i} \min_u E(i_r, j_u) \; > \; E(i_t) + \sum_{j \neq i} \max_u E(i_t, j_u) \qquad \textbf{(1-6)}$$

Single rotamer elimination at one step modifies the conformational space, thus allowing further elimination at subsequent steps. The iteration continues until no dead-ending rotamers can be found. In addition, various elimination criteria have been developed in order to improve the elimination efficiency of the original DEE. The so-called Goldstein DEE provides a more restrictive DEE criterion [94]. It considers that i_r can be eliminated if its energy contribution is always lowered by using an alternate rotamer i_t in the same conformational context (absent in the original formulation above) as expressed by the following equation:

$$E(i_r) - \; E(i_t) + \sum_{j \neq i} \min_u [\, E(i_r, j_u) - \; E(i_t, j_u) \,] \; > 0 \qquad \textbf{(1-7)}$$

The **Fig 1-5** exemplifies these two well-known single DEE pruning criterion.



a) DEE Desmet *b) DEE Goldstein*

Fig 1-5 Example of DEE elimination criteria

Other important improvements include splitting criteria that allows to use several alternate rotamers in order to conjointly eliminate a given rotamer [95]. For a given rotamer under analysis, it splits the conformation space in two (or more) regions and seeks an alternate rotamer for i_r in each region. The two regions must have a different rotamer choice for at least one position ($k \neq i$).

Following these developments, Donald and co-workers have also proposed a DEE algorithm that uses a *divide-and-conqueror* technique for more splitting efficiency [96]. It splits the conformation space into different partitions (subspaces). Indeed, partitions are pruned independently to each other. Thus, this algorithm can take advantage of parallel computing to speed up the search. The minimum of the remaining conformations is the GMEC.

Ultimately, most protein design procedures include an energy minimization step after the optimization step. Hence the so-called rigid-GMEC obtained using traditional DEE presented above and considering fixed backbone and discrete side chain rotamers may suffer from some geometrical restraints leading to high energies that will need to be relaxed. This remark pointed out the need to develop versions of DEE criteria provably accurate for design process that includes energy minimization steps. Donald and coworkers have proposed such criteria where rotamers are allowed to move within a defined voxel [97]. The computed GMEC is denoted min-GMEC, by analogy to the rigid-GMEC. In addition, for the sake of further flexibility, criteria to handle the backbone flexibility have also been defined as well as for handling continuous rotamers (briefly introduced in Section 1.1.2) [24], [27], [41].

For the sake of computational efficiency, variants of DEE criteria use Monte Carlo computed "reference" energy (which should not be confused with the unfolded state energy) to eliminate rotamers [98] and identify dead-ending rotamer pairs. When the DEE procedure includes this stochastic bounding criteria as well as its flagging version for dead-ending pairs, it provides the GMEC when the algorithm converges. Otherwise, the remaining conformations dependent on the computed reference energy. Thence, whenever exact, the process is no longer deterministic.

The A* *search algorithm*

$\boldsymbol{A}^*$ is a Best-First Search algorithm widely used in artificial intelligence [99]. In CPD, after DEE pruning, this complete search algorithm can be applied to the remaining rotamers [80]. Despite its exponential complexity, the advantage of the A^* algorithm is that, the first identified conformation is the GMEC. In addition, it can also enumerate energy conformations within a given threshold from the GMEC ranked in an increasing energy order.

Briefly, the $\boldsymbol{A}^*$ algorithm maintains a priority queue of nodes open for expansion. These nodes are the visited nodes so far for which children nodes are not visited yet. For a given node, its priority score is simply a lower bound on the lowest energy conformation accessible from that node. $\boldsymbol{A}^*$ picks the highest priority node from the expansion queue and selects one unassigned residue. The selected node is expanded by assigning the selected residue to each of its allowed rotamers. The associated energy lower bound is then computed using a so-called *admissible heuristic*. An admissible heuristic guarantees that the computed cost is a lower bound of the optimum. The search is terminated (for the GMEC search) if every variable residue has been assigned.

The optimality of $\boldsymbol{A}^*$ relies on the admissibility of the heuristic used to compute the lower bound. The heuristic is given by the following equation [80]:

$$f(i,n) = g(i) + h(i,n) \tag{1-8}$$

In this equation, $f(i,n)$ is the (under)estimated cost from node i to the goal node (n). The component $g(i)$ is the cheapest cost so far from the root node to the current node (i) and corresponds to the contributions of the partial assignment. Thus the terms $g(i)$ are simply given by:

$$g(i) = E_c + \sum_{j<\mathrm{i}} E(j_r) + \sum_{l=1}^{i} \sum_{j=0}^{l-1} E(i_r, j_s) \tag{1-9}$$

The condition $j < i$ in the equation is required to select only the already assigned variable.

The term $h(i,n)$ is a heuristic function used to compute a lower bound on the cheapest cost from the current node to the goal node considering the contribution of unassigned variables. The admissibility criterion requires to not overestimating this cost. Simply put, it estimates the cost required to complete the partially assigned conformation in an optimal way. The closer is this heuristic to the real cost, the efficient is the search performed. The term $h(i,n)$ is given by:

$$h(i,n) = \sum_{j=i}^{n} \min_{s} \left[E(j_s) + \sum_{l=0}^{i-1} E(l_r, j_s) + \sum_{k=i, i<j}^{j} \min_{t} E(k_t, j_s) \right] \tag{1-10}$$

Following this formulation of f, the first leaf node to be reached is the GMEC because every time a node is expanded, an underestimate of it score is used as its f value and there is no successor node to be expanded at leaf nodes. Thus, the f scores of leaf nodes are the true cost of the associated conformations. An illustration the A^* algorithm is given in **Fig 1-6**.

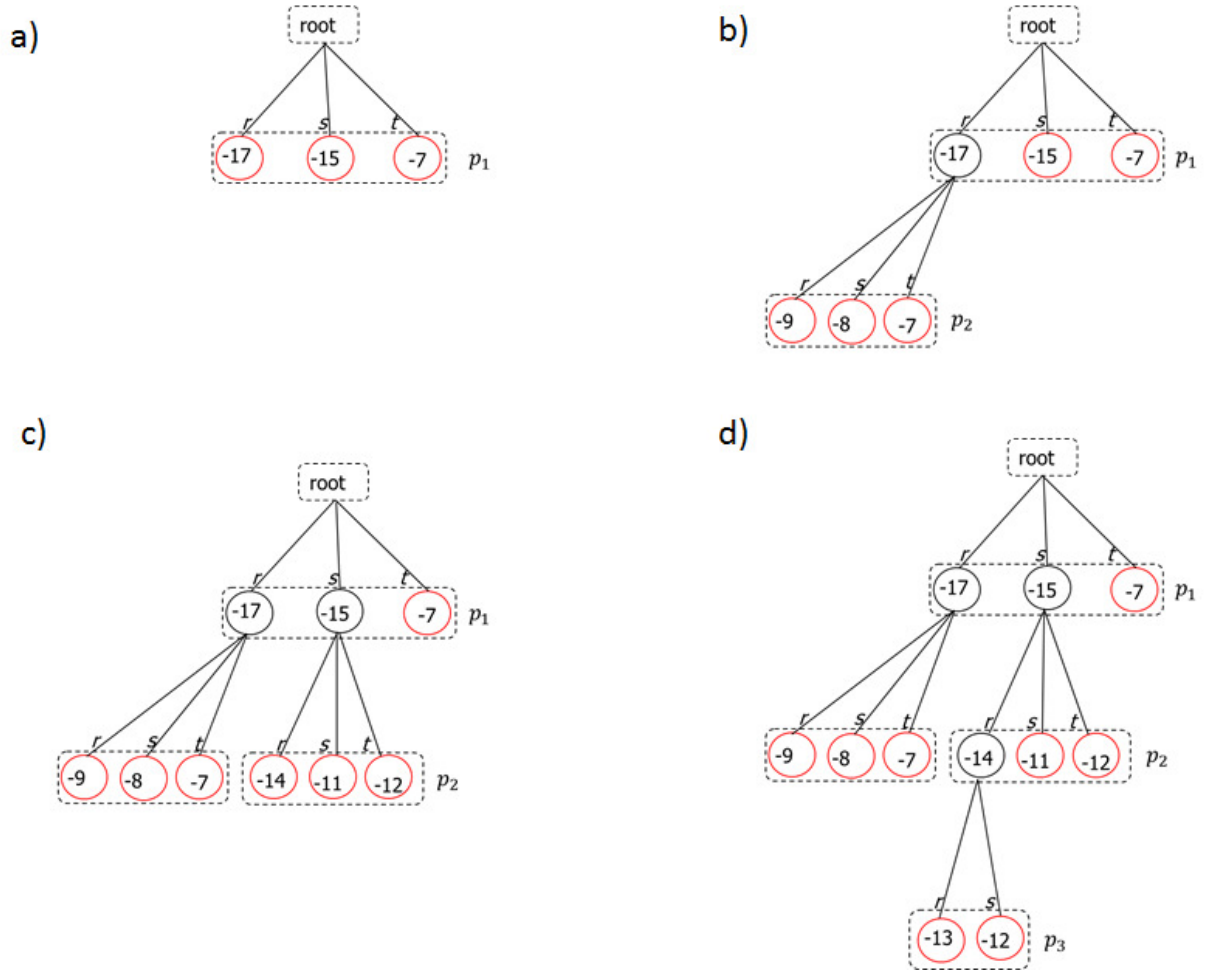


Fig 1-6 Illustration of the A^* algorithm. In this toy CPD example, we have three variable positions $\{p_1, p_2, p_3\}$ with respectively rotamer sets $\{r, s, t\}, \{r, s, t\}, \{r, s\}$. Each variable corresponds to a level in the search tree. The values associated to nodes are the corresponded (f) score. At each step, the nodes present in the expansion queue are colored in red. Already expanded nodes are colored in black (i.e. nodes for which we explored the successors). Subfigures a)-d) represent the state of the search after four consecutive expansions. The expanded nodes from one step to the next step switch its color from red to black. At root, no variable have been assigned yet. In a) p_1 is selected as an initial variable. For all its allowed rotamers the associated score is computed and the new nodes are then added to the expansion queue (identified by the red color). In the next step in b), the best node (lowest score) from the expanded node is selected and the process in a) is repeated for variable p_2. The resulting expansion adds three additional nodes to the expansion queue. Now, the best node has cost -15. Variable p_2 is selected again and expanded for that node leading to three additional nodes again with cost $(-14, -11, -12)$. Hence, the best open node has cost -14 and the only unassigned variable from that node is p_3. In d), p_3 is assigned to all its rotamers and two new nodes are added to the expansion queue (cost -13 and -12). Finally, the best node has cost -13 and all variables are assigned. Hence the GMEC has a cost -13 and the associated conformation is $(p_1, s); (p_2, r); (p_3, r)$.

1.1.6 Some selected software implementations

After introducing important algorithms developed in the CPD field or borrowed and adapted from artificial intelligence and computer sciences along with energy functions and modeling

trends in problem formulation, we outline in **Table 1-1** some freely available academic software implementations dedicated to a pairwise CPD formulation that have been extensively tested over the last decade. For each program is indicated the type of modeling (discrete, continuous or both) with respect to the optimization step, the methods of optimization (deterministic, heuristic or both), and the main type of energy force field used (physics-based or knowledge-based).

Table 1-1 Some selected CPD software implementations

Software	*Modeling*	*Optimization*	*Force field*
osprey [100]	Both	Deterministic	Physics-based
rosettaDesign [101]	Discrete	Heuristic	Knowledge-based
xplor/proteus [102]	Discrete	Both	Physics-based
designer [63]	Discrete	Both	Physics-based
ipro [44]	Discrete	Heuristic	Physics-based
orbit [103]	Discrete	Both	Physics-based
protdes [104]	Discrete	Heuristic	Physics-based

1.2 Theoretical advances and applied interest of CPD

In this section, we will present some selected examples of structure-based computational protein design studies which target different objectives. A special focus will be placed on key results with respect to the underlying fundamental advances and the potential future applications rendered possible both at the theoretical level and for biotechnology and biomedicine fields. Of course the section is not meant to be exhaustive; the CPD field is rich of interesting publications in this context.

1.2.1 From improved thermal stability protein to novel protein scaffold design

The mainstay paradigm in computational engineering of protein is based on the assumption that the core of proteins is mainly composed of hydrophobic residues and their surface is rich in hydrophilic ones, although protein core also includes polar residues and hydrophobic residues can also be found at protein surface [105]. Hence, pioneering CPD works mainly focused on the redesign of the core of existing proteins in order to improve their stability and develop robust biocatalysts able to perform under harsh conditions what is of main interest for the development of industrial bioprocesses [1]–[6].

In 1997, Mayo and coworkers used an automated framework to design *de novo* a protein sequence aiming at folding into a target zinc finger structure. Upon experimental validation, the designed sequence adopted a three-dimensional structure in solution whose root mean square deviation (rmsd) differed by less than 1.04 Å compared to the crystal structure [6]. Thus, for the first time, a CPD experiment was confirmed by an experimental crystallographic structure. Following this CPD success with proven structural accuracy, Kim and coworkers designed *de novo* several new proteins including proteins with structural features never observed so far in nature (namely right-hand α-helical trimers and tetramers) [25]. These designs led to protein structures with an impressive accuracy approaching 0.2 Å.

Due to geometric restraints induced by the protein backbone conformation, the redesign of naturally occurring proteins remains to some extent dependent on the protein scaffold used as template [7]. Indeed, an arbitrary backbone not found in nature may not be designable [106]. The utilization of such hypothetical scaffold to create novel proteins is a real challenge for computational protein design [106]. Therefore, the work conducted in 2003 by David Baker and coworkers [7] was a true tour-de-force as they successfully achieved for the first time the design with atomic-level accuracy of a 93-residue protein, called Top7, adopting a novel globular α/β fold never reported before. This achievement opened new ways to create original proteins not yet observed in nature and demonstrates the ability of CPD methodologies to design novel protein folds.

1.2.2 Protein interfaces

The design of protein interfaces holds great practical applications in the development of protein therapeutics and in nanobiotechnology through the design of self-assembling protein nanostructures [107]. In addition, both the creation of new vaccines through the design of proteins mimicking antigenic epitopes and the prediction of drug-resistant mutants involve the design of specific protein-protein binding interfaces.

In contrast to the design of protein scaffolds, the design of protein-protein interfaces faces a different challenge that aims at considering both the stability of the two binding partners (in order to preserve the folding of protein variants) and the affinity of redesigned or *de novo* designed interfaces. A basis for such task in CPD relies on the observation that hydrophobic patches are found at natural protein-protein interfaces. Although this is the easiest way to design binding interfaces, native protein-protein interfaces also exhibit important features such as networks of polar amino acid residues, more difficult to design artificially. An additional feature that makes the design of protein-protein interfaces more difficult is the presence of large conformational changes likely to occur upon the binding of two proteins. The binding event may be also accompanied by variations in the solvent exposure of some amino acid residues. These phenomena have to be carefully considered when designing protein-protein interfaces in order to be as accurate as possible [17].

The first structurally verified computational design of a protein-protein interface was reported by Clark and coworkers for the enhancement of the binding affinity between a specific antibody with the domain I of human integrin (VLA1), a cell-surface reporter present on some T-cells [9]. The crystal structure of a designed quadruple mutant confirmed the predicted contacts. On the same trend, in a study conducted by Baker and coworkers, a protein was computationally designed to bind the conserved stem region of influenza hemagglutinin by enhancing the shape complementarity of the binding interface [10]. As a result, two designed mutants (out of 73) exhibited binding affinity toward the stem region. Albeit the success rate of this study is very low, it demonstrates nevertheless the potential of computational tools for biomedical applications and underlines the necessary interplay between computation and experiments.

Finally, following the observation that β-strand pairing stabilizes many naturally occurring interfaces (up to 8.8% of contacts in homodimers [108]) including antibody-antigen interactions [109], Kuhlman and coworkers demonstrated the applicability of this feature in a general compu-

tational procedure to redesign binding interfaces mediated by intermolecular β-sheets [110]. As a benchmark study, a monomeric protein was remodeled to form a homodimer through intermolecular interactions involving solvent-exposed β-strands. Upon experimental characterization, one out of four designs confirmed the predicted model with a 1.0 Å rmsd.

Noteworthy, while the majority of reported computational designs of protein interfaces aimed at altering existing binding interfaces and often led to low binding affinities, the first *de novo* design of a novel binding interface formed between two proteins was recently carried out and led to an impressive sub-nanomolar affinity [111]. Indeed, the *de novo* designed interface was experimentally shown to have a 1000-fold better dissociation constant (compared to previous reports). Two mutants obtained through directed evolution further improved the initial design (dissociation constant improved from 130nM to 180pM).

1.2.3 Metal-mediated enzymes and biosensors

The computational design of metal binding interfaces has been widely explored in order to validate the potential of CPD methods within the context of small molecule binders and transition metal-mediated enzyme design. In 1991, Richards and coworkers introduced successfully, with experimental verification, a copper-binding site into the *Escherichia coli* thioredoxin using the natural geometry of the copper-binding site [112]. To do so, the geometric description of functional groups (and their degrees of freedom) relevant to copper-binding and the surface shape complementary around the ligand were used to build a novel active site onto an inert protein scaffold [113]. To build the new ligand binding site, the algorithm searches through a constellation of backbone positions, one that could lead to the stabilization of the ligand via appropriate amino acid side-chains. The new binding site is required to satisfy the user defined geometry and the degrees of freedom of appropriate partners and maintain intact the protein backbone. Albeit the experimental copper-binding mode turned out different from the predicted one, the designed protein variant was able to bind to copper providing thereby a simple and elegant proof-of-concept of the applicability of the CPD methodology to engineer molecular recognition using simple geometric criteria. Following this pioneering work, many successes have been reported in the field of metalloenzyme computational design, starting with the *de novo* design of a novel enzyme catalyzing the dismutation of the superoxide anion [114], the design of mononuclear iron-sulfur center with electron transfer activity [115], the design of zinc ion-mediated biosensor from maltose receptor through the manipulation of conformational equilibrium between maltose-bound and -unbound states [116], and the design of biosensor specificity toward a number of small molecules including serotonin and trinitrotoluene [117]. These studies were followed in 2004 by the *de novo* design of a four-helix bundle fold with O_2-dependent phenol-oxidase activity [118] and by the design of a calmodulin-dependent protein displaying up to 900-fold increase in binding specificity [11] as well as a metal-mediated symmetric homodimer of Rab4 binding domain whose x-ray structure showed a C_α rmsd of 1.4Å compared to the computed model [12].

1.2.4 Redesign and *de novo* design of enzymes

Naturally occurring enzymes have already been found to catalyze a very broad range of chemical reactions. However, there is still a crucial need for novel biocatalysts able to perform not yet re-

ported reactions (catalyzed by an enzyme) or displaying properties compatible with the needs of biotechnological and biomedical applications. In response to these needs, the *de novo* design of enzymes from scratch to develop tailored catalysts remains the holy grail of structure-based computational design of enzymes. In recent years, important progress has been made in this field by means of computational methods alone or by combining the computational design with directed evolution techniques to accelerate the optimization process [119]. Some of the most important achievements are listed hereafter.

The computational design of new functional proteins from scratch toward new ligand represents a great fundamental challenge and a real breakthrough for the development of enzyme-based biotechnological processes. A first issue is that one has to find a relevant starting structure for the active site and protein scaffold. There is no assurance concerning the designability of the protein scaffold for the *de novo* designed reaction [106]. This is one main difference compared to the redesign of existing enzymes toward new ligand specificity which usually require to preserve an existing catalytic function while designing new recognition or altering the existing activity. The ultimate difficulty when designing for a new reaction is that failures are often hard to explain, whether they are due to the fact that the selected scaffold is not designable for the target reaction or to the inaccuracy of the CPD process [106].

Earliest reports rather focused on the redesign of existing enzymes in order to improve the binding affinity or to perform a substrate specificity switch, the design of metalloenzyme binding site, or the grafting of a catalytic activity already present in nature into an inert protein scaffold. One of the pioneering work aiming at introducing (nonmetal-mediated) catalytic activity into an arbitrary protein scaffold was conducted in 2001 by Mayo and coworkers who designed a histidine-bearing catalyst for the hydrolysis of *p*-nitrophenyl acetate (PNPA) into *p*-nitrophenol [103]. The input scaffold used was an inert thioredoxin with respect to PNPA, thus demonstrating the ability of CPD methodology to generate *de novo* enzyme-like protein catalysts (*protozymes*) with catalytic activities comparable to those of catalytic antibodies (Abs).

A momentum was achieved in 2008 when David Baker and coworkers created from scratch an entirely new catalyst able to catalyze the Kemp elimination reaction, a proton transfer reaction that takes place in one step restricted by high activation-energy barriers [13]. Such reaction is not catalyzed by any known natural enzyme and therefore, this was the first demonstration of the creation of a novel catalyst with no counterpart in nature. In this work, the first challenge was to generate a detailed molecular model of the chemical reaction center including important functional groups and catalytic residues oriented in productive conformation with respect to the substrate. Using Quantum Chemical calculations, a library of the putative transition state models of the reaction were obtained in order to define the minimal active site (called *theozymes*) [120]. Next, inverse rotamers tree and geometric hashing techniques were used to select out of a large set of existing folds, the most suitable protein scaffolds able to accommodate the grafted theozymes [20]. In the last stage, the overall complex was optimized through the introduction of selected mutations and the rearrangement of amino acid side-chains. All designs were then evaluated and ranked using scoring functions to identify the most promising models to evaluate experimentally. In this study, out of 59 designs (covering 17 different scaffolds), 8 exhibited measurable activity. These active designs contained from 10 to 20 mutations. The moderate activity found

for the computationally predicted enzymes was subsequently enhanced at experimental level through several rounds of directed evolution which aimed at introducing subtle (distal) mutations beneficial for the activity of the designed variants. Indeed, these mutations are not straightforward to predict. They also overcome some of the weaknesses and approximations of CPD methods, in particular inaccuracies in the consideration of molecular flexibility induced by mutations during the design procedure and the use of objective functions not well-adapted to target biochemical fitness. Such combined strategy led to a more than 200-fold improvement of the catalytic efficiency (k_{cat}/k_{uncat} ratio) compared to initial designed enzymes and the introduction of between 4 to 8 additional mutations. Hence, the final best design led up to a total of 10^6 k_{cat}/k_{uncat} ratio.

The same year and using similar approach, David Baker and coworkers took up the challenge one step further with the creation also from scratch of a novel enzyme able to catalyze a retro-aldol reaction, that involves multiple steps, thus multiple intermediates that considerably complicated the design procedure [14]. Out of the 72 experimentally characterized predictions, 32 designs showed detectable catalytic activity with up to 10^4 of catalytic efficiency for the best design. These variants spanned over 10 different scaffolds and led to the introduction of up to 20 mutations. Four different active site motifs were covered by the design. The crystal structure of two variants confirmed the correct placement of the catalytic residues, although their active site loops exhibited a distinct conformation due to the absence of explicit backbone flexibility [14].

Finally, another important accomplishment was achieved in 2010 with the *de novo* design of an enzyme able to catalyze a stereoselective and bimolecular reaction, the Diels-Alder reaction. This reaction is a carbon-carbon bond forming reaction. The Diels-Alder reaction is a bimolecular and regio-selective reaction in one step. The relative bound position and orientation of the two substrates have to be strictly satisfied. Hence, this was the most complex *de novo* enzymes successfully designed so far. Out of a total of 84 designs, two exhibited the so-called Diels-Alderase catalytic activity. These primary designs were further improved by experimentally introducing 6 additional mutations at positions in the vicinity of the catalytic site. This led to 100-fold improvement of the catalytic activity for six mutations for one of the initial design. The second initial design was further optimized in four of its variants with up to 20-fold improvement in the catalytic activity.

However, although these realizations were considered as a real breakthrough for the fields of enzyme engineering, the methods have not yet reached a high level of quantitative predictive capability. The catalytic efficiency of these *de novo* designed enzymes remains far below that of naturally occurring enzymes [121]. Further work is still needed to improve computational methods by better accounting for the flexibility of protein backbone and ligand as well as integrating more accurate energy functions. Also at a more fundamental level, there is a need to understand in more details enzyme catalytic mechanisms, key factors involved in molecular recognition, and more particularly the dynamics of the interactions which are often critical when studying protein-ligand or protein-protein interactions.

Usually, CPD methods focus on the remodeling of a single state (or multiple static states) of the complex that is assumed to be critical for the catalytic step. The *in silico* design aims then at optimizing the set of mutations to introduce in the static protein in order to maximize the recognition of the substrate in its productive state. However, recent studies on the enzyme dynamics are pinpointing the importance of dynamics (protein flexibility, substrate binding, substrate accessibility to buried active sites, product release …) onto catalysis. Although its role is still largely discussed in the literature, it is more and more established that molecular flexibility, involving both local flexibility but also large-scale conformational rearrangements occurring at different stages of catalysis, can be critical to pre-organize the catalytic step even though they might not be directly involved in the chemical step [68]. Although these aspects are still not very well taken into account during the design process by CPD methods, more and more, stability and dynamical behavior of top-ranked protein designs are evaluated using different types of molecular modeling techniques dedicated to the study of molecular motions. Methods such as classical Molecular Dynamics (MD), or biased Molecular Dynamics (Steered or Targeted Molecular Dynamics (SMD, TMD) [122], Random Accelerating Molecular Dynamics (RAMD) [123]) and also Molecular Robotics techniques [30] have been widely used to check protein stability before experimental construction or after to verify reasons of failures, but also to identify key amino acid residues located farther away from the active site but still playing a role in protein activity or substrate recognition.

Finally, the importance of considering an ensemble of conformations instead of the only most stable conformation in order to achieve binding have also been well-established for non-covalent complexes [69]. Indeed, it is worth noting that non-covalent binding is achieved by a *thermodynamic ensemble of conformations*. Hence, in order to account for such fundamental ground, rotameric conformation ensemble-based scoring approaches such as K^* have been introduced [70]. This also incorporates to some extend the dynamical aspect of binding as several conformations contribute to the score. Donald and coworkers applied the K^* ranking approach to successfully redesign the phenylalanine adenylation domain (GrsA-PheA) with the goal of performing the same chemical reaction on a set of non-cognate substrates including tyrosine, arginine, glutamine, lysine, asparagine and leucine [71]. Experimental results on top ranked variants exhibited a specificity improvement toward leucine, arginine, glutamate, lysine, and aspartate and ultimately, a specificity switch from phenylalanine to leucine was achieved for several predicted mutants. In order to consider the additional macromolecular flexibility brought by the introduced mutation, this study included an additional step of *in silico* directed evolution to predict additional mutations distal to the active site that improve the fitness of the mutants.

1.3 Concluding remarks, new trends and challenges ahead

We conclude this brief overview of methodological advances and selected outstanding results accomplished in the CPD field by recalling some important challenges that the CPD technology has to face in the near future in order to produce more accurate results.

1.3.1 Search space modeling

Regarding the modeling aspect, although the formulation of the CPD problem is based on the use of a protein scaffold during the design process, the inherent dynamical property of the molecular system is still poorly taken into account. In particular, it is crucial to integrate some of the flexibility of the polypeptide chain in the design process in order to alleviate the sensitivity of energy functions and improve accuracy of the predictions. However, the introduction of these additional degrees of freedom increases tremendously the complexity of the search space, what remains very difficult to handle efficiently.

In addition to the flexibility of the main chain, a discrete representation of the side chain degrees of freedom alone is not sufficient to properly produce physically meaningful low energy conformations. Indeed, the discretization of the search space is an important basis of the CPD that enormously reduces the size of the combinatorial problem. However, it introduces critical biases that also have to be carefully corrected for the search to be more accurate.

To address this issue, two current solutions in use are the introduction of additional minimization of discrete states within a defined voxel in dihedral angle space along with the development of energy functions less sensitive to small geometric variations due to the discreteness of this geometrical space [24], [124].

1.3.2 Energy and objective functions

Considering energy and objective functions, critically, the evaluation of the thermodynamic stability of proteins requires the definition of the unfolded state which has no counterpart in 3D structure databases. Thus, *in silico* models are usually built to fulfill this need and it is not obvious to know what an arbitrary protein would look like in the unfolded state even though an important assumption is that only very local interactions are allowed in this state. In order to design stable proteins, many studies in the field attempt to take advantage of structural features observed in protein structures. Examples are the burial of hydrophobic residues and the exposure of polar ones. However, the importance of buried polar residues in protein core have been highlighted by Mayo and coworkers [125]. The study exhibited a correlation between the number of core polar residues and the size of proteins from a survey of 263 globular proteins. It exemplified the importance of intramolecular H-bonds for buried polar atoms, as previously revealed by McDonald and Thornton [126]. These interactions compensate for the unfavorable burial of polar amino acids. An elegant work have been initiated by Mayo and coworkers in order to account for such aspects by placing additional constraints on the introduction of polar residues in proteins core in contrast to the exclusion of polar residue from core design as widely applied [127]. Hydrogen bonding rules were generated for the introduction of polar and charged amino acids in proteins core [127]. These rules stem from statistical analysis of protein structures and define a minimum number of hydrogen bonds required for each polar side-chain. In the test case of the thioredoxin core design, this method led to improved thermodynamic stability in comparison to both the wild-type enzyme and the “no polar” strategy (where no polar amino acid is allowed in the protein core). More specifically for the recognition of small molecules, Hellinga and co-workers have successfully used hydrogen bonding satisfaction requirements to design soluble receptors

that bind trinitrotoluene, L-lactate or serotonin with high selectivity and affinity [117]. Their rule requires that donors and acceptors in the ligand must have a partner in the protein, thus controlling the introduction of polar amino acids in the vicinity of the ligand. A good understanding of such key features and their incorporation in objective functions may enhance the design of active sites, thermodynamic stability and structural specificity.

Fundamentally, the solvent is as important as the protein and its partners for the design to be realistic. However, its explicit consideration in CPD is intractable wherein variation is sequence is considered in addition to the already challenging size of the conformation space at fixed sequence. Even the most accurate implicit representation of the solvent [60], [128] is impractical in CPD. Both physics-based and knowledge-based terms are designed in order to meet the requirement of accurate but yet fast to compute the solvent contribution. There is still a room for theoretical advances in the implicit representation of such effect along with algorithmic development for effective calculations. In addition, because the implicit representation of the solvent is often inherently dependent on the coordinate of the whole macromolecular system, another question which remains unsolved is the building of accurate pairwise approximation of its representation [60].

More generally, the development of energy functions often involves the incorporation of additional terms in order to account for some experimental observation in both designed systems and 3D crystal structures of naturally occurring proteins. Thus, from a physico-chemical point of view, all the terms are not always independent and weights are derived from training set (3D structures) in order to find a good balance. Handling the tradeoff of such cross-terms also remains an unsolved issue [46]. On the same trend, the engineering of biological functions involves concurrent objectives that have to be balanced in terms or physico-chemical properties and function. Multiple objective function design methods, first developed in the field of social sciences and economics are finding application in this context [74]. However, further studies are still needed in order to draw conclusions of general usage for both the understanding and the tailoring of biochemical functions.

1.3.3 Imperfection of predictions

Ultimately, because approximations are made in the CPD modeling, both manual inspection and correction of the outputs of the optimization are often performed as well as *in silico* refinement of the CPD predictions using more accurate but also more costly methods on some selected designs. Manual corrections include reversion of some mutations not adequate. Recent studies are automating these tasks as part of the CPD experiment [129]. Also, *in silico* experiments such as MD simulations in the most efficient implicit solvent model as well as in explicit solvent are performed in order to assess the quality of the predictions. In addition, when designing for the binding of a small molecule, short MD simulations gives insightful results with respect to conformational equilibrium state of the mutants. It also allows the accurate prediction of binding free energies [130]. On the same trend, a key element to the maturation of the CPD methodologies is feedback from experimental validation as both successful and failed designs can still bring important information regarding the aspects to improve in the models and methods [17], [131].

1.4 REFERENCES

[1] J. H. Hurley, W. A. Baase, and B. W. Matthews, "Design and structural analysis of alternative hydrophobic core packing arrangements in bacteriophage T4 lysozyme.," *J. Mol. Biol.*, vol. 224, no. 4, pp. 1143–59, Apr. 1992.

[2] P. B. Harbury, B. Tidor, and P. S. Kim, "Repacking protein cores with backbone freedom: structure prediction for coiled coils.," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 92, no. 18, pp. 8408–12, Aug. 1995.

[3] J. R. Desjarlais and T. M. Handel, "De novo design of the hydrophobic cores of proteins.," *Protein Sci.*, vol. 4, no. 10, pp. 2006–18, Oct. 1995.

[4] S. F. Betz and W. F. DeGrado, "Controlling topology and native-like behavior of de novo-designed peptides: design and characterization of antiparallel four-stranded coiled coils.," *Biochemistry*, vol. 35, no. 21, pp. 6955–62, May 1996.

[5] B. I. Dahiyat and S. L. Mayo, "Protein design automation.," *Protein Sci.*, vol. 5, no. 5, pp. 895–903, 1996.

[6] B. I. Dahiyat, C. a Sarisky, and S. L. Mayo, "De novo protein design: towards fully automated sequence selection.," *J. Mol. Biol.*, vol. 273, no. 4, pp. 789–796, 1997.

[7] B. Kuhlman, G. Dantas, G. C. Ireton, G. Varani, B. L. Stoddard, and D. Baker, "Design of a novel globular protein fold with atomic-level accuracy.," *Science*, vol. 302, no. 5649, pp. 1364–8, Nov. 2003.

[8] P. Luo, R. J. Hayes, C. Chan, D. M. Stark, M. Y. Hwang, J. M. Jacinto, P. Juvvadi, H. S. Chung, A. Kundu, M. L. Ary, and B. I. Dahiyat, "Development of a cytokine analog with enhanced stability using computational ultrahigh throughput screening.," *Protein Sci.*, vol. 11, no. 5, pp. 1218–26, May 2002.

[9] L. A. Clark, P. A. Boriack-Sjodin, J. Eldredge, C. Fitch, B. Friedman, K. J. M. Hanf, M. Jarpe, S. F. Liparoto, Y. Li, A. Lugovskoy, S. Miller, M. Rushe, W. Sherman, K. Simon, and H. Van Vlijmen, "Affinity enhancement of an in vivo matured therapeutic antibody using structure-based computational design.," *Protein Sci.*, vol. 15, no. 5, pp. 949–60, May 2006.

[10] S. J. Fleishman, T. A. Whitehead, D. C. Ekiert, C. Dreyfus, J. E. Corn, E.-M. Strauch, I. A. Wilson, and D. Baker, "Computational design of proteins targeting the conserved stem region of influenza hemagglutinin.," *Science*, vol. 332, no. 6031, pp. 816–21, May 2011.

[11] E. Yosef, R. Politi, M. H. Choi, and J. M. Shifman, "Computational design of calmodulin mutants with up to 900-fold increase in binding specificity.," *J. Mol. Biol.*, vol. 385, no. 5, pp. 1470–80, Feb. 2009.

[12] B. S. Der, M. Machius, M. J. Miley, J. L. Mills, T. Szyperski, and B. Kuhlman, "Metal-mediated affinity and orientation specificity in a computationally designed protein homodimer.," *J. Am. Chem. Soc.*, vol. 134, no. 1, pp. 375–85, Jan. 2012.

[13] D. Röthlisberger, O. Khersonsky, A. M. Wollacott, L. Jiang, J. DeChancie, J. Betker, J. L. Gallaher, E. a Althoff, A. Zanghellini, O. Dym, S. Albeck, K. N. Houk, D. S. Tawfik, and D. Baker, "Kemp elimination catalysts by computational enzyme design.," *Nature*, vol. 453, no. 7192, pp. 190–5, May 2008.

[14] L. Jiang, E. a Althoff, F. R. Clemente, L. Doyle, D. Röthlisberger, A. Zanghellini, J. L. Gallaher, J. L. Betker, F. Tanaka, C. F. B. Iii, D. Hilvert, K. N. Houk, B. L. Stoddard, D. Baker, and C. F. Barbas, "De novo computational design of retro-aldol enzymes.," *Science*, vol. 319, no. 5868, pp. 203–277, Mar. 2008.

[15] J. B. Siegel, A. Zanghellini, H. M. Lovick, G. Kiss, A. R. Lambert, J. L. St Clair, J. L. Gallaher, D. Hilvert, M. H. Gelb, B. L. Stoddard, K. N. Houk, F. E. Michael, and D. Baker, "Computational design of an enzyme catalyst for a stereoselective bimolecular Diels-Alder reaction.," *Science*, vol. 329, no. 5989, pp. 309–13, Jul. 2010.

[16] L. Serrano, A. G. Day, and A. R. Fersht, "Step-wise mutation of barnase to binase. A procedure for engineering increased stability of proteins and an experimental analysis of the evolution of protein stability.," *J. Mol. Biol.*, vol. 233, no. 2, pp. 305–12, Sep. 1993.

[17] P. B. Stranges and B. Kuhlman, "A comparison of successful and failed protein interface designs highlights the challenges of designing buried hydrogen bonds.," *Protein Sci.*, vol. 22, no. 1, pp. 74–82, Jan. 2013.

[18] F. C. Bernstein, T. F. Koetzle, G. J. Williams, E. F. Meyer, M. D. Brice, J. R. Rodgers, O. Kennard, T. Shimanouchi, and M. Tasumi, "The Protein Data Bank. A computer-based archival file for macromolecular structures.," *Eur. J. Biochem.*, vol. 80, pp. 319–324, 1977.

[19] A. G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia, "SCOP: a structural classification of proteins database for the investigation of sequences and structures.," *J. Mol. Biol.*, vol. 247, no. 4, pp. 536–40, Apr. 1995.

[20] A. Zanghellini, L. I. N. Jiang, A. M. Wollacott, G. Cheng, J. Meiler, E. A. Althoff, and D. Ro, "New algorithms and an in silico benchmark for computational enzyme design," no. Russell 1998, pp. 2785–2794, 2006.

[21] B. Stevens, R. Lilien, and I. Georgiev, “Redesigning the PheA domain of gramicidin synthetase leads to a new understanding of the enzyme’s mechanism and selectivity,” *Biochemistry*, 2006.

[22] N. Koga, R. Tatsumi-Koga, G. Liu, R. Xiao, T. B. Acton, G. T. Montelione, and D. Baker, “Principles for designing ideal protein structures.,” *Nature*, vol. 491, no. 7423, pp. 222–7, Nov. 2012.

[23] J. W. Ponder and F. M. Richards, “Tertiary templates for proteins. Use of packing criteria in the enumeration of allowed sequences for different structural classes.,” *J. Mol. Biol.*, vol. 193, no. 4, pp. 775–91, Feb. 1987.

[24] P. Gainza, K. E. Roberts, and B. R. Donald, “Protein design using continuous rotamers,” *PLoS Comput. Biol.*, vol. 8, no. 1, p. e1002335, Jan. 2012.

[25] P. B. Harbury, J. J. Plecs, B. Tidor, T. Alber, and P. S. Kim, “High-resolution protein design with backbone freedom.,” *Science*, vol. 282, no. 5393, pp. 1462–1467, Nov. 1998.

[26] J. R. Desjarlais and T. M. Handel, “Side-chain and backbone flexibility in protein core design.,” *J. Mol. Biol.*, vol. 290, no. 1, pp. 305–18, Jul. 1999.

[27] I. Georgiev and B. Donald, “Dead-end elimination with backbone flexibility,” *Bioinformatics*, vol. 23, no. 13, pp. i185–94, Jul. 2007.

[28] H. K. Fung, M. S. Taylor, and C. A. Floudas, “Novel formulations for the sequence selection problem in de novo protein design with flexible templates,” *Optimization Methods and Software*, vol. 22. pp. 51–71, 2007.

[29] H. K. Fung, C. a Floudas, M. S. Taylor, L. Zhang, and D. Morikis, “Toward full-sequence de novo protein design with flexible templates for human beta-defensin-2.,” *Biophys. J.*, vol. 94, no. 2, pp. 584–99, Jan. 2008.

[30] J. Cortés, T. Siméon, M. Remaud-Siméon, and V. Tran, “Geometric algorithms for the conformational analysis of long protein loops.,” *J. Comput. Chem.*, vol. 25, no. 7, pp. 956–67, May 2004.

[31] E. A. Coutsias, C. Seok, M. P. Jacobson, and K. E. N. A. Dill, “A Kinematic View of Loop Closure,” 2004.

[32] A. Lee, I. Streinu, and O. Brock, “A methodology for efficiently sampling the conformation space of molecular structures.,” *Phys. Biol.*, vol. 2, no. 4, pp. S108–15, Nov. 2005.

[33] A. A. Canutescu and R. L. Dunbrack, “Cyclic coordinate descent: A robotics algorithm for protein loop closure.,” *Protein Sci.*, vol. 12, no. 5, pp. 963–72, May 2003.

[34] X. Hu, H. Wang, H. Ke, and B. Kuhlman, “High-resolution design of a protein loop.,” *Proc. Natl. Acad. Sci. U. S. A.*, vol. 104, no. 45, pp. 17668–73, Nov. 2007.

[35] P. M. Murphy, J. M. Bolduc, J. L. Gallaher, B. L. Stoddard, and D. Baker, “Alteration of enzyme specificity by computational loop remodeling and design.,” *Proc. Natl. Acad. Sci. U. S. A.*, vol. 106, no. 23, pp. 9215–20, Jun. 2009.

[36] I. W. Davis, W. B. A. Iii, D. C. Richardson, J. S. Richardson, and W. B. Arendall, “The backrub motion: how protein backbone shrugs when a sidechain dances.,” *Structure*, vol. 14, no. 2, pp. 265–274, Feb. 2006.

[37] I. Georgiev, D. Keedy, J. S. Richardson, D. C. Richardson, and B. R. Donald, “Algorithm for backrub motions in protein design.,” *Bioinformatics*, vol. 24, no. 13, pp. i196–204, Jul. 2008.

[38] C. A. Smith and T. Kortemme, “Backrub-like backbone simulation recapitulates natural protein conformational variability and improves mutant side-chain prediction.,” *J. Mol. Biol.*, vol. 380, no. 4, pp. 742–56, Jul. 2008.

[39] M. A. Hallen, D. A. Keedy, and B. R. Donald, “Dead-end elimination with perturbations (DEEPer): a provable protein design algorithm with continuous sidechain and backbone flexibility.,” *Proteins*, vol. 81, no. 1, pp. 18–39, Jan. 2013.

[40] D. A. Keedy, I. Georgiev, E. B. Triplett, B. R. Donald, D. C. Richardson, and J. S. Richardson, “The role of local backrub motions in evolved and designed mutations.,” *PLoS Comput. Biol.*, vol. 8, no. 8, p. e1002629, Jan. 2012.

[41] M. a Hallen, D. A. Keedy, and B. R. Donald, “Dead-end elimination with perturbations (DEEPer): A provable protein design algorithm with continuous sidechain and backbone flexibility,” *Proteins Struct. Funct. Bioinforma.*, vol. 81, no. 1, pp. 18–39, Jan. 2013.

[42] L. L. Looger, M. A. Dwyer, J. J. Smith, and H. W. Hellinga, “Computational design of receptor and sensor proteins with novel functions.,” *Nature*, vol. 423, no. 6936, pp. 185–90, May 2003.

[43] J. K. Lassila, H. K. Privett, B. D. Allen, and S. L. Mayo, “Combinatorial methods for small-molecule placement in computational enzyme design,” vol. 103, no. 45, 2006.

[44] M. C. Saraf, G. L. Moore, N. M. Goodey, V. Y. Cao, S. J. Benkovic, and C. D. Maranas, “IPRO: an iterative computational protein library redesign and optimization procedure.,” *Biophys. J.*, vol. 90, no. 11, pp. 4167–80, Jun. 2006.

[45] K. T. Simons, I. Ruczinski, C. Kooperberg, B. a Fox, C. Bystroff, and D. Baker, “Improved recognition of native-like protein structures using a combination of sequence-dependent and sequence-independent features of proteins.,” *Proteins*, vol. 34, no. 1, pp. 82–95, Jan. 1999.

[46] Y. Huang and C. Bystroff, "Exploring objective functions and cross-terms in the optimization of an energy function for protein design," in *Proceedings of the ACM Conference on Bioinformatics, Computational Biology and Biomedicine - BCB '12*, 2012, pp. 155–162.

[47] Z. Li, Y. Yang, J. Zhan, L. Dai, and Y. Zhou, "Energy functions in de novo protein design: current challenges and future prospects.," *Annu. Rev. Biophys.*, vol. 42, pp. 315–35, Jan. 2013.

[48] B. K. Shoichet, W. A. Baase, R. Kuroki, and B. W. Matthews, "A relationship between protein stability and protein function.," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 92, no. 2, pp. 452–6, Jan. 1995.

[49] C. Wilson, J. E. Mace, and D. A. Agard, "Computational method for the design of enzymes with altered substrate specificity.," *J. Mol. Biol.*, vol. 220, no. 2, pp. 495–506, Jul. 1991.

[50] K. T. Simons, C. Kooperberg, E. Huang, and D. Baker, "Assembly of protein tertiary structures from fragments with similar local sequences using simulated annealing and Bayesian scoring functions.," *J. Mol. Biol.*, vol. 268, no. 1, pp. 209–25, Apr. 1997.

[51] B. I. Dahiyat and S. L. Mayo, "Probing the role of packing specificity in protein design.," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 94, no. 19, pp. 10172–7, Sep. 1997.

[52] B. Kuhlman and D. Baker, "Native protein sequences are close to optimal for their structures.," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 97, no. 19, pp. 10383–10388, 2000.

[53] N. Pokala and T. T. M. Handel, "Energy functions for protein design: adjustment with protein-protein complex affinities, models for the unfolded state, and negative design of solubility and specificity.," *J. Mol. Biol.*, vol. 347, no. 1, pp. 203–277, Mar. 2005.

[54] T. Kortemme, A. V. Morozov, and D. Baker, "An Orientation-dependent Hydrogen Bonding Potential Improves Prediction of Specificity and Structure for Proteins and Protein–Protein Complexes," *J. Mol. Biol.*, vol. 326, no. 4, pp. 1239–1259, Feb. 2003.

[55] T. E. Cheatham and P. A. Kollman, "Molecular dynamics simulation of nucleic acids.," *Annu. Rev. Phys. Chem.*, vol. 51, pp. 435–71, Jan. 2000.

[56] M. Karplus and J. A. McCammon, "Molecular dynamics simulations of biomolecules.," *Nat. Struct. Biol.*, vol. 9, no. 9, pp. 646–52, Oct. 2002.

[57] F. Fogolari, A. Brigo, and H. Molinari, "The Poisson-Boltzmann equation for biomolecular electrostatics: a tool for structural biology.," *J. Mol. Recognit.*, vol. 15, no. 6, pp. 377–92.

[58] L. Wesson and D. Eisenberg, "Atomic solvation parameters applied to molecular dynamics of proteins in solution.," *Protein Sci.*, vol. 1, no. 2, pp. 227–35, Mar. 1992.

[59] a G. Street and S. L. Mayo, "Pairwise calculation of protein solvent-accessible surface areas.," *Fold. Des.*, vol. 3, no. 4, pp. 253–8, Jan. 1998.

[60] T. Gaillard and T. Simonson, "Pairwise decomposition of an MMGBSA energy function for computational protein design.," *J. Comput. Chem.*, May 2014.

[61] J. J. Havranek and P. B. Harbury, "Tanford-Kirkwood electrostatics for protein modeling.," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 96, no. 20, pp. 11145–50, Sep. 1999.

[62] T. Lazaridis and M. Karplus, "Effective energy function for proteins in solution.," *Proteins*, vol. 35, no. 2, pp. 133–52, May 1999.

[63] L. Wernisch, S. Hery, and S. J. Wodak, "Automatic protein design with all atom force-fields by exact and heuristic optimization.," *J. Mol. Biol.*, vol. 301, no. 3, pp. 713–36, Aug. 2000.

[64] S. M. Lippow, K. D. Wittrup, and B. Tidor, "Computational design of antibody affinity improvement beyond in vitro maturation," *Nat. Biotech.*, vol. 25, pp. 1171–1176, 2007.

[65] R. Guerois, J. E. Nielsen, and L. Serrano, "Predicting changes in the stability of proteins and protein complexes: a study of more than 1000 mutations.," *J. Mol. Biol.*, vol. 320, no. 2, pp. 369–87, Jul. 2002.

[66] T. P. Creamer, R. Srinivasan, and G. D. Rose, "Modeling unfolded states of peptides and proteins.," *Biochemistry*, vol. 34, no. 50, pp. 16245–50, Dec. 1995.

[67] N. Tokuriki, F. Stricher, L. Serrano, and D. S. Tawfik, "How protein stability and new functions trade off.," *PLoS Comput. Biol.*, vol. 4, no. 2, p. e1000002, Feb. 2008.

[68] M. P. Frushicheva, M. J. Mills, P. Schopf, M. K. Singh, R. B. Prasad, and A. Warshel, "Computer aided enzyme design and catalytic concepts," *Curr. Opin. Chem. Biol.*, vol. 21, pp. 56–62, Aug. 2014.

[69] M. K. Gilson, J. A. Given, B. L. Bush, and J. A. McCammon, "The statistical-thermodynamic basis for computation of binding affinities: a critical review.," *Biophys. J.*, vol. 72, no. 3, pp. 1047–69, Mar. 1997.

[70] R. H. Lilien, B. W. Stevens, A. C. Anderson, and B. R. Donald, "A novel ensemble-based scoring and search algorithm for protein redesign and its application to modify the substrate specificity of the gramicidin synthetase a phenylalanine adenylation enzyme.," *J. Comput. Biol.*, vol. 12, no. 6, pp. 740–61, Jan. 2005.

[71] C. Chen, I. Georgiev, A. C. Anderson, and B. R. Donald, "Computational structure-based redesign of enzyme activity," vol. 106, no. 10, 2009.

[72] J. D. Bloom, S. T. Labthavikul, C. R. Otey, and F. H. Arnold, "Protein stability promotes evolvability.," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 103, no. 15, pp. 5869–74, May 2006.

[73] N. Tokuriki, F. Stricher, J. Schymkowitz, L. Serrano, and D. S. Tawfik, "The stability effects of protein mutations appear to be universally distributed.," *J. Mol. Biol.*, vol. 369, no. 5, pp. 1318–32, Jun. 2007.

[74] M. Suárez, P. Tortosa, J. Carrera, and A. Jaramillo, "Pareto Optimization in Computational Protein Design with Multiple Objectives," vol. 12, 2008.

[75] M. Suarez, P. Tortosa, M. M. Garcia-Mira, D. Rodríguez-Larrea, R. Godoy-Ruiz, B. Ibarra-Molero, J. M. Sanchez-Ruiz, and A. Jaramillo, "Using multi-objective computational design to extend protein promiscuity.," *Biophys. Chem.*, vol. 147, no. 1–2, pp. 13–9, Mar. 2010.

[76] N. A. Pierce and E. Winfree, "Protein design is NP-hard.," *Protein Eng.*, vol. 15, no. 10, pp. 779–782, Oct. 2002.

[77] B. Chazelle, C. Kingsford, and M. Singh, "A Semidefinite Programming Approach to Side Chain Positioning with New Rounding Strategies," *INFORMS J. Comput.*, vol. 16, no. 4, pp. 380–392, Nov. 2004.

[78] C. a Voigt, D. B. Gordon, and S. L. Mayo, "Trading accuracy for speed: A quantitative comparison of search algorithms in protein sequence design.," *J. Mol. Biol.*, vol. 299, no. 3, pp. 789–803, Jun. 2000.

[79] J. Desmet, M. De Maeyer, B. Hazes, and I. Lasters, "The dead-end elimination theorem and its use in protein side-chain positioning.," *Nature*, vol. 356, no. 6369, pp. 539–542, 1992.

[80] A. R. Leach and A. P. Lemon, "Exploring the conformational space of protein side chains using dead-end elimination and the A* algorithm.," *Proteins*, vol. 33, no. 2, pp. 227–239, 1998.

[81] C. L. Kingsford, B. Chazelle, and M. Singh, "Solving and analyzing side-chain positioning problems using linear and integer programming.," *Bioinformatics*, vol. 21, no. 7, pp. 1028–1036, Apr. 2005.

[82] A. Leaver-Fay, B. Kuhlman, and J. Snoeyink, "An adaptive dynamic programming algorithm for the side chain placement problem.," *Pac. Symp. Biocomput.*, pp. 16–27, 2005.

[83] A. Roitberg and R. Elber, "Modeling side chains in peptides and proteins: Application of the locally enhanced sampling and the simulated annealing methods to find minimum energy conformations," *J. Chem. Phys.*, vol. 95, no. 12, p. 9277, 1991.

[84] H. Kono and J. Doi, "Energy minimization method using automata network for sequence and side-chain conformation prediction from given backbone geometry.," *Proteins*, vol. 19, no. 3, pp. 244–55, Jul. 1994.

[85] M. Vásquez, "Modeling side-chain conformation," *Curr. Opin. Struct. Biol.*, vol. 6, no. 2, pp. 217–221, Apr. 1996.

[86] M. Delarue and P. Koehl, "The inverse protein folding problem: self consistent mean field optimisation of a structure specific mutation matrix.," *Pac. Symp. Biocomput.*, pp. 109–21, Jan. 1997.

[87] C. Lee, "Predicting protein mutant energetics by self-consistent ensemble optimization," *J. Mol. Biol.*, 1994.

[88] Z. Li and H. a Scheraga, "Monte Carlo-minimization approach to the multiple-minima problem in protein folding.," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 84, no. 19, pp. 6611–5, Oct. 1987.

[89] S. Polydorides, N. Amara, C. Aubard, P. Plateau, T. Simonson, and G. Archontis, "Computational protein design with a generalized Born solvent model: application to Asparaginyl-tRNA synthetase.," *Proteins*, vol. 79, no. 12, pp. 3448–68, Dec. 2011.

[90] D. T. Jones, "De novo protein design using pairwise potentials and a genetic algorithm.," *Protein Sci.*, vol. 3, no. 4, pp. 567–74, Apr. 1994.

[91] J. Desjarlais and T. Handel, "De novo design of the hydrophobic cores of proteins," *Protein Sci.*, vol. 4, no. 10, pp. 2006–18, Oct. 1995.

[92] J. Desmet, J. Spriet, and I. Lasters, "Fast and accurate side-chain topology and energy refinement (FASTER) as a new method for protein structure optimization.," *Proteins*, vol. 48, no. 1, pp. 31–43, 2002.

[93] B. D. Allen and S. L. Mayo, "Dramatic performance enhancements for the FASTER optimization algorithm.," *J. Comput. Chem.*, vol. 27, no. 10, pp. 1071–5, Jul. 2006.

[94] R. F. Goldstein, "Efficient rotamer elimination applied to protein side-chains and related spin glasses.," *Biophys. J.*, vol. 66, no. 5, pp. 1335–1340, 1994.

[95] N. A. Pierce, J. A. Spriet, J. Desmet, and S. L. Mayo, "Conformational splitting: A more powerful criterion for dead-end elimination," *J. Comput. Chem.*, vol. 21, no. 11, pp. 999–1009, 2000.

[96] I. Georgiev, R. H. Lilien, and B. R. Donald, "Improved Pruning algorithms and Divide-and-Conquer strategies for Dead-End Elimination, with application to protein design.," *Bioinformatics*, vol. 22, no. 14, pp. 530–545, Jul. 2006.

[97] I. Georgiev, R. H. Lilien, and B. R. Donald, "The minimized dead-end elimination criterion and its application to protein redesign in a hybrid scoring and search algorithm for computing partition functions over molecular ensembles.," *J. Comput. Chem.*, vol. 29, no. 10, pp. 1527–42, Jul. 2008.

[98] D. B. Gordon, G. K. Hom, S. L. Mayo, and N. a Pierce, "Exact rotamer optimization for protein design.," *J. Comput. Chem.*, vol. 24, no. 2, pp. 232–43, Jan. 2003.

[99] P. Hart, N. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. 4, no. 2, pp. 100–107, Jul. 1968.

[100] P. Gainza, K. E. Roberts, I. Georgiev, R. H. Lilien, D. A. Keedy, C.-Y. Chen, F. Reza, A. C. Anderson, D. C. Richardson, J. S. Richardson, and others, "OSPREY: Protein design with ensembles, flexibility, and provable algorithms," *Methods Enzym.*, 2012.

[101] A. Leaver-Fay, M. Tyka, S. M. Lewis, O. F. Lange, J. Thompson, R. Jacak, K. Kaufman, P. D. Renfrew, C. A. Smith, W. Sheffler, I. W. Davis, S. Cooper, A. Treuille, D. J. Mandell, F. Richter, Y.-E. A. Ban, S. J. Fleishman, J. E. Corn, D. E. Kim, S. Lyskov, M. Berrondo, S. Mentzer, Z. Popović, J. J. Havranek, J. Karanicolas, R. Das, J. Meiler, T. Kortemme, J. J. Gray, B. Kuhlman, D. Baker, and P. Bradley, "ROSETTA3: an object-oriented software suite for the simulation and design of macromolecules.," *Methods Enzymol.*, vol. 487, pp. 545–74, Jan. 2011.

[102] T. Simonson, T. Gaillard, D. Mignon, M. Schmidt am Busch, A. Lopes, N. Amara, S. Polydorides, A. Sedano, K. Druart, and G. Archontis, "Computational protein design: the Proteus software and selected applications.," *J. Comput. Chem.*, vol. 34, no. 28, pp. 2472–84, Oct. 2013.

[103] D. N. Bolon and S. L. Mayo, "Enzyme-like proteins by computational design.," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 98, no. 25, pp. 14274–9, Dec. 2001.

[104] M. Suárez, P. Tortosa, and A. Jaramillo, "PROTDES: CHARMM toolbox for computational protein design.," *Syst. Synth. Biol.*, vol. 2, no. 3–4, pp. 105–13, Dec. 2008.

[105] B. J. Hillier, H. M. Rodriguez, and L. M. Gregoret, "Coupling protein stability and protein function in Escherichia coli CspA.," *Fold. Des.*, vol. 3, no. 2, pp. 87–93, Jan. 1998.

[106] R. Chakrabarti, A. M. Klibanov, and R. A. Friesner, "Sequence optimization and designability of enzyme active sites.," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 102, no. 34, pp. 12035–40, Aug. 2005.

[107] N. P. King, J. B. Bale, W. Sheffler, D. E. McNamara, S. Gonen, T. Gonen, T. O. Yeates, and D. Baker, "Accurate design of co-assembling multi-component protein nanomaterials," *Nature*, May 2014.

[108] M. Guharoy and P. Chakrabarti, "Secondary structure based analysis and classification of biological interfaces: identification of binding motifs in protein-protein interactions.," *Bioinformatics*, vol. 23, no. 15, pp. 1909–18, Aug. 2007.

[109] Y. G. Ni, S. Di Marco, J. H. Condra, L. B. Peterson, W. Wang, F. Wang, S. Pandit, H. A. Hammond, R. Rosa, R. T. Cummings, D. D. Wood, X. Liu, M. J. Bottomley, X. Shen, R. M. Cubbon, S. Wang, D. G. Johns, C. Volpari, L. Hamuro, J. Chin, L. Huang, J. Z. Zhao, S. Vitelli, P. Haytko, D. Wisniewski, L. J. Mitnaul, C. P. Sparrow, B. Hubbard, A. Carfí, and A. Sitlani, "A PCSK9-binding antibody that structurally mimics the EGF(A) domain of LDL-receptor reduces LDL cholesterol in vivo.," *J. Lipid Res.*, vol. 52, no. 1, pp. 78–86, Jan. 2011.

[110] P. B. Stranges, M. Machius, M. J. Miley, A. Tripathy, and B. Kuhlman, "Computational design of a symmetric homodimer using β-strand assembly.," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 108, no. 51, pp. 20562–7, Dec. 2011.

[111] J. Karanicolas, J. E. Corn, I. Chen, L. a Joachimiak, O. Dym, S. H. Peck, S. Albeck, T. Unger, W. Hu, G. Liu, S. Delbecq, G. T. Montelione, C. P. Spiegel, D. R. Liu, and D. Baker, "A de novo protein binding pair by computational design and directed evolution.," *Mol. Cell*, vol. 42, no. 2, pp. 250–60, Apr. 2011.

[112] H. W. Hellinga and F. M. Richards, "Construction of new ligand binding sites in proteins of known structure. I. Computer-aided modeling of sites with pre-defined geometry.," *J. Mol. Biol.*, vol. 222, no. 3, pp. 763–85, Dec. 1991.

[113] H. W. Hellinga and F. M. Richards, "Construction of new ligand binding sites in proteins of known structure. I. Computer-aided modeling of sites with pre-defined geometry.," *J. Mol. Biol.*, vol. 222, no. 3, pp. 763–85, Dec. 1991.

[114] A. L. Pinto, H. W. Hellinga, and J. P. Caradonna, "Construction of a catalytically active iron superoxide dismutase by rational protein design.," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 94, no. 11, pp. 5562–7, May 1997.

[115] D. E. Benson, M. S. Wisz, W. Liu, and H. W. Hellinga, "Construction of a novel redox protein by rational design: conversion of a disulfide bridge into a mononuclear iron-sulfur center.," *Biochemistry*, vol. 37, no. 20, pp. 7070–6, May 1998.

[116] J. S. Marvin and H. W. Hellinga, "Manipulation of ligand binding affinity by exploitation of conformational coupling.," *Nat. Struct. Biol.*, vol. 8, no. 9, pp. 795–798, 2001.

[117] L. L. Looger, M. a Dwyer, J. J. Smith, and H. W. Hellinga, "Computational design of receptor and sensor proteins with novel functions.," *Nature*, vol. 423, no. 6936, pp. 185–190, 2003.

[118] J. Kaplan and W. F. DeGrado, "De novo design of catalytic proteins.," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 101, no. 32, pp. 11566–70, Aug. 2004.

[119] A. Barrozo, R. Borstnar, G. Marloie, and S. C. L. Kamerlin, "Computational Protein Engineering: Bridging the Gap between Rational Design and Laboratory Evolution.," *Int. J. Mol. Sci.*, vol. 13, no. 10, pp. 12428–60, Jan. 2012.

[120] D. J. Tantillo, J. Chen, and K. N. Houk, "Theozymes and compuzymes: theoretical models for biological catalysis.," *Curr. Opin. Chem. Biol.*, vol. 2, no. 6, pp. 743–50, Dec. 1998.

[121] J. K. Lassila, "Conformational diversity and computational enzyme design," *Current Opinion in Chemical Biology*, vol. 14. pp. 676–682, 2010.

[122] J. S. Patel, A. Berteotti, S. Ronsisvalle, W. Rocchia, and A. Cavalli, "Steered molecular dynamics simulations for studying protein-ligand interaction in cyclin-dependent kinase 5.," *J. Chem. Inf. Model.*, vol. 54, no. 2, pp. 470–80, Feb. 2014.

[123] S. K. Lüdemann, V. Lounnas, and R. C. Wade, "How do substrates enter and products exit the buried active site of cytochrome P450cam? 1. Random expulsion molecular dynamics investigation of ligand access channels and mechanisms.," *J. Mol. Biol.*, vol. 303, no. 5, pp. 797–811, Nov. 2000.

[124] I. Georgiev, R. H. Lilien, and B. R. Donald, "Improved Pruning algorithms and Divide-and-Conquer strategies for Dead-End Elimination, with application to protein design.," *Bioinformatics*, vol. 22, no. 14, pp. e174–e183, Jul. 2006.

[125] D. N. Bolon and S. L. Mayo, "Polar residues in the protein core of Escherichia coli thioredoxin are important for fold specificity.," *Biochemistry*, vol. 40, no. 34, pp. 10047–53, Aug. 2001.

[126] I. K. McDonald and J. M. Thornton, "Satisfying hydrogen bonding potential in proteins.," *J. Mol. Biol.*, vol. 238, no. 5, pp. 777–93, May 1994.

[127] D. N. Bolon, J. S. Marcus, S. A. Ross, and S. L. Mayo, "Prudent modeling of core polar residues in computational protein design.," *J. Mol. Biol.*, vol. 329, no. 3, pp. 611–22, Jun. 2003.

[128] M. Feig, A. Onufriev, M. S. Lee, W. Im, D. a Case, and C. L. Brooks, "Performance comparison of generalized born and Poisson methods in the calculation of electrostatic solvation energies for protein structures.," *J. Comput. Chem.*, vol. 25, no. 2, pp. 265–84, Jan. 2004.

[129] L. G. Nivón, S. Bjelic, C. King, and D. Baker, "Automating human intuition for protein design.," *Proteins*, Oct. 2013.

[130] T. Simonson, G. Archontis, and M. Karplus, "Free energy simulations come of age: protein-ligand recognition.," *Acc. Chem. Res.*, vol. 35, no. 6, pp. 430–7, Jun. 2002.

[131] S. J. Fleishman, T. A. Whitehead, E.-M. Strauch, J. E. Corn, S. Qin, H.-X. Zhou, J. C. Mitchell, O. N. A. Demerdash, M. Takeda-Shitaka, G. Terashi, I. H. Moal, X. Li, P. A. Bates, M. Zacharias, H. Park, J. Ko, H. Lee, C. Seok, T. Bourquard, J. Bernauer, A. Poupon, J. Azé, S. Soner, S. K. Ovali, P. Ozbek, N. Ben Tal, T. Haliloglu, H. Hwang, T. Vreven, B. G. Pierce, Z. Weng, L. Pérez-Cano, C. Pons, J. Fernández-Recio, F. Jiang, F. Yang, X. Gong, L. Cao, X. Xu, B. Liu, P. Wang, C. Li, C. Wang, C. H. Robert, M. Guharoy, S. Liu, Y. Huang, L. Li, D. Guo, Y. Chen, Y. Xiao, N. London, Z. Itzhaki, O. Schueler-Furman, Y. Inbar, V. Potapov, M. Cohen, G. Schreiber, Y. Tsuchiya, E. Kanamori, D. M. Standley, H. Nakamura, K. Kinoshita, C. M. Driggers, R. G. Hall, J. L. Morgan, V. L. Hsu, J. Zhan, Y. Yang, Y. Zhou, P. L. Kastritis, A. M. J. J. Bonvin, W. Zhang, C. J. Camacho, K. P. Kilambi, A. Sircar, J. J. Gray, M. Ohue, N. Uchikoga, Y. Matsuzaki, T. Ishida, Y. Akiyama, R. Khashan, S. Bush, D. Fouches, A. Tropsha, J. Esquivel-Rodríguez, D. Kihara, P. B. Stranges, R. Jacak, B. Kuhlman, S.-Y. Huang, X. Zou, S. J. Wodak, J. Janin, and D. Baker, "Community-wide assessment of protein-interface modeling suggests improvements to design methodology.," *J. Mol. Biol.*, vol. 414, no. 2, pp. 289–302, Nov. 2011.

2 Cost Function Network optimization

Contents

2.1 Cost Function Networks 43
2.2 Local Consistencies 44
2.3 Weighted Constraint Satisfaction Problem solvers – Complete search 50
2.4 Alternative and complementary strategies 51
2.5 References 54

Cost Function Networks (CFN) offer a powerful framework to address various combinatorial optimization problems. In this chapter, we introduce some basic definitions and formalisms of the CFN framework with the purpose of providing fundamental grounds that will help to understand how CFN was used in Chapter 3 for the first time to model and solve Computational Protein Design (CPD) problems.

A brief formal definition of the CFN model and its underlying optimization problem is first given in Subsection 2.1. Solving a CFN optimization problem involves enforcing some local properties called local consistencies. These notions are introduced in Subsections 0 afterwards. In Subsection 0 we outline a general CFN solver. We conclude this chapter by defining some alternative search strategies and metaheuristics in Subsection 2.4.

2.1 Cost Function Networks

A Constraint Network is a mathematical model where a set of constraints are defined over a set of discrete variables. Each constraint restricts the permitted values for one or a subset of variables. The Constraint Satisfaction Problem (CSP) is to find a value for all variables that simultaneously satisfies all constraints (also called a solution). The CSP is NP-complete [1]. A *Cost Function Network* (CFN) extends the Constraint Network Framework by replacing constraints with cost functions [2], [3]. In a CFN, we are given a set of variables with an associated finite domain and a set of local cost functions (i.e., involving only a subset of all variables). The Weighted Constraint Satisfaction Problem (WCSP) is to find a value for all variables that minimizes the sum of all cost functions. CFNs have been used as a modeling framework for representing and solving various combinatorial optimization problems in many areas including bioinformatics and resource allocation [4]–[6].

Formally, a CFN P is a triple $P = (X, D, C)$ where $X = \{1,2,3,..,n\}$ is a set of n variables. Each variable $i \in X$ has a discrete domain $D_i \in$ D. In the triple, C is a set of local cost functions. Each cost function $c_S \in C$ is defined over a subset of variables $S \subseteq X$ (called its scope), has domain $\prod_{i \in S} D_i$ and takes its values in $\mathbb{N} \cup \{+\infty\}$. Forbidden value, pair and higher order joined assignments are represented by infinite costs called hard constraints and all cost functions must be non-negative. In practice, we may often know a "good" solution of cost k, making all solutions of cost above k uninteresting. All costs above k can then be considered as infinite. The projection of an assignment A on a set of variable S is an assignment of the variables S to their values in A. The cost of A for a local cost function is the value of the cost function for the projection of A to the scope (S) of the function. The global cost of A is the sum of the costs of A over all local cost functions. It is usually assumed that C contains one constant cost function, with an empty scope, denoted $c_\emptyset$. A CFN P defines a joint cost distribution over all the variables X defined by the cost of the assignments. Since all cost functions in a CFN are non-negative, the constant cost function $c_\emptyset \in C$ defines a lower bound on this joint cost distribution.

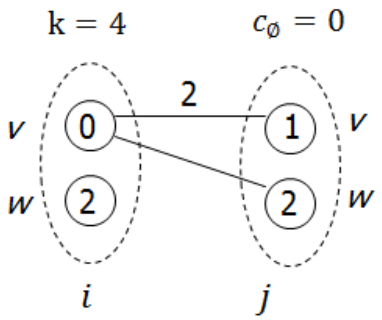


Fig 2-1 A toy example of CFN

A so-called microstructure representation of CFN as a graph for a toy example is given in **Fig 2-1**. It has two variables $\{i, j\}$ represented by dashed circles. The domain values are represented by small circles (vertices, $D_x = D_y = \{v, w\}$). The labels of the vertices are the corresponding unary costs. Edges represent binary terms; the corresponding label is the cost. For clarity, edges which have zero cost are not represented and the label is not written when the cost = 1. The initial upper bound of the problem is 4.

2.2 Local Consistencies

CFNs are simplified by enforcing a set of Local Consistency properties (LC). The enforcing algorithms (presented thereafter) take as input any CFN and perform a set of transformations that preserve equivalence of problems and yield a final equivalent problem that satisfies the CL property. Two CFNs defined other the same set of variables are equivalent if they define the same cost distribution on complete assignments [7]. These types of transformations are denoted Equivalence Preserving Transformations (EPT).

Hierarchically, the simplest notion of local consistency is the Node Consistency *(NC)* [7]. A CFN satisfies the NC property if two conditions hold for all variables. Precisely, for each $i \in X$: 1) $\exists\ a \in D_i$ such that $c_i(a) = 0$ and 2) $c_\emptyset + c_i(a) < k$ for all $a \in D_i$.

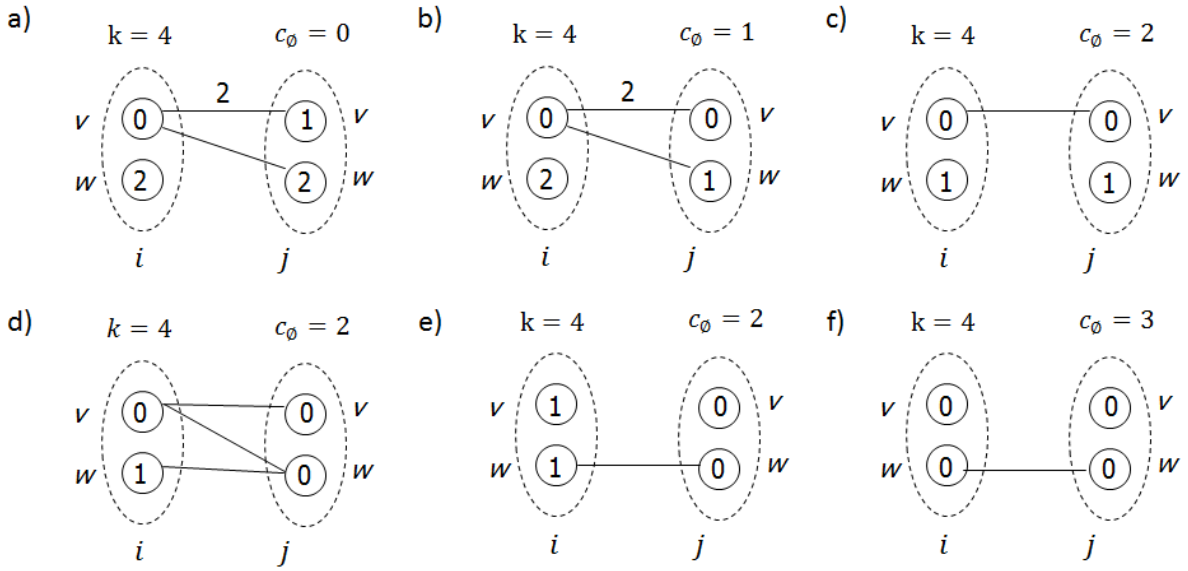


Fig 2-2 Six equivalent WCSPs

The value which satisfies the first condition is denoted the unary support of i. The $ProjectUnary(i)$ EPT operation in **Algorithm 2-1** looks for and creates if needed a unary support for variable i. In our toy example, v is a unary support for i. If the NC property does not hold for a given CFN, it can be enforced by the $NC()$ procedure (**Algorithm 2-2**). It performs the $ProjectUnary(i)$ operations for all variables in order to establish the first condition of the NC property. This process may increase the lower bound $c_\emptyset$. The second condition is established by the loop at line 2 of the $NC()$ procedure. For all variables, it prunes each domain value a such that $c_i(a) + c_\emptyset >= k$. Note that these operations are local to single variables and do not involve simultaneously more than one variable. Hence, the associated time and space complexities are $O(nd)$ where d is the maximum domain size [1]. In **Fig 2-2**, problem b) is an equivalent NC variant of problem a). Application of $ProjectUnary(i)$ on problem a) has no effect because the minimum unary cost of variable i is 0 and $ProjectUnary(j)$ increases $c_\emptyset$ by 1 because the min-

imum unary cost of j is 1. Finally the loop on line 2 of the $NC()$ procedure has no effect on the resulting problem because all values already satisfy the second condition of the NC property.

A similar property exists at the arc level, denoted as Arc Consistency (AC) [1]. An assignment (i, a) is arc consistent with respect to c_{ij} if it is node consistent and $\exists b \in D_j$ such that $c_{ij}(a, b) = 0$. The value b is called the (simple) support of a with respect to c_{ij}. A variable verifies the AC property if all its values are AC with respect to all c_{ij}. Finally, A CFN is AC if it is NC and each variable i have a simple support toward all binary constraints c_{ij}

Algorithm 2-1: Algorithms to propagate costs [7].

```
   /* Remove forbidden assignement from D_i */
   Function PruneVar (i): boolean
       flag := false ;
1      foreach a ∈ D_i do
           if (c_∅ + c_i(a) ≥ k) then
               D_i := D_i/{a} ;
               flag := true ;
       return flag ;

   /* EPT: Send cost from unary constraint toward c_∅ */
   Procedure ProjectUnary (i)
2      α := min_{a∈D_i}{c_i(a)} ;
3      c_∅ := c_∅ + α ;
4      foreach a ∈ D_i do
           c_i(a) := c_i(a) − α ;

   /* EPT: Send cost from binary constraint c_ij(a, .) toward unary constraint c_i(a) */
   Procedure Project (i, a, j, α)
5      c_i(a) := c_i(a) + α ;
6      foreach b ∈ D_j do
           c_ij(a, b) := c_ij(a, b) − α ;

   /* EPT: Send cost from unary constraint c_i(a) toward binary constraint c_ij(a, .) */
   Procedure Extend (i, a, j, α)
7      foreach b ∈ D_j do
           c_ij(a, b) := c_ij(a, b) + α ;
8      c_i(a) := c_i(a) − α ;
```

The AC property is enforced by the $AC()$ procedure in **Algorithm 2-2** with time complexity $O(ed^3)$, where e is the number of constraints. The space complexity is $O(ed)$ because a support has to be stored for each domain value for all constraints. It requires the $FindSupports(i,j)$ procedure defined in **Algorithm 2-3** in order to enforce simple supports (performed by the loop at

line 2 of the $AC()$ procedure). It requires the queue Q that maintains the list of variables for which simple supports are to be found. It is initially filled by all variables X.

Algorithm 2-2: Enforcing NC, AC, DAC and $FDAC$[8]**.** Initially $Q = R = X$.

```
   Procedure NC()
1  |  foreach i ∈ X do
   |  |  ProjectUnary(i) ;
2  |  foreach i ∈ X do
   |  |  PruneVar(i) ;

   Procedure AC()
1  |  while (Q ≠ ∅) do
   |  |  j := pop(Q) ;
2  |  |  foreach c_ij ∈ C do
   |  |  |  if (FindSupports(i, j)) then R := R ∪ {i} ;
3  |  foreach i ∈ X do
   |  |  if (PruneVar(i)) then Q := Q ∪ {i} ;

   Procedure DAC()
1  |  while (R ≠ ∅) do
   |  |  j := pop(R) ;
   |  |  if (PruneVar(j)) then Q := Q ∪ {j} ;
2  |  |  foreach c_ij ∈ Cs.t.i < j do
   |  |  |  if (FindFullSupports(i, j)) then R := R ∪ {i} ;
3  |  foreach i ∈ X do
   |  |  if (PruneVar(i)) then Q := Q ∪ {i} ;

   Procedure FDAC()
   |  while (Q ≠ ∅||R ≠ ∅) do
1  |  |  AC() ;
2  |  |  DAC() ;
```

The problem in **Fig 2-2** c) is an equivalent AC variant of the problems a) and b). Enforcing simple supports for a given variable is a local operation that involves only one cost function and its variables simultaneously. This local property limits the complexity of local consistency algorithms. Similarly to the $ProjectUnary(i)$ procedure for unary supports, cost units are projected from binary constraints toward unary constraints in order to enforce binary support (line 3 of the $FindSupports(i,j)$ procedure). The $Project(i,a,j,\alpha)$ procedure (**Algorithm 2-1**) performs this task by adding α cost units to $c_i(a)$ and subtracts the same amount of cost units to all $c_{ij}(a,b), b \in D_j$. We transform problem b) into problem c) (**Fig 2-2** again) by applying $Project(i,v,j,1)$. The reverse operation can be accomplished by the $Extend(i,a,j,\beta)$ proce-

dure that adds β cost units to all $c_{ij}(a,b), b \in D_j$ and retracts the same amount of cost units from $c_i(a)$.

An arc consistency property called Full Arc Consistency (FAC) has been introduced with a stronger notion of support [7]. An assignment (i,a) is full arc consistent with respect to c_{ij} if it is node consistent and $\exists b \in D_j$ such that $c_{ij}(a,b) + c_j(b) = 0$. Value b is called the full support of a with respect to constraint c_{ij}. A variable satisfies the FAC property if all its values are FAC with respect to all c_{ij}. A CFN is FAC if it is NC and each variable i has a full support with respect to all binary constraints c_{ij}. Full supports are enforced by the $FindFullSupports(i,j)$ procedure in **Algorithm 2-3**. It requires both $Project(i,j,a,\alpha)$ and $Extend(i,a,j,\beta)$ EPT operations [7].

Indeed, being a full support is stronger than being a simple support, but unfortunately, FAC is not a practical property because it may not be possible to enforce it for any arbitrary CFN [7]. Thence, intermediary arc consistency properties have been introduced to strengthen the CFN framework further than the basic AC property. A common simplification is to consider an ordering of the variable set and enforce the arc consistency properties in one direction or another. Namely, a given arc consistency property with respect to the binary constraint c_{ij} can be either enforced for all $i < j$ or for all $i > j$.

Thus, a variable i is Directional Arc Consistent (DAC) if all its domain values have a full support with respect to every constraint c_{ij} such that $i < j$. The property is established by the $DAC()$ procedure with time complexity $O(ed^2)$ and space complexity $O(ed)$. In addition, if all its domain values have a simple support with respect to all c_{ij} such that $i > j$, then i is said to be Full Directional Arc Consistent ($FDAC$). Finally, the CFN is $FDAC$ if all its variables are NC and full directional arc consistent. It is established by the $FDAC()$ procedure and the associated time complexity is $O(end^3)$ [7]. The space complexity remains $O(ed)$. It requires Q as in $AC()$ and an additional queue R which maintains the list of variable for which a full support has to be found.

If we consider problem in **Fig 2-2** c) and the ordering $i < j$, application of $FindFullSupports(i,j)$ produces problem d) after extension and problem e) after projection. Indeed, the quantities that can be projected ($P[]$) and extended ($E[]$) as computed by the algorithm are $[v] = \min(1+0, 0+1) = 1$; $P[w] = \min(0+0, 0+1) = 0$; $E[v] = \max(1-1, 0-0) = 0$ and $E[w] = \max(1-0, 0-0) = 1$. Subsequently, $ProjectUnary(i)$ builds problem f) and increases the lower bound ($c_\emptyset = 3$).

The $FDAC$ property above ensures that every variable has a full support in one direction and a simple support in the other for all c_{ij}. In the direction of the simple support, let say $i > j$, further work can be done in order to get closer to full directional arc consistency [7]. To reach this goal, an additional new notion of local consistency, called Existential Arc Consistency (EAC) has been introduced as a complementary property to $FDAC$ [7].

Algorithm 2-3: Algorithms to enforce supports [7].

```
   /* Find support for all a ∈ D_i w.r.t. constaint c_ij */
   Function FindSupports (i, j): boolean
       flag := false ;
1      foreach a ∈ D_i do
           α := min_{b∈D_j}{c_ij(a, b)};
2          if (α > 0 ∧ c_i(a) = 0) then flag := true ;
3          Project (i, a, j, α) ;
4      ProjectUnary (i) ;
       return flag ;

   /* Find full support for all a ∈ D_i w.r.t. constaint c_ij */
   Function FindFullSupports (i, j): boolean
       flag := false ;
5      foreach a ∈ D_i do
           P[a] := min_{b∈D_j}{c_ij(a, b) + c_j(b)};
6          if (P[a] > 0 ∧ c_i(a) = 0) then flag := true ;
7          foreach b ∈ D_j do
               E[b] := max_{a∈D_i}{P[a] − c_ij(a, b)};
8          foreach b ∈ D_j do
               Extend(j, b, i, E[b]) ;
9          foreach a ∈ D_i do
               Project(i, a, j, P[a]) ;
10     ProjectUnary (i) ;
       return flag ;

   Procedure FindExistentialSupport ((i): boolean)
       flag := false;
11     α := min_{a∈D_i}{c_i(a) + Σ_{c_ij∈Cs.t.j<i} min_{b∈D_j}{c_ij(a, b) + c_j(b)}};
12     if (α > 0 ) then
13         foreach c_ij ∈ Cs.t.j < i do
               flag := flag ∨ FindFullSupports(i,j) ;
       return flag ;
```

In contrast to previous forms of arc consistency that require that all domain values have a binary support (either simple or full), EAC requires that at least, one value $a \in D_i$, such that $c_i(a) = 0$ has a full support with respect to every c_{ij} [7]. Such value is called the existential support of the variable and it is enforced by the $FindExistentialSupport(i)$ procedure in **Algorithm 2-3** (in one direction).

The Existential Directional Arc Consistency ($EDAC$) property is satisfied by a given CFN if it satisfies both EAC and $FDAC$. **Algorithm 2-4** enforces $EDAC$ for any arbitrary binary WCSP with time complexity $O(ed^2 \max\{nd, k\})$ and space complexity $O(ed)$ [7]. Because $EDAC$ implies $FDAC$, it also implies AC, DAC, and NC (as explicit in **Algorithm 2-4**). The addition of

EAC makes $EDAC$ stronger (i.e., it can lead to tighter lower bound of an optimal solution) than any local consistency properties mentioned above.

Algorithm 2-4: Enforcing $EDAC$. S is an auxiliary queue. Initially, $Q = R = S = X$ [7].

```
   Procedure EDAC()
1  |  while (Q ≠ ∅ ∨ R ≠ ∅ ∨ S ≠ ∅) do
2  |  |  P := {l | i ∈ S, l > i, c_il ∈ C} ∪ S ;
   |  |  S := ∅;
   |  |
   |  |  /* EAC */
3  |  |  while (P ≠ ∅) do
   |  |  |  i := popMin(P) ;
   |  |  |  if (FindExistentialSupport(i)) then
   |  |  |  |  R := R ∪ {i} ;
4  |  |  |  |  foreach c_ij ∈ Cs.t.j > i do
   |  |  |  |  |  P := P ∪ {j} ;
   |  |  |
   |  |  /* DAC */
5  |  |  while (R ≠ ∅) do
   |  |  |  j := popMax(R) ;
6  |  |  |  foreach c_ij ∈ Cs.t.i < j do
   |  |  |  |  if (FindFullSupports(i, j)) then
   |  |  |  |  |  R := R ∪ {i} ;
   |  |  |  |  |  S := S ∪ {i} ;
   |  |  |
   |  |  /* AC */
7  |  |  while (Q ≠ ∅) do
   |  |  |  j := popMin(Q) ;
8  |  |  |  foreach c_ij ∈ Cs.t.i > j do
   |  |  |  |  if (FindSupports(i, j)) then
   |  |  |  |  |  R := R ∪ {i} ;
   |  |  |  |  |  S := S ∪ {i} ;
   |  |  |
   |  |  /* NC */
9  |  |  foreach i ∈ X do
   |  |  |  if (PruneVar(i)) then Q := Q ∪ {i} ;
```

Algorithm 2-4 requires three queues (Q, R, P) that store the set of variables for which a given local consistency property is to be enforced. The queues Q and R have the same meaning as in $DAC()$. The queue P maintains a list of variables for which an existential support has to be sought. An auxiliary queue S is used to efficiently build and maintain P. Note that R is traversed in a reverse order to the ordering used for Q and P. The algorithm runs as long as one of the three queues is not empty (main loop at line 1) [7]. Four inner loops sequentially enforce EAC, DAC, AC and NC. The propagation queues are dynamically filled when the associated local consistency

property is possibly broken due to EPT operations performed during support enforcing or value pruning at any time (indicated by the Boolean value returned by support enforcing or pruning algorithms).

2.3 WCSP solvers – Complete search

A WCSP solver is usually a tree search algorithm that maintains a set of local consistency properties both at a processing step (at the root of the tree) and incrementally during the search at every visited node. The incremental nature of LC properties makes WCSP solvers powerful because values are pruned during search in addition to preprocessing pruning. This behavior leads to smaller, easier to solve problems. Generally a complete Depth-First Branch and Bound (DFBB) search is performed because of its low space complexity, but any exhaustive search strategy may apply [1]. Indeed, for every visited node in the DFBB search, a tight lower bound of the cost of an optimal solution of the corresponding subspace ($c_\emptyset$) is computed by the embedded incremental local consistency property as exemplified by the $DFBB(t,k,X,D,C)$ procedure in **Algorithm 2-5**. Hence $c_\emptyset$ is increased during LC enforcing in a given subspace. A generic local consistency enforcing procedure is called at line 4 to check and enforce if needed locally consistency for the new subproblem. The partial assignment is recorded in t, and k is the current upper bound (which can be initially infinite). During this process, backtrack occurs when an inconsistency is detected: a domain wipeout due to value pruning by NC. When a leaf node is reached, the cost of the complete assignment is used to update the upper bound of the WCSP. In other words, every subsequent WCSP associated to explored nodes must give a lower cost to be further expanded.

Algorithm 2-5: Depth-First Branch and Bound Algorithm to maintain local consistencies [1].

```
   Procedure DFBB (t, k, X, D, C)
1  |  if X = ∅ then
   |  |  return c_∅ ;
   |  else
   |  |  i := selectVar(X) ;
   |  |  foreach a ∈ D_i do
2  |  |  |  DD := D; CC := C; Nt := t + (i, a) ;
3  |  |  |  if (LocalConsist(k, X − {i}, DD, CC)) then
4  |  |  |  |  k := DFBB(Nt, k, X − {i}, DD, CC) ;
   |  return k ;
```

DFBB explores a tree where each node is an assignment to a variable (or a domain split) [1]. Each leaf node corresponds to a complete assignment and internal nodes correspond to partial assignment. At each internal node, a variable i is selected form unassigned variables, and all its domain values are propagated either sequentially or by splitting D_i into two or more subsets considered also sequentially. The subtree below any given node is pruned if its lower bound is higher or equal to the current upper bound, meaning that a better solution cannot be found through any

extension of the associated partial assignment. Otherwise, a new unassigned variable i is selected and local consistencies are enforced again for new nodes created by assigning i.

Finally, sophisticated variable and value ordering heuristics can be designed in order to efficiently solve WCSPs by guiding the search. These notions will be extensively introduced in **Chapter 3** of this manuscript along with different strategies to perform branching and backtracking.

2.4 Alternative and complementary strategies

2.4.1 Alternative complete solvers

In this subsection, we briefly summarize some alternative and hybrid approaches available. The aim here is to give some clues about available approaches and their complementary aspect. A full description of the corresponding algorithms can be found elsewhere in the associated papers. A common practice is to combine different methods, wherever heuristic or complete in order to get an efficient solver. This is the philosophy adopted for the *toulbar2* solver [9].

Among complete methods, an appealing alternative strategy to Branch Bound in the Constraint Programming (CP) community (and others) is the Variable Elimination (VE) which is a dynamic programming approach [10]. It is also known as Bucket Elimination (BE) [11]. It synthetizes all optimal solutions by removing variables one by one. In BE, The order in which variables are eliminated is predefined. For each variable, BE dynamically infers an assignment that preserves the optimum. Prior to this elimination phase, each variable i is associated with a new constraint g_i resulting from the removal of i from X. The aggregate constraint g_i sums up the set of all the constraints (B_i) having i in their scope. B_i is called the bucket associated to i. Each bucket is associated to the highest order variable of its scope.

BE has an exponential worst case space complexity because the additional constraints g_i are built other all the neighbors of i (i.e., the other variables of all constraints where i is element of the scope). This characteristic limits the applicability of BE to combinatorial problems where the number of neighbors is small. Thence, an incomplete variant based on the concept of Mini-Bucket (MB) has been introduced in order to overcome this drawback [12], [13]. Instead of considering all the neighboring variables of i to build B_i, only a subset is considered. MB does not guarantee to find the optimal solution but has been useful in providing tight lower bound for WCSP (and other type of) problems [14].

Because of the quality of its lower bound (near to the optimum), hybrid strategies have been devised in order to boost complete search by MB, commonly, a hybrid DFBB-MB [14], [15]. This hybrid was found to outperform several approaches in solving complex WCSP problems, but it is not usually better than local consistency based solvers.

Apart from these complete approaches described above to handle WCSPs, computer science is rich of complete and heuristic methods that can find application in the CPD field. Thence, we have already selected some strategies to tackle CPD including 0/1 Linear Programming, 0/1

Quadratic, Programming, 0/1 Quadratic Optimization, Weighted Partial MaxSAT, Graphical Model optimization problems [16]. Some of the approaches were found impractical for CPD. Although some of them showed good performance with respect to available approach is the CPD community, none of the ones we assessed did outperform the CFN framework.

2.4.2 Metaheuristics and hybrid strategies to solve WCSPs

Because WCSP is NP-complete, several metaheuristics have been designed to address it. Local search and population based strategies constitute an important class of metaheuristic approaches which have found useful application for solving WCSP problems. In Chapter 1, we briefly introduced Monte Carlo search (with Simulated Annealing) which is a local search and Genetic Algorithm which is population based approach.

A well-known local search approach widely used in the CP community is Tabu Search (TS) [17], [18]. TS is a metaheuristic that can be built on top of another heuristic. It allows overcoming local minima through an intelligent interplay between two « opposite features ». First, during optimization, it maintains a list of forbidden (or restricted) moves, referred to as the Tabu list. This list is derived from previously visited solutions. As a consequence, Tabu list prevents local minima by restricting cycling moves. However, application of moves in the Tabu list may lead to solutions better than the best solution so far. Thus, a second ingredient, called aspiration criterion, is introduced to override Tabu moves when it is beneficial.

While local search such as TS produces a new solution at each step, population based approaches, as implied by their name, produces at each step a set of candidate solutions. A branch and bound approach of this type is Beam Search (BS) [19]. BS maintains a limited list of promising candidates constituting the so-called Beam (partial solutions), denoted B. The size of the beam is called the beam width (k_{bw}). An initial beam is created by assigning unassigned variables (one or more such that we get k_{bw} elements). At each step of the algorithm, each element of the Beam (B) is extended in k_{ext} different ways by assigning values to an unassigned variable. The best k_{bw} elements of the newly produced partial solutions make up the Beam for the next step, and the process continues until complete assignments are found. Hence, some nodes are pruned because of the limited size of the Beam, and BS is an incomplete algorithm combining Branch and Bound, Best-First and Breadth-First search strategies. As a consequence of its incompleteness, the quality of BS is strongly dependent on the quality of the lower bound that drives the selections for the beam generation [20]. This emphasizes its possible complementary with MB or LC which provide a “good” lower bound.

A second widespread population based approaches are Evolutionary Algorithms (EAs). EAs apply operations inspired from biological evolution to generate hybrid solutions for the next generation of the population. A representative approach of this class is Genetic Algorithm (GA) that we have already discussed in Chapter 1. Memetic Algorithm (MA) is an extension of GA were no explicit mutation is required during the evolution process [21], [22]. MA can be outlined trough four types of operations: *i)* generation of an initial population ; *ii)* cooperation of the elements of the current population to produce recombinant solutions ; *iii)* improvement of the newly generat-

ed solutions by local search; *iv)* competition between the improved solutions to produce the next generation of the population. This process goes on until a termination condition is met.

MA incorporates problem specific knowledge during the population generation in contrast to a sheer stochastic sampling, such as in the mutation step of GA. It is thus designed to benefit from local search and other heuristics. The formulation of the MA framework allows its synergic combination with other algorithms in order to build efficient solvers. In a representative strategy, BS is used to produce partial solutions [20]. MB produces a tight lower bound on the resulting subproblem (can be replaced by LC), overcoming the drawback of BS. MB is used again to combine partial solutions (crossover operations) without any explicit mutations. TS subsequently complete and improve partial solutions. Such MA-MB-BS-TS hydride approaches are attractive and have been proven useful in solving challenging WCSP problems, but remains incomplete [20].

2.5 REFERENCES

[1] J. Larrosa and T. Schiex, "Solving weighted CSP by maintaining arc consistency," *Artif. Intell.*, vol. 159, no. 1–2, pp. 1–26, 2004.

[2] T. Schiex, H. Fargier, and G. Verfaillie, "Valued constraint satisfaction problems: Hard and easy problems," *Int. Jt. Conf. Artif. Intell.*, vol. 14, pp. 631–639, 1995.

[3] S. Bistarelli, U. Montanari, and F. Rossi, "Semiring based constraint solving and Optimization," *J. ACM*, vol. 44, no. 2, pp. 201–236, 1997.

[4] B. Cabon, S. De Givry, L. Lobjois, T. Schiex, and J. P. Warners, "Radio link frequency assignment," *Constraints*, vol. 4, no. 1, pp. 79–89, 1999.

[5] S. de Givry, Z. Vitezica, I. Palhiere, and T. Schiex, "MendelSoft: Mendelian error detection in complex pedigree using weighted constraint satisfaction techniques," in *8th World Congress on Genetics Applied to Livestock Production*, 2006, p. 2p.

[6] M. Zytnicki, C. Gaspin, and T. Schiex, "DARN! A weighted constraint solver for RNA motif localization," *Constraints*, vol. 13, no. 1, pp. 91–109, 2008.

[7] S. De Givry, M. Zytnicki, F. Heras, and J. Larrosa, "Existential arc consistency: Getting closer to full arc consistency in weighted CSPs," *IJCAI*, vol. 19, p. 84, 2005.

[8] J. Larrosa and T. Schiex, "In the quest of the best form of local consistency for weighted CSP," *Int. Jt. Conf. Artif. Intell.*, vol. 18, pp. 239–244, 2003.

[9] D. Allouche, S. de Givry, and T. Schiex, "ToulBar2, an open source exact cost function network solver," 2010.

[10] U. Bertelé and F. Brioshi, *Nonserial Dynamic Programming*. Academic Press, 1972.

[11] R. Dechter, "Bucket elimination: A unifying framework for reasoning," *Artif. Intell.*, 1999.

[12] R. Dechter, "Mini-Buckets: A General Scheme for Generating Approximations in Automated Reasoning," in *IJCAI*, 1997, pp. 1297–1303.

[13] R. Dechter and I. Rish, "Mini-buckets: A general scheme for bounded inference," *J. ACM*, vol. 50, no. 2, pp. 107–153, 2003.

[14] J. Larrosa and R. Dechter, "Boosting search with variable elimination in constraint optimization and constraint satisfaction problems," *Constraints*, 2003.

[15] E. Rollon and J. Larrosa, "Depth-first mini-bucket elimination," *Princ. Pract. Constraint Program. …*, 2005.

[16] D. Allouche, I. André, S. Barbe, J. Davies, S. de Givry, G. Katsirelos, B. O'Sullivan, S. Prestwich, T. Schiex, and S. Traoré, "Computational protein design as an optimization problem," *Artif. Intell.*, vol. 212, pp. 59–79, Mar. 2014.

[17] F. Glover, "Tabu search-part I," *ORSA J. Comput.*, 1989.

[18] F. Glover, "Tabu search—part II," *ORSA J. Comput.*, 1990.

[19] A. Barr and E. Feigenbaum, "The handbook of artificial intelligence: Vols. 1-2. Los Altos, CA: William Kaufmann," 1981.

[20] J. Gallardo, C. Cotta, and A. Fernández, "Solving weighted constraint satisfaction problems with memetic/exact hybrid algorithms," *J. Artif. Intell.*, 2009.

[21] P. Moscato and C. Cotta, "A gentle introduction to memetic algorithms," *Handb. metaheuristics*, 2003.

[22] N. Krasnogor and J. Smith, "A tutorial for competent memetic algorithms: model, taxonomy, and design issues," *Evol. Comput. IEEE*, 2005.

3 CFN-based Framework for Computational Protein Design

Contents

3.1 CPD as a Cost Function Network optimization problem 59
3.2 CPD as an optimization problem 69
3.3 A new framework for CPD through Cost Function Network optimization 105
3.4 Fast search algorithms for CPD 133
3.5 References 150

Within this chapter, cutting-edge optimization methods originating from artificial intelligence were combined to molecular modeling techniques to propose novel approaches aiming at handling the high complexity of combinatorial sequence-conformation spaces inherent to Computational Protein Design (CPD).

The chapter is composed of independent articles (that are already published or will shortly be submitted to an international journal). The article presented in Section 3-5 assesses the CFN modeling framework on the GMEC identification problem. The Section 3-6 considers a larger data set and modeling frameworks in order to assess furthermore the efficiency of the CFN framework. In Section 3-7, we address the suboptimal set enumeration problem. Finally, the CFN methods are integrated in a CPD-dedicated software and hybrid strategies are designed to enrich the proposed CFN-based framework in Section 3-8.

3.1 Computational Protein Design as a Cost Function Network Optimization Problem

In collaboration with the group of Thomas Schiex (MIAT-INRA, Auzeville) specialized in combinatorial optimization methods, we modeled the CPD problem as a binary Cost Function Network (CFN) and 0/1 Linear Programming (LP) problem. The performances of the CFN solver *toulbar2* and the 0/1 LP solver *cplex* were compared to those of well-established CPD approaches to identify the Global Minimum Energy Conformation (GMEC) on a set of 12 protein design problems. Results highlighted the efficiency of the CFN methodology over other methods to solve CPD instances. The CFN-methods enabled the resolution of more CPD problems (10 vs 4 for CFN and CPD-dedicated solver, respectively) and in most cases, it gave important speedups compared to one of the most commonly used deterministic algorithm in the field of Protein Design, the Dead End Elimination (DEE)/A*, implemented in the *osprey* package.

The results are detailed in an article published in the Proceedings of the 18th International Conference on Principles and Practice of Constraint Programming (Québec, Canada, October, 8-12 2012) [1] and are presented hereafter.

Computational Protein Design as a Cost Function Network Optimization Problem

David Allouche[1*], Seydou Traoré[2*], Isabelle André[2], Simon de Givry[1], George Katsirelos[1], Sophie Barbe[2**], and Thomas Schiex[1**]

[1] UBIA, UR 875, INRA, F-31320 Castanet Tolosan, France
[2] LISBP, INSA, UMR INRA 792/CNRS 5504, F-31400 Toulouse, France

Abstract. Proteins are chains of simple molecules called amino acids. The three-dimensional shape of a protein and its amino acid composition define its biological function. Over millions of years, living organisms have evolved and produced a large catalog of proteins. By exploring the space of possible amino-acid sequences, protein engineering aims at similarly designing tailored proteins with specific desirable properties. In Computational Protein Design (CPD), the challenge of identifying a protein that performs a given task is defined as the combinatorial optimization problem of a complex energy function over amino acid sequences.
In this paper, we introduce the CPD problem and some of the main approaches that have been used to solve it. We then show how this problem directly reduces to Cost Function Network (CFN) and 0/1LP optimization problems. We construct different real CPD instances to evaluate CFN and 0/1LP algorithms as implemented in the `toulbar2` and `cplex` solvers. We observe that CFN algorithms bring important speedups compared to the CPD platform `osprey` but also to `cplex`.

1 Introduction

A protein is a sequence of basic building blocks called amino acids. Proteins are involved in nearly all structural, catalytic, sensory, and regulatory functions of living systems [11]. Performance of these functions generally requires the assembly of proteins into well-defined three-dimensional structures specified by their amino acid sequence. Over millions of years, natural evolutionary processes have shaped and created proteins with novel structures and functions by means of sequence variations, including mutations, recombinations and duplications. Protein engineering techniques coupled with high-throughput automated procedures offer today the possibility to mimic the evolutionary process on a greatly accelerated time-scale, and thus increase the odds to identify the proteins of interest for technological uses [29]. This holds great interest for medicine, biotechnology, synthetic biology and nanotechnologies [27, 32, 15].

With a choice among 20 naturally occuring amino acids at every position, the size of the combinatorial sequence space is however clearly out of reach of current experimental methods, even for small proteins. Computational protein design (CPD) methods therefore try to intelligently guide this process by producing a collection of proteins, intended to be rich in functional proteins and whose size is small enough to be experimentally evaluated. The challenge of choosing a sequence of amino acids to perform a given task is formulated as an optimization problem, solvable computationally. It is often described as the inverse problem of protein folding [28]: the three-dimensional structure is known and we have to find amino acid sequences that folds into it. It can also be considered as a highly combinatorial variant of side-chain positioning [35] because of possible amino acid changes.

* These authors contributed equally to this work.
** To whom correspondence should be addressed (Thomas.Schiex@toulouse.inra.fr and Sophie.Barbe@insa-toulouse.fr).

Different computational methods have been proposed over the years to solve this problem and several success stories have demonstrated the outstanding potential of CPD methods to engineer proteins with improved or novel properties. CPD has been successfully applied to increase protein thermostability and solubility; to alter specificity towards some other molecules; and to design various binding sites and construct *de novo* enzymes (see for example [18]).

Despite these significant advances, CPD methods still have to mature in order to better guide and accelerate the construction of tailored proteins. In particular, more efficient computational optimization techniques are needed to explore the vast protein sequence-conformation combinatorial space.

In this paper, we model CPD problems as either binary Cost Function Network (CFN) or 0/1LP problems. We compare the performance of the CFN solver `toulbar2` and the 0/1LP solver `cplex` against that of well-established CPD approaches on various protein design problems. On the various problems considered, the direct application of `toulbar2`, a Depth First Branch and Bound algorithm maintaining soft local consistencies, resulted in an improvement of several orders of magnitude compared to dedicated CPD methods and also outperformed `cplex`. These preliminary results can probably be further improved both by tuning our solver to the specific nature of the problem considered and by incorporating dedicated CPD preprocessing methods.

2 The Computational Protein Design approach

In CPD, we are given an existing protein corresponding to a native sequence of amino acids folded into a 3D structure, which has previously been determined experimentally. The task consists in modifying a given property of the protein (such as stability or functional efficiency) through the mutation of a specific subset of amino acid residues in the sequence, i.e. by affecting their identity and their 3D orientation (rotamers). The resulting designed protein retains the overall folding of the original protein since we consider the protein *backbone* as fixed and only alter the amino acid *side chains* (Fig. 1). The stability and functional efficiency of a protein is correlated to its energy [1]. Therefore, we aim at finding the conformation possessing the minimum total energy, called *GMEC* (Global Minimum Energy Conformation). The energy of a conformation can be directly computed from the amino acid sequence and rotamers by introducing substitutions within the native structure.

Rotamers. The distribution of accessible conformations available to each amino acid side chain is approximated using a set of discrete conformations defined by the value of their inner dihedral angles. These conformations, or *rotamers*, are derived from the most frequent conformations in the experimental repository of known protein structures PDB (Protein Data Bank, `www.wwpdb.org`).

Energy function. Typical energy function approximations [3] use the assumption that the amino acid identity substitutions and rotamers do not modify the folding of the protein. They include non-bonded terms such as van der Waals and electrostatics, often in conjunction with empirical contributions describing hydrogen bond. The surrounding solvent effect is generally treated implicitly as a continuum. In addition, statistical terms may be added in order to approximate the effect of mutations on the unfolded state or the contribution of conformational entropy.

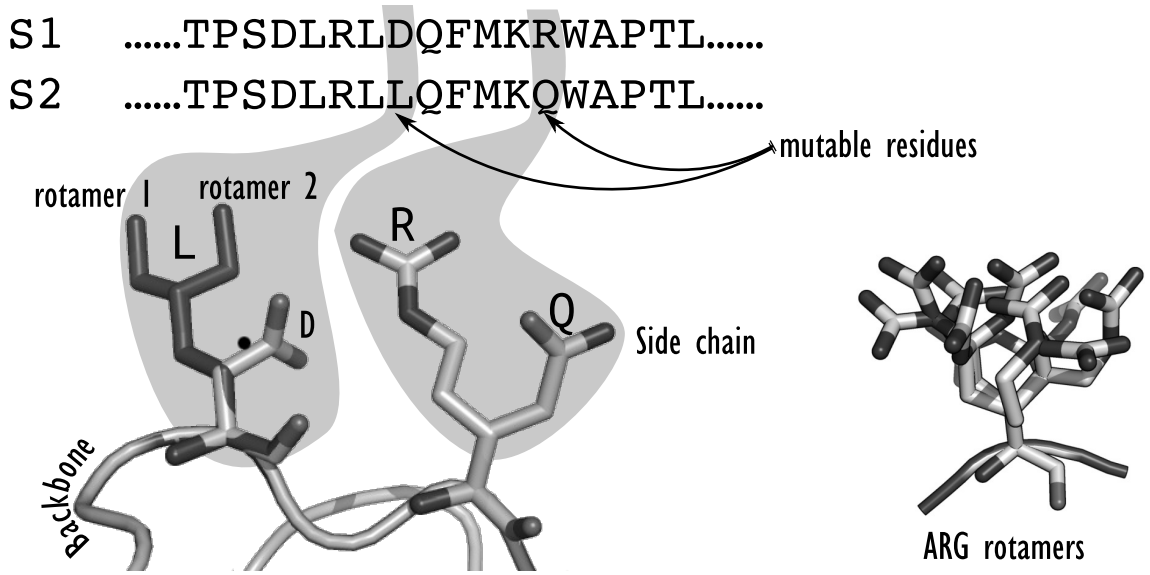


Fig. 1. A local view of combinatorial sequence exploration considering a common backbone. Changes can be caused by amino acid identity substitutions (*for example* D/L or R/Q) or by amino acid side-chain reorientations (rotamers) for a given amino acid. A typical rotamer library for one amino acid is shown on the right (ARG=Arginine).

These energy functions can be reformulated in such a way that the terms are locally decomposable. Then, the energy of a given protein defined by a choice of one specific amino acid with an associated conformation (rotamer) for each residue, can be written as:

$$E = E_c + \sum_i E(i_r) + \sum_i \sum_{j>i} E(i_r, j_s) \tag{1}$$

where E is the potential energy of the protein, E_c is a constant energy contribution capturing interactions between fixed parts of the model, $E(i_r)$ is the self energy of rotamer r at position i capturing internal interactions or with fixed regions, and $E(i_r, j_s)$ is the pairwise interaction energy between rotamer r at position i and rotamer s at position j [9]. All terms are measured in $kcal/mol$ and can be pre-computed and cached.

3 Existing approaches for the CPD

The protein design problem as defined above, with a rigid backbone, a discrete set of rotamers, and pairwise energy functions has been proved to be NP-hard [31]. Hence, a variety of meta-heuristics have been applied to it, including Monte Carlo simulated annealing [21], genetic algorithms [33], and other algorithms [10]. The main weakness of these approaches is that they may remain stuck in local minima and miss the GMEC without notice.

However, there are several reasons motivating the exact solving of the problem. First, because they know when an optimum is reached, exact methods may stop before meta-heuristics. Voigt et al. [36] reported that the accuracy of metaheuristics also degrades as problem size increases. More importantly, the use of exact search algorithms becomes crucial in the usual experimental design cycle that goes through CPD modeling, solving, protein synthesis and experimental evaluation: when unexpected experimental results are obtained, the only possible culprit lies in the CPD model and not in the algorithm.

Current exact methods for CPD mainly rely on the dead-end-elimination (DEE) theorem [9, 8] and the A^* algorithm [24, 13]. From a constraint satisfaction perspective, the DEE theorem can be seen as an extension of neighborhood substitutability [7, 20, 2]. DEE is used as a pre-processing technique and removes rotamers that are locally dominated by other rotamers, until a fixpoint is reached. The rotamer r at position i is removed if there exists another rotamer u at the same position such that [9]:

$$E(i_r) - E(i_u) + \sum_{j \neq i} \min_s E(i_r, j_s) - \sum_{j \neq i} \max_s E(i_u, j_s) > 0$$

That is, r is removed if for any conformation with this r, we get a conformation with lower energy if we substitute u for r.

Extensions to higher orders have been considered [14, 30, 25, 12]. These DEE criteria preserve the optimum but may remove suboptimal solutions.

This DEE preprocessing is usually followed by an A^* search method. After DEE pruning, the A^* algorithm allows to expand a sequence-conformation tree, so that sequence-conformations are extracted and sorted on the basis of their energy values. At depth d of the tree, the lower bound used by $A*$ [13] is exactly the PFC-DAC lower bound [37, 23] used in WCSP and later obsoleted by soft arc consistencies [34, 22, 5]:

$$\underbrace{\sum_{i=1}^{d} E(i_r) + \sum_{j=i+1}^{d} E(i_r, j_s)}_{\text{Assigned}} + \sum_{j=d+1}^{n} [\underbrace{\min_s (E(j_s) + \sum_{i=1}^{d} E(i_r, j_s)}_{\text{Forward checking}} + \underbrace{\sum_{k=j+1}^{n} \min_u E(j_s, k_u))}_{\text{DAC counts}}]$$

If the DEE algorithm does not significantly reduce the search space, the A^* search tree is too memory demanding and the problem cannot be solved. Therefore, to circumvent these limitations and increase the ability of CPD to tackle problems with larger sequence-conformation space, novel alternative methods are needed. Here, we show that state-of-the-art methods for solving Cost Function Networks offer an attractive alternative to this combined DEE/A^* approach, to solve highly complex case studies of protein design.

4 Cost Function Network model

A Cost Function Network (CFN) is a pair (X, W) where $X = \{1, \ldots, n\}$ is a set of n variables and W a set of cost functions. Each variable $i \in X$ has a finite domain D_i of values than can be assigned to it. A value $a \in D_i$ is denoted i_a. For a set of variables $S \subseteq X$, D_S denotes the Cartesian product of the domain of the variables in S. For a given tuple of values t, $t[S]$ denotes the projection of t over S. A cost function $w_S \in W$, with scope $S \subseteq X$, is a function $w_S : D_S \mapsto [0, k]$ where k is a maximum integer cost used for forbidden assignments. The Weighted Constraint Satisfaction Problem (WCSP) is to find a complete assignment t minimizing the combined cost function $\sum_{w_S \in W} w_S(t[S])$. This optimization problem has an associated NP-complete decision problem.

Modeling the CPD problem as a CFN is straightforward. The set of variables X has one variable i per residue i. The domain of each variable is the set of *(amino acid,conformation)* pairs in the rotamer library used. The energy function can be represented by 0-ary, unary and binary cost functions respectively capturing the constant energy term E_c, the unary energy terms $E(i_r)$ and the binary energy terms $E(i_r, j_s)$. There is just one discrepancy between the original formulation and the CFN model: energies are represented as arbitrary floating point numbers while CFN use positive integer

costs. This can simply be fixed by first subtracting the minimum energy to all energies and then by multiplying energies by a large integer constant M.

5 Integer Linear programming model

The resulting CFN can also be represented as a 0/1 linear programming problem using the encoding proposed in [20]. For every value i_r, there is a boolean variable $d_{i,r}$ which is equal to 1 iff $i = r$. Additional constraints enforce that exactly one value is selected for each variable. For every pair of values of different variables (i_r, j_s) involved in a binary energy term, there is a boolean variable $p_{i,r,j,s}$ which is equal to 1 iff the pair (i_r, j_s) is used. Constraints enforce that a pair is used iff the corresponding values are used. Then, finding a GMEC reduces to the following ILP:

$$\begin{aligned} \min & \textstyle\sum_{i,r} E(i_r).d_{i,r} + \sum_{i,r,j,s} E(i_r, j_s).p_{i,r,j,s} \\ \text{s.t.} & \textstyle\sum_r d_{i,r} = 1 && (\forall i) \\ & \textstyle\sum_s p_{i,r,j,s} = d_{i,r} && (\forall i, r, j) \end{aligned}$$

This model is also the ILP model IP1 proposed in [19] for side-chain positioning. The continuous relaxation of this 0/1 linear programming model is known do be the dual of the LP problem encoded by Optimal Soft Arc Consistency [6, 5]. When the upper bound k is infinite, OSAC is known to be stronger than any other soft "arc level" arc consistency and especially stronger than the default Existential Directional Arc Consistency (EDAC) [22] used in `toulbar2`. However, as soon as the upper bound k decreases to a finite value, soft local consistencies may prune values and EDAC becomes incomparable with OSAC.

6 Experimental Results

We used a set of 12 protein design cases to evaluate the performance of `toulbar2`, `cplex` and compare them with the DEE/A* approach implemented in `osprey` (open source dedicated Java CPD software). This set comprises 9 protein structures derived from the PDB which were chosen for the high resolution of their 3D-structures and their distribution of sizes and types. Diverse sizes of sequence-conformation combinatorial spaces were considered, varying by the number of mutable residues, the number of alternative amino acid types at each position and the number of conformations for each amino acid (Table 1). The *Penultimate* rotamer library was used [26].

Preparation of CPD instances. Missing heavy atoms in crystal structures and hydrogen atoms were added with the *tleap* module of the AMBER9 software package [4]. Each molecular system was then minimized in implicit solvent (Generalized Born model [17]) using the *Sander* program and the all-atom *ff99* force field of AMBER9. All E_c, $E(i_r)$, and $E(i_r, j_s)$ energies of rotamers (see Equation 1) were pre-computed using `osprey`. The energy function consisted of the Amber electrostatic, van der Waals, and dihedral terms. These calculations were performed on an Altix ICE 8200 supercomputer with 2,816 Intel Nehalem EX 2.8 GHz cores. We used 32 cores and 128GB of RAM. The sequential CPU time needed to compute the set of all energy cost functions is given in Table 1. Although these computation times can be very large, they are also highly parallelizable. For n residues to optimize with d possible (amino acid,conformation) pairs, there are n unary and $\frac{n.(n-1)}{2}$ binary cost functions which can be computed independently.

DEE/A optimization.* To solve the different protein design cases, we used `osprey` version 1.0 (`cs.duke.edu/donaldlab/osprey.php`) which first filters rotamers i_r such that $E(i_r) > 30kcal/mol$ and pairs (i_r, j_s) such that $E(i_r, j_s) > 100kcal/mol$ (*pruningE* and *stericE* parameters). This step is followed by extensive DEE pre-processing (*algOption* = 3, includes simple Goldstein, Magic bullet pairs, 1 and 2-split positions, Bounds and pairs pruning) and A^* search. Only the GMEC conformation is generated by A^* (*initEw*=0). Computations were performed on a single core of an AMD Operon 6176 at 2.3 GHz, 4 GB of RAM, and a 100-hour time-out. There were no memory-out errors.

CFN and ILP optimization. The same problems (before DEE preprocessing and using $M = 10^8$) have been tackled by `cplex` version 12.2 (parameters EPAGAP, EPGAP and EPINT set to zero to avoid premature stop) and `toulbar2` version 0.9.5 (`mulcyber.toulouse.inra.fr/projects/toulbar2/`) using binary branching with an initial limited discrepancy search phase [16] with a maximum discrepancy of 2 (options `-d: -l=2`, and other default options including EDAC and no initial upper bound) and domains sorted with increasing unary costs $E(i_r)$. These computations were performed on a single core of an Intel Xeon E5430 core at 2.66 GHz with 64GB of RAM with a 100-hour time-out.

With the exception of one instance (1CM1), `cplex` significantly outperforms `osprey`. On the other hand, `toulbar2` is always faster than both `cplex` and `osprey` by at least one order of magnitude and often many more, even accounting for the performance discrepancy arising from the difference in the hardware we used. We have also verified that the minimum energy reported by all 3 solvers is identical.

Table 1. For each instance: protein (PDB id.), amino acid sequence length, number of mutable residues, maximum number of (amino acid, conformation) pairs, sequential time for computing $E(\cdot)$ energy functions, and CPU-time for solving using `osprey`, `cplex`, and `toulbar2`. A '-' indicates that the 100-hour limit has been reached.

System name	Size	n	d	$E(\cdot)$	`osprey`	`cplex`	`toulbar2`
Thioredoxin (2TRX)	108	11	44	304 min.	27.1 sec.	2.6 sec.	0.1 sec.
Protein G (1PGB)	56	11	45	76 min.	49.3 sec.	14.7 sec.	0.1 sec.
Protein L (1HZ5)	64	12	45	114 min.	1,450 sec.	17.7 sec.	0.1 sec.
Ubiquitin (1UBI)	76	13	45	270 min.	-	405.0 sec.	0.6 sec.
Protein G (1PGB)	56	11	148	1,096 min.	-	2,245 min.	13.9 sec.
Protein L (1HZ5)	64	12	148	831 min.	-	1,750 min.	14.6 sec.
Ubiquitin (1UBI)	76	13	148	1,967 min.	-	-	378 min.
Plastocyanin (2PCY)	99	18	44	484 min.	-	89.5 sec.	0.5 sec.
Haloalkane Dehalogenase (2DHC)	310	14	148	45,310 min.	-	-	77.4 sec.
Calmodulin (1CM1)	161	17	148	11,326 min.	121.9 sec.	1,707 sec.	2.0 sec.
Peptidyl-prolyl cis-trans Isomerase (1PIN)	153	28	148	40,491 min.	-	-	-
Cold-Shock (1C9O)	132	55	148	84,089 min.	-	-	-

6.1 Explaining the differences

The ILP solver CPLEX is a totally closed-source black box. More generally, solvers are complex systems involving various mechanisms. The effect of their interactions during solving is hard to predict. Therefore, explaining the differences in efficiency observed between the different approaches is not really obvious.

If we consider `osprey` first, it uses an obsolete lower bound instead of the more recent incremental and stronger lower bounds offered by soft local consistencies such as EDAC [22]. This, together with the associated informed value ordering provided by these

local consistencies, may explain why `toulbar2` outperforms `osprey`. Similarly, the LP relaxation lower bound used in ILP is known (by duality) to be the same as the Optimal Soft AC lower bound (when no upper bounding occurs, i.e. when $k = +\infty$). Since OSAC dominates all other local consistencies at the arc level, this provides an explanation for the efficiency of `cplex` compared to `osprey`. Finally, the problem is deeply non linear. It can be concisely formulated as a CFN but the ILP formulation is much more verbose. This probably contributes, together with the upper bounding (provided by node consistency) and value ordering heuristics of `toulbar2`, to the efficiency of `toulbar2` compared to `cplex`.

7 Conclusion

The simplest formal optimization problem underlying CPD looks for a Global Minimum Energy Conformation (GMEC) over a rigid backbone and altered side-chains (identity and conformation). It can easily be reduced to a binary Cost Function Network, with a very dense graph and relatively large domains or to 0/1LP with a large number of variables.

On a variety of real instances, we have shown that state-of-the-art CFN algorithms but also 0/1LP algorithms give important speedups compared to usual CPD algorithms combining Dead End Elimination with $A*$ as implemented in the `osprey` package. CFN algorithms are the most efficient by far and have the advantage of requiring reasonable space.

Although existing CFN algorithms still need to be extended and adapted to tackle such problems, the rigid backbone method reported herein may contribute to the development of more sophisticated flexible methods.

Acknowledgements This work has been partly funded by the "Agence nationale de la Recherche", reference ANR-10-BLA-0214. We would like to thank Damien Leroux for his help in the generation of `cplex` encodings using Python. We thank the Computing Center of Region Midi-Pyrénées (CALMIP, Toulouse, France) and the GenoToul Bioinformatics Platform of INRA-Toulouse for providing computing resources and support.

References

1. Anfinsen, C.: Principles that govern the folding of protein chains. Science 181(4096), 223–253 (1973)
2. Bistarelli, S., Faltings, B., Neagu, N.: Interchangeability in soft csps. In: International Workshop on Constraint Solving and Constraint Logic Programming. pp. 31–46 (2002)
3. Boas, F., Harbury, P.: Potential energy functions for protein design. Current opinion in structural biology 17(2), 199–204 (2007)
4. Case, D., Darden, T., Cheatham III, T., Simmerling, C., Wang, J., Duke, R., Luo, R., Merz, K., Pearlman, D., Crowley, M., Walker, R.C., Zhang, W., Wang, B., Hayik, S., Roitberg, A., Seabra, G., Wong, K.F., Paesani, F., Wu, X., Brozell, S., Tsui, V., Gohlke, H., Yang, L., Tan, C., Mongan, J., Hornak, V., Cui, G., Beroza, P., Mathews, D.H., Schafmeister, C., Ross, W.S., Kollman, P.A.: Amber 9. University of California, San Francisco (2006)
5. Cooper, M., de Givry, S., Sanchez, M., Schiex, T., Zytnicki, M., Werner, T.: Soft arc consistency revisited. Artificial Intelligence 174, 449–478 (2010)
6. Cooper, M.C., de Givry, S., Schiex, T.: Optimal soft arc consistency. In: Proc. of IJCAI'2007. pp. 68–73. Hyderabad, India (Jan 2007)
7. Cooper, M.: Fundamental properties of neighbourhood substitution in constraint satisfaction problems. Artificial Intelligence 90(1-2), 1–24 (1997)
8. Dahiyat, B., Mayo, S.: Protein design automation. Protein Science 5(5), 895–903 (1996)
9. Desmet, J., Maeyer, M., Hazes, B., Lasters, I.: The dead-end elimination theorem and its use in protein side-chain positioning. Nature 356(6369), 539–542 (1992)

10. Desmet, J., Spriet, J., Lasters, I.: Fast and accurate side-chain topology and energy refinement (FASTER) as a new method for protein structure optimization. Proteins: Structure, Function, and Bioinformatics 48(1), 31–43 (2002)
11. Fersht, A.: Structure and mechanism in protein science: a guide to enzyme catalysis and protein folding. WH Freemean and Co., New York (1999)
12. Georgiev, I., Lilien, R., Donald, B.: Improved pruning algorithms and divide-and-conquer strategies for dead-end elimination, with application to protein design. Bioinformatics 22(14), e174–e183 (2006)
13. Georgiev, I., Lilien, R., Donald, B.: The minimized dead-end elimination criterion and its application to protein redesign in a hybrid scoring and search algorithm for computing partition functions over molecular ensembles. Journal of Computational Chemistry 29(10), 1527–1542 (2008)
14. Goldstein, R.: Efficient rotamer elimination applied to protein side-chains and related spin glasses. Biophysical Journal 66(5), 1335–1340 (1994)
15. Grunwald, I., Rischka, K., Kast, S., Scheibel, T., Bargel, H.: Mimicking biopolymers on a molecular scale: nano (bio) technology based on engineered proteins. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 367(1894), 1727–1747 (2009)
16. Harvey, W.D., Ginsberg, M.L.: Limited discrepency search. In: Proc. of the 14^{th} IJCAI. Montréal, Canada (1995)
17. Hawkins, G., Cramer, C., Truhlar, D.: Parametrized models of aqueous free energies of solvation based on pairwise descreening of solute atomic charges from a dielectric medium. The Journal of Physical Chemistry 100(51), 19824–19839 (1996)
18. Khare, S., Kipnis, Y., Takeuchi, R., Ashani, Y., Goldsmith, M., Song, Y., Gallaher, J., Silman, I., Leader, H., Sussman, J., et al.: Computational redesign of a mononuclear zinc metalloenzyme for organophosphate hydrolysis. Nature Chemical Biology 8(3), 294–300 (2012)
19. Kingsford, C., Chazelle, B., Singh, M.: Solving and analyzing side-chain positioning problems using linear and integer programming. Bioinformatics 21(7), 1028–1039 (2005)
20. Koster, A., van Hoesel, S., Kolen, A.: Solving frequency assignment problems via tree-decomposition. Tech. Rep. RM/99/011, Universiteit Maastricht, Maastricht, The Netherlands (1999)
21. Kuhlman, B., Baker, D.: Native protein sequences are close to optimal for their structures. Proceedings of the National Academy of Sciences 97(19), 10383 (2000)
22. Larrosa, J., de Givry, S., Heras, F., Zytnicki, M.: Existential arc consistency: getting closer to full arc consistency in weighted CSPs. In: Proc. of the 19^{th} IJCAI. pp. 84–89. Edinburgh, Scotland (Aug 2005)
23. Larrosa, J., Meseguer, P., Schiex, T., Verfaillie, G.: Reversible DAC and other improvements for solving max-CSP. In: Proc. of AAAI'98. Madison, WI (Jul 1998)
24. Leach, A., Lemon, A., et al.: Exploring the conformational space of protein side chains using dead-end elimination and the A* algorithm. Proteins Structure Function and Genetics 33(2), 227–239 (1998)
25. Looger, L., Hellinga, H.: Generalized dead-end elimination algorithms make large-scale protein side-chain structure prediction tractable: implications for protein design and structural genomics1. Journal of molecular biology 307(1), 429–445 (2001)
26. Lovell, S., Word, J., Richardson, J., Richardson, D.: The penultimate rotamer library. Proteins: Structure, Function, and Bioinformatics 40(3), 389–408 (2000)
27. Nestl, B., Nebel, B., Hauer, B.: Recent progress in industrial biocatalysis. Current Opinion in Chemical Biology 15(2), 187–193 (2011)
28. Pabo, C.: Molecular technology: designing proteins and peptides. Nature 301, 200 (1983)
29. Peisajovich, S., Tawfik, D.: Protein engineers turned evolutionists. Nature methods 4(12), 991–994 (2007)
30. Pierce, N., Spriet, J., Desmet, J., Mayo, S.: Conformational splitting: A more powerful criterion for dead-end elimination. Journal of computational chemistry 21(11), 999–1009 (2000)
31. Pierce, N., Winfree, E.: Protein design is NP-hard. Protein Engineering 15(10), 779–782 (2002)
32. Pleiss, J.: Protein design in metabolic engineering and synthetic biology. Current Opinion in Biotechnology 22(5), 611–617 (2011)
33. Raha, K., Wollacott, A., Italia, M., Desjarlais, J.: Prediction of amino acid sequence from structure. Protein Science 9(6), 1106–1119 (2000)
34. Schiex, T.: Arc consistency for soft constraints. In: Principles and Practice of Constraint Programming - CP 2000. LNCS, vol. 1894, pp. 411–424. Singapore (Sep 2000)
35. Swain, M., Kemp, G.: A CLP approach to the protein side-chain placement problem. In: Principles and Practice of Constraint Programming—CP 2001. pp. 479–493. Springer (2001)
36. Voigt, C., Gordon, D., Mayo, S.: Trading accuracy for speed: a quantitative comparison of search algorithms in protein sequence design. Journal of Molecular Biology 299(3), 789–803 (2000)
37. Wallace, R.: Directed arc consistency preprocessing. In: Meyer, M. (ed.) Selected papers from the ECAI-94 Workshop on Constraint Processing, pp. 121–137. No. 923 in LNCS, Springer, Berlin (1995)

3.2 **Computational protein design as an optimization problem**

On the basis of the encouraging results of the first study, we evaluated the performances of other combinatorial optimization methods derived from artificial intelligence (0/1 Quadratic Programming, 0/1 Quadratic optimization, Weighted Partial Mas SAT and Graphical Model Optimization) on a larger set of CPD cases (40 vs 12) using a variety of solvers. Overall, this showed again that the CFN approach outperformed by several orders of magnitude other evaluated methods as well as the exact DEE/A* algorithm. Noteworthy, the incorporation of the suitably modified DEE algorithms enabled to further improve the results. The results are also included in this current chapter under the form of an article published in the *Artificial Intelligence* journal [2].

Computational Protein Design as an Optimization Problem

David Allouche, Jessica Davies, Simon de Givry, George Katsirelos, Thomas Schiex*

UBIA, UR-875, INRA, F-31320 Castanet Tolosan, France

Seydou Traoré, Isabelle André, Sophie Barbe

LISBP, INSA, UMR INRA 792/CNRS 5504, F-31400 Toulouse, France

Steve Prestwich, Barry O'Sullivan

Cork Constraint Computation Centre, University College Cork, Ireland

Abstract

Proteins are chains of simple molecules called amino acids. The three-dimensional shape of a protein and its amino acid composition define its biological function. Over millions of years, living organisms have evolved a large catalog of proteins. By exploring the space of possible amino acid sequences, protein engineering aims at similarly designing tailored proteins with specific desirable properties. In Computational Protein Design (CPD), the challenge of identifying a protein that performs a given task is defined as the combinatorial optimization of a complex energy function over amino acid sequences.

In this paper, we introduce the CPD problem and some of the main approaches that have been used by structural biologists to solve it, with an emphasis on the exact method embodied in the dead-end elimination/A* algorithm (DEE/A*). The CPD problem is a specific form of binary Cost Function Network (CFN, aka Weighted CSP). We show how DEE algorithms can be incorporated and suitably modified to be maintained during search, at reasonable computational cost.

We then evaluate the efficiency of CFN algorithms as implemented in our solver `toulbar2`, on a set of real CPD instances built in collaboration with structural biologists. The CPD problem can be easily reduced to 0/1 Linear Programming, 0/1 Quadratic Programming, 0/1 Quadratic Optimization, Weighted Partial MaxSAT and Graphical Model optimization problems. We compare `toulbar2` with these different approaches using a variety of solvers. We observe tremendous differences in the difficulty that each approach has on these instances.

Overall, the CFN approach shows the best efficiency on these problems, improving by several orders of magnitude against the exact DEE/A* approach. The introduction of dead-end elimination before or during search allows to further improve these results.

Keywords: weighted constraint satisfaction problem, soft constraints, neighborhood substitutability, constraint optimization, graphical model, cost function networks, integer linear programming, quadratic programming, computational protein design, bioinformatics, maximum a posteriori inference, maximum satisfiability

*Corresponding author

1. Introduction

A protein is a sequence of basic building blocks called *amino acids*. Proteins are involved in nearly all structural, catalytic, sensory, and regulatory functions of living systems [26]. Performing these functions generally requires that proteins are assembled into well-defined three-dimensional structures specified by their amino acid sequence. Over millions of years, natural evolutionary processes have shaped and created proteins with novel structures and functions by means of sequence variations, including mutations, recombinations and duplications. Protein engineering techniques coupled with high-throughput automated procedures make it possible to mimic the evolutionary process on a greatly accelerated time-scale, and thus increase the odds to identify the proteins of interest for technological uses [71]. This holds great interest for medicine, synthetic biology, nanotechnologies and biotechnologies [67, 75, 39]. In particular, protein engineering has become a key technology to generate tailored enzymes able to perform novel specific transformations under specific conditions. Such biochemical transformations enable to access a large repertoire of small molecules for various applications such as biofuels, chemical feedstocks and therapeutics [45, 11]. The development of enzymes with required substrate selectivity, specificity and stability can also be profitable to overcome some of the difficulties encountered in synthetic chemistry. In this field, the *in vitro* use of artificial enzymes in combination with organic chemistry has led to innovative and efficient routes for the production of high value molecules while meeting the increasing demand for ecofriendly processes [61, 13]. Nowadays, protein engineering is also being explored to create non-natural enzymes that can be combined in vivo with existing biosynthetic pathways, or be used to create entirely new synthetic metabolic pathways not found in nature to access novel biochemical products [28]. These latest approaches are central to the development of synthetic biology. One significant example in this field is the full-scale production of the antimalarial drug (artemisinin) from the engineered bacteria *Escherichia coli* [66].

With a choice among 20 naturally occurring amino acids at every position, the size of the combinatorial sequence space is out of reach for current experimental methods, even for short sequences. Computational protein design (CPD) methods therefore try to intelligently guide the protein design process by producing a *collection* of proteins, that is rich in functional proteins, but small enough to be experimentally evaluated. The challenge of choosing a sequence of amino acids to perform a given task is formulated as an optimization problem, solvable computationally. It is often described as the inverse problem of protein folding [70]: the three-dimensional structure is known and we have to find amino acid sequences that fold into it. It can also be considered as a highly combinatorial variant of side-chain positioning [82] because of possible amino acid mutations.

Various computational methods have been proposed over the years to solve this problem and several success stories have demonstrated the outstanding potential of CPD methods to engineer proteins with improved or novel properties. CPD has been successfully applied to increase protein thermostability and solubility; to alter specificity towards some other molecules; and to design various binding sites and construct *de novo* enzymes (see for example [46]).

Despite these significant advances, CPD methods must still mature in order to better guide and accelerate the construction of tailored proteins. In particular, more efficient computational optimization techniques are needed to explore the vast combinatorial space, and to facilitate the incorporation of more realistic, flexible protein models. These methods need to be capable of not only identifying the optimal model, but also of enumerating solutions close to the optimum.

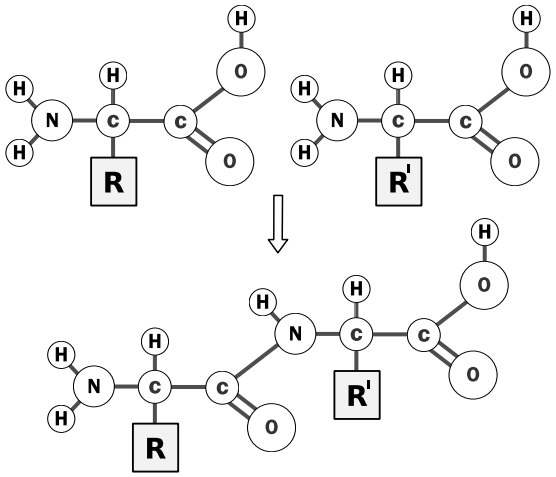


Figure 1: A representation of how amino acids, carrying specific side chains R and R', can link together through their core to form a chain (modified from wikipedia). One molecule of water is also generated in the process.

We begin by defining the CPD problem with rigid backbone, and then introduce the approach commonly used in structural biology to exactly solve CPD. This approach relies on dead-end elimination (DEE), a specific form of dominance analysis that was introduced in [24], and later strengthened in [37]. If this polynomial-time analysis does not solve the problem, an A^* algorithm is used to identify an optimal protein design.

We observe that the rigid backbone CPD problem can be naturally expressed as a Cost Function Network (aka Weighted Constraint Satisfaction Problem). In this context, DEE is similar to neighbourhood substitutability [27]. We show how DEE can be suitably modified so as to be maintained during search at reasonable computational cost, in collaboration with the usual soft local consistencies.

To evaluate the efficiency of the CFN approach, we model the CPD problem using several combinatorial optimization formalisms. We compare the performance of the 0/1 linear programming and 0/1 quadratic programming solver `cplex`, the semidefinite programming based Boolean quadratic optimization tool `biqmac`, several weighted partial MaxSAT solvers, the Markov random field optimization solvers `daoopt` and `mplp` [80], and the CFN solver `toulbar2`, against that of a well-established CPD approach implementing DEE/A^*, on various realistic protein design problems. We observe drastic differences in the difficulty that these instances represent for different solvers, despite often closely related models and solving techniques.

2. The Computational Protein Design approach

A protein is a sequence of organic compounds called amino acids. All amino acids consist of a common *peptidic core* and a *side chain* with varying chemical properties (see Figure 1). In a protein, amino acid cores are linked together in sequence to form the *backbone* of the protein. A given protein *folds* into a 3D shape that is determined from the sequence of amino acids. Depending upon the amino acid considered, the side chain of each individual amino acid can be rotated along up to 4 dihedral angles relative to the backbone. After Anfinsen's work [3], the 3D structure of a protein can be considered to be defined by the backbone and the set of side-chain rotations. This is called the *conformation* of the protein and it determines its chemical reactivity and biological function.

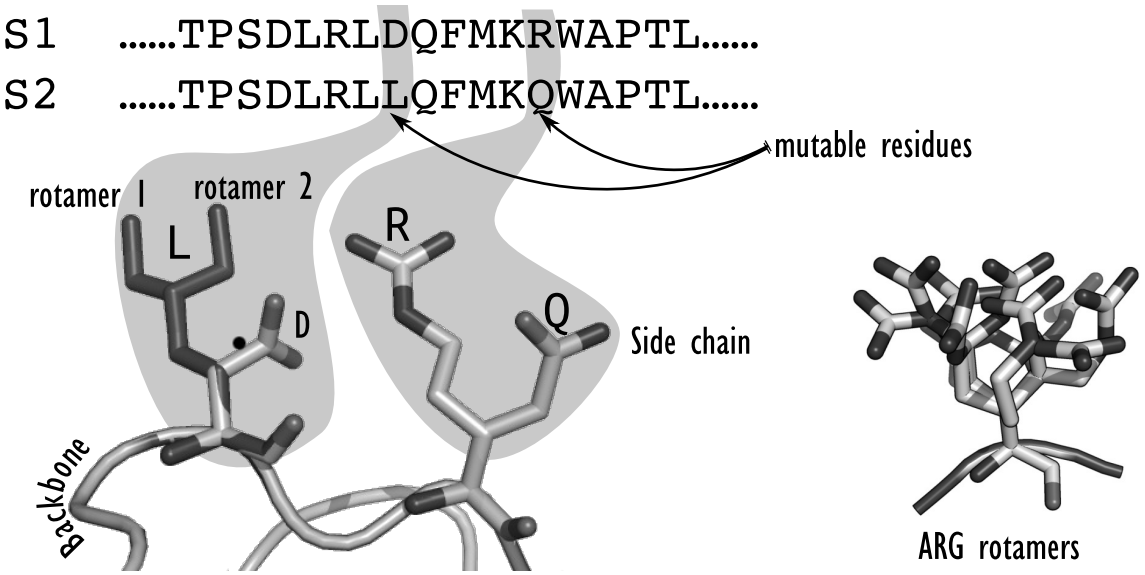


Figure 2: A local view of combinatorial sequence exploration considering a common backbone. Changes can be caused by amino acid identity substitutions (*for example* D/L or R/Q) or by amino acid side-chain reorientations (rotamers) for a given amino acid. A typical rotamer library for one amino acid is shown on the right (ARG=Arginine).

Computational Protein Design is faced with several challenges. The first lies in the exponential size of the conformational and protein sequence space that has to be explored, which rapidly grows out of reach of computational approaches. Another obstacle to overcome is the accurate structure prediction for a given sequence [47, 38]. Therefore, the design problem is usually approached as an inverse folding problem [70], in order to reduce the problem to the identification of an amino acid sequence that can fold into a target 3D-scaffold that matches the design objective [9]. In structural biology, the stability of aconformation can be directly evaluated through the energy of the conformation, a stable fold being of minimum energy [3].

In CPD, two approximations are common. First, it is assumed that the resulting designed protein retains the overall folding of the chosen scaffold: the protein *backbone* is considered fixed. At specific positions chosen by the computational biologist (or automatic selection), the amino acid can be modified by changing the *side chain* as shown in Fig. 2. Second, the domain of conformations available to each amino acid side chain is actually continuous. This continuous domain is approximated using a set of discrete conformations defined by the value of their inner dihedral angles. These conformations, or *rotamers* [44], are derived from the most frequent conformations in the experimental repository of known protein structures, PDB (Protein Data Bank, `www.wwpdb.org`). Different discretizations have been used in constraint-based approaches to protein structure prediction [10].

The CPD is then formulated as the problem of identifying a conformation of minimum energy via the mutation of a specific subset of amino acid residues, i.e. by affecting their identity and their 3D orientations (rotamers). The conformation that minimizes the energy is called the *GMEC* (Global Minimum Energy Conformation).

In order to solve this problem, we need a computationally tractable energetic model to evaluate the energy of any combination of rotamers. We also require computational optimization techniques that can efficiently explore the sequence-conformation space to find the sequence-conformation model of global minimum energy.

Energy functions. Various energy functions have been defined to make the energy computation manageable [7]. These energy functions include non-bonded terms such as van der Waals and electrostatics terms, often in conjunction with empirical contributions describing hydrogen bonds. The surrounding solvent effect is generally treated implicitly as a continuum. Statistical terms may be added in order to approximate the effect of mutations on the unfolded state or the contribution of conformational entropy. Finally, collisions between atoms (steric clashes) are also taken into account. In this work, we used the state-of-the-art energy functions implemented in the CPD dedicated tool `osprey` 2.0 [30].

These energy functions can be reformulated in such a way that the terms are locally decomposable. Then, the energy of a given protein conformation, defined by a choice of one specific amino acid with an associated conformation (rotamer) for each residue, can be written as:

$$E = E_{\varnothing} + \sum_{i} E(i_r) + \sum_{i} \sum_{j>i} E(i_r, j_s) \tag{1}$$

where E is the potential energy of the protein, $E_{\varnothing}$ is a constant energy contribution capturing interactions between fixed parts of the model, $E(i_r)$ is the energy contribution of rotamer r at position i capturing internal interactions (and a reference energy for the associated amino acid) or interactions with fixed regions, and $E(i_r, j_s)$ is the pairwise interaction energy between rotamer r at position i and rotamer s at position j [24]. This decomposition brings two properties:

- Each term in the energy can be computed for each amino acid/rotamer (or pair for $E(i_r, j_s)$) independently.
- These energy terms, in $kcal/mol$, can be precomputed and cached, allowing to quickly compute the energy of a design once a specific rotamer (an amino acid-conformation pairing) has been chosen at each non-rigid position.

The rigid backbone dicrete rotamer Computational Protein Design problem is therefore defined by a fixed backbone with a corresponding set of positions (residues), a rotamer library and a set of energy functions. Each position i of the backbone is associated with a subset D_i of all (amino-acid,rotamer) pairs in the library. The problem is to identify at each position i a pair from D_i such that the overall energy E is minimized. In practice, based on expert knowledge or on pecific design protocols, each position can be fixed (D_i is a singleton), flexible (all pairs in D_i have the same amino-acid) or mutable (the general situation).

2.1. Exact CPD methods

The protein design problem as defined above, with a rigid backbone, a discrete set of rotamers, and pairwise energy functions has been proven to be NP-hard [74]. Hence, a variety of meta-heuristics have been applied to it, including Monte Carlo simulated annealing [53], genetic algorithms [77], and other algorithms [25]. The main weakness of these approaches is that they may remain stuck in local minima and miss the GMEC without notice.

However, there are several important motivations for solving the CPD problem exactly. First, because they know when an optimum is reached, exact methods may stop before meta-heuristics. Voigt et al. [84] reported that the accuracy of meta-heuristics also degrades as problem size increases. More importantly, the use of exact search algorithms

becomes crucial in the usual experimental design cycle, that goes through modelling, solving, protein synthesis and experimental evaluation: when unexpected experimental results are obtained, the only possible culprit lies in the CPD model and not in the algorithm.

Current exact methods for CPD mainly rely on the dead-end elimination (DEE) theorem [24, 19] and the A^* algorithm [58, 33]. DEE is used as a pre-processing technique and removes rotamers that are locally dominated by other rotamers, until a fixpoint is reached. The rotamer r at position i (denoted by i_r) is removed if there exists another rotamer u at the same position such that [24]:

$$E(i_r) + \sum_{j \neq i} \min_s E(i_r, j_s) \geq E(i_u) + \sum_{j \neq i} \max_s E(i_u, j_s) \tag{2}$$

This condition guarantees that for any conformation with this r, we get a conformation with lower energy if we substitute u for r. Then, r can be removed from the list of possible rotamers at position i. This local dominance criterion was later improved by Goldstein [37] by directly comparing energies of each rotamer in the same conformation:

$$E(i_r) - E(i_u) + \sum_{j \neq i} \min_s [E(i_r, j_s) - E(i_u, j_s)] \geq 0 \tag{3}$$

where the best and worst-cases are replaced by the worst difference in energy. It is easy to see that this condition is always weaker than the previous one, and therefore applicable to more cases. These two properties define polynomial time algorithms that prune dominated values.

Since its introduction in 1992 by Desmet, DEE has become the fundamental tool of exact CPD, and various extensions have been proposed [73, 63, 32]. All these DEE criteria preserve the optimum but may remove suboptimal solutions. However CPD is NP-hard, and DEE cannot solve all CPD instances. Therefore, DEE pre-processing is usually followed by an A^* search. After DEE pruning, the A^* algorithm allows to expand a sequence-conformation tree, so that sequence-conformations are extracted and sorted on the basis of their energy values. The admissible heuristic used by A^* is described in [33].

When the DEE algorithm does not significantly reduce the search space, the A^* search tree can be too slow or memory demanding and the problem cannot be solved. Therefore, to circumvent these limitations and increase the ability of CPD to tackle problems with larger sequence-conformation spaces, novel alternative methods are needed. We now describe alternative state-of-the-art methods for solving the GMEC problem that offer attractive alternatives to DEE/A^*.

3. From CPD to CFN

CPD instances can be directly represented as Cost Function Networks.

Definition 1. *A Cost Function Network (CFN) is a pair (X, W) where $X = \{1, \ldots, n\}$ is a set of n variables and W is a set of cost functions. Each variable $i \in X$ has a finite domain D_i of values that can be assigned to it. A value $r \in D_i$ is denoted i_r. For a set of variables $S \subseteq X$, D_S denotes the Cartesian product of the domains of the variables in S. For a given tuple of values t, $t[S]$ denotes the projection of t over S. A cost function $w_S \in W$, with scope $S \subseteq X$, is a function $w_S : D_S \mapsto [0, k]$ where k is a maximum integer cost used for forbidden assignments.*

We assume, without loss of generality, that every CFN includes at least one unary cost function w_i per variable $i \in X$ and a nullary cost function $w_\varnothing$. All costs being non-negative, the value of this constant function, $w_\varnothing$, provides a lower bound on the cost of any assignment.

The Weighted Constraint Satisfaction Problem (WCSP) is to find a complete assignment t minimizing the combined cost function $\bigoplus_{w_S \in W} w_S(t[S])$, where $a \oplus b = \min(k, a+b)$ is the k-bounded addition. This optimization problem has an associated NP-complete decision problem. Notice that if $k = 1$, then the WCSP is nothing but the classical CSP (and not the Max-CSP).

Modeling the CPD problem as a CFN is straightforward. The set of variables X has one variable i per residue i. The domain of each variable is the set of *(amino acid,conformation)* pairs in the rotamer library used. The global energy function can be represented by 0-ary, unary and binary cost functions, capturing the constant energy term $w_\varnothing = E_\varnothing$, the unary energy terms $w_i(r) = E(i_r)$, and the binary energy terms $w_{ij}(r, s) = E(i_r, j_s)$, respectively. In the rest of the paper, for simplicity and consistency, we use notations $E_\varnothing$, $E(\cdot)$ and $E(\cdot, \cdot)$ to denote cost functions and restrict ourselves to binary CFN (extensions to higher orders are well-known).

Notice that there is one discrepancy between the original formulation and the CFN model: energies are represented as arbitrary floating point numbers while CFN uses positive costs. This can simply be fixed by first subtracting the minimum energy from all energies. These positive costs can then be multiplied by a large integer constant M and rounded to the nearest integer if integer costs are required.

3.1. Local consistency in CFN

The usual exact approach to solve a CFN is to use a depth-first branch-and-bound algorithm (DFBB). A family of efficient and incrementally computed lower bounds is defined by local consistency properties.

Node consistency [54] (NC) requires that the domain of every variable i contains a value r that has a zero unary cost ($E(i_r) = 0$). This value is called the unary support for i. Furthermore, in the scope of the variable i, all values should have a cost below k ($\forall r \in D_i, E_\varnothing + E(i_r) < k$).

Soft arc consistency (AC*) [79, 54] requires NC and also that every value r of every variable i has a support on every cost function $E(i_r, j_s)$ involving i. A support of i_r is a value $j_s \in D_j$ such that $E(i_r, j_s) = 0$.

Stronger local consistencies such as Existential Directional Arc Consistency (EDAC) have also been introduced [55]. See [14] for a review of existing local consistencies.

As in classical CSP, enforcing a local consistency property on a problem P involves transforming $P = (X, W)$ into a problem $P' = (X, W')$ that is equivalent to P (all complete assignments keep the same cost) and that satisfies the considered local consistency property. Enforcing a local consistency may increase $E_\varnothing$ and thus improve the lower bound on the optimal cost. This bound is used to prune the search tree during DFBB.

Local consistency is enforced using *Equivalence Preserving Transformations* (EPTs) that move costs between different cost functions [79, 54, 57, 18, 55, 15, 17, 16, 14]. For example, a variable i violating the NC property because all its values i_r have a non-zero $E(i_r)$ cost, can be made NC by subtracting the minimum cost from all $E(i_r)$ and adding this cost to $E_\varnothing$. The resulting network is equivalent to the original network, but it has an increased lower bound $E_\varnothing$.

Interestingly, in CPD, the admissible heuristic used in the DEE/A^* algorithm at depth d of the search tree is [33]:

$$\underbrace{\sum_{i=1}^{d}\Big[E(i_r)+\sum_{j=i+1}^{d}E(i_r,j_s)\Big]}_{\text{Assigned}}+\sum_{j=d+1}^{n}\Big[\underbrace{\min_{s}(E(j_s)+\sum_{i=1}^{d}E(i_r,j_s)}_{\text{Forward checking}}+\underbrace{\sum_{k=j+1}^{n}\min_{u}E(j_s,k_u))}_{\text{DAC counts}}\Big]$$

From a WCSP perspective, interpreting energies as cost functions, this heuristic is exactly the PFC-DAC lower bound [85, 56] used in WCSP. In WCSP, this lower bound is considered obsolete, and indeed it is proven to be weaker than soft arc consistency [79].

3.2. Maintaining dead-end elimination

Dead-end elimination is the key algorithmic tool of exact CPD solvers. From an AI perspective, in the context of CSP (if $k = 1$), the DEE Equation 3 is equivalent to neighborhood substitutability [27]. For MaxSAT, it is equivalent to the *Dominating 1-clause rule* [68]. In the context of CFN, the authors of [59] introduced partial soft neighborhood substitutability with a definition that is equivalent to Equation 3 for pairwise decomposed energies.

The DEE Equation 3 (cf. Section 2.1) can be strengthened and adapted to the CFN context as follows:

$$E(i_r)-E(i_u)+\sum_{j\neq i}\min_{s\ E_\varnothing+E(i_r)+E(j_s)+E(i_r,j_s)<k}[E(i_r,j_s)-E(i_u,j_s)]\geq 0 \tag{4}$$

This new condition differs from Equation 3 by the fact that some values have been discarded from the min operation. These values correspond to forbidden assignments because the sum of the corresponding binary term plus the two unary costs plus the current lower bound $E_\varnothing$ (produced by soft arc consistency) is greater than or equal to the current upper bound k. Such values s do not need to be considered by the min operation because $\{i_r, j_s\}$ does not belong to any optimal solution, whereas $\{i_u, j_s\}$ may[1].

Example 1. *Let $X = \{1, 2, 3\}$ be a set of three variables with domains $D_1 = \{a, b, c\}$, $D_2 = \{e, f\}$, and $D_3 = \{g, h\}$. Suppose there are three cost functions, where $E(1_b) = 2$, $E(1_a, 2_e) = 2$, $E(1_b, 2_e) = E(1_c, 2_f) = 1$, $E(1_a, 3_g) = E(1_c, 3_h) = 2$, and all other costs are null. Let $k = 3$. The problem is EDAC. Then, 1_a dominates 1_b as shown by the new rule of Equation 4 that is satisfied: $E(1_b) - E(1_a) + E(1_b, 2_f) - E(1_a, 2_f) + \min(E(1_b, 3_g) - E(1_a, 3_g), E(1_b, 3_h) - E(1_a, 3_h)) = 2 - 0 + 0 - 2 \geq 0$, discarding tuple $\{2_e\}$ because $E(1_b, 2_e) + E(1_b) + E(2_e) + E_\varnothing = 1 + 2 + 0 + 0 \geq k$, whereas the old rule of Equation 3 is unsatisfied: $E(1_b) - E(1_a) + \min(E(1_b, 2_e) - E(1_a, 2_e), E(1_b, 2_f) - E(1_a, 2_f)) + \min(E(1_b, 3_g) - E(1_a, 3_g), E(1_b, 3_h) - E(1_a, 3_h)) = 2 - 0 - 1 - 2 < 0$.*

In the following, we recall how to enforce Equation 4 by an immediate adaptation of the original algorithm in [59]. Then, we present a modified version to partially enforce a novel combination of Equation 4 and Equation 2 with a much lower time complexity.

[1] Depending on the definition of soft arc consistency, from [54] (as presented in Section 3.1) or from [18], Equation 4 is stronger than or equivalent to Equation 3.

3.2.1. Enforcing DEE

Assuming a soft arc consistent WCSP (see *e.g.,* W-AC*2001 algorithm in [57]), enforcing DEE is described by Algorithm 1. For each variable i, all the pairs of values $(u, r) \in D_i \times D_i$ with $u < r$ are checked by the function DominanceCheck to see if r is dominated by u or, if not, vice versa (line 3). At most one dominated value is added to the value removal queue Δ at each inner loop iteration (line 2). Removing dominated values (line 4) can make the problem arc inconsistent, requiring us to enforce soft arc consistency again. Procedure AC*-DEE successively enforces AC* and DEE until no value removal is made by the enforcing algorithms.

Function DominanceCheck(i, u, r) computes the sum of worst-cost differences as defined by Equation 4 and returns a non-empty set containing value r if Equation 4 is true, meaning that r is dominated by value u. It exploits early breaks as soon as Equation 4 can be falsified (lines 5 and 6). Worst-cost differences are computed by the function getDifference(j, i, u, r) applied to every binary cost function related to i, discarding forbidden assignments with $\{i_r, j_s\}$ (line 8), as suggested by Equation 4. Worst-cost differences are always negative or zero (line 7) due to AC*.

The worst-case time complexity of getDifference is $O(d)$ for binary WCSPs. DominanceCheck is $O(nd)$ assuming a complete graph. Thus, the time complexity of one iteration of Algorithm 1 (DEE) is $O(nd^2nd + nd) = O(n^2d^3)$. Interleaving DEE and AC* until a fixed point is reached is done at most nd times, resulting in a worst-case time complexity in $O(n^3d^4)$. Its space complexity is $O(nd^2)$ when using the *residues* structure [59].

Note that using the new Equation 4 (line 8) or the Equation 3 (without line 8) does not change the complexities.

3.2.2. Enforcing DEE[1]

In order to reduce the time (and space) complexity of pruning by dominance, we test only one pair of values per variable. Hence the name, DEE[1], for the new algorithm described in Algorithm 2. We select the pair $(u, r) \in D_i \times D_i$ in an optimistic way such that u is associated with the minimum unary cost and r to the maximum unary cost (lines 9 and 10). Because arc consistency also implies node consistency, we always have $E(i_u) = 0$.[2] If all the unary costs (including the maximum) are equal to zero (line 11), we select as r the maximum domain value (or its minimum if this value is already used by u). By doing so, we should favor more pruning on max-closed or submodular subproblems[3].

Instead of just checking the new Equation 4 for the pair (u, r) alone, we use the opportunity to also check the original DEE rule of Equation 2 for all the pairs (u, v) such that $v \in D_i \setminus \{u\}$. This is done in the function MultipleDominanceCheck (lines 15 and 16). Notice that Equation 2 simplifies to $E(i_v) \geq ub_u$ (line 16) due to AC*. This function computes at the same time the sum of maximum costs ub_u for value u (lines 12 and 13) and the sum of worst-cost differences δ_{ur} for the pair (u, r). The new function getDifference-Maximum(j, i, u, r) now returns the worst-cost difference, as suggested by Equation 4, and also the maximum cost in $E(i, j)$ for i assigned u. When the maximum cost of a value is null for all its cost functions, we can directly remove all the other values in the domain avoiding any extra work (line 14). Finally, if the selected pair (u, r) for

[2] In practice, we set the value u to the *unary support* offered by NC [54] or EDAC [55].

[3] Assuming a problem with two variables i and j having the same domain and a single submodular cost function, *e.g.,* $E(i_u, j_s) = 0$ if $u \leq s$ else $u - s$, or a single max-closed constraint, *e.g.,* $u < s$, then DEE[1] assigns $min(D_i)$ to i and $max(D_j)$ to j.

Algorithm 1: Enforce DEE [59]

```
Procedure DEE((X, W): AC* consistent WCSP)
    Δ := ∅ ;
1   foreach i ∈ X do
2       foreach (u, r) ∈ D_i × D_i such that u < r do
            R := DominanceCheck(i, u, r) ;
3           if R = ∅ then R := DominanceCheck(i, r, u) ;
            Δ := Δ ∪ R ;
4   foreach i_r ∈ Δ do
        remove r from D_i ;
        Q := Q ∪ {i} ;

/* Check if value u dominates value r */
Function DominanceCheck(i, u, r): set of dominated values
    δ_ur := E(i_r) − E(i_u) ;
5   if δ_ur < 0 then return ∅ ;
    foreach j ∈ X \ {i} do
        δ := getDifference(j, i, u, r) ;
        δ_ur := δ_ur + δ ;
6       if δ_ur < 0 then return ∅ ;
    return {i_r} /* δ_ur ≥ 0 */ ;

/* Compute smallest difference in costs when using a instead of b */
Function getDifference(j, i, u, r): cost
7   δ_ur := 0 ;
    foreach s ∈ D_j do
8       if E(i_r, j_s) + E(i_r) + E(j_s) + E_∅ < k then
            δ_ur := min(δ_ur, E(i_r, j_s) − E(i_u, j_s)) ;
    return δ_ur ;

/* Enforce AC* and DEE */
Procedure AC*-DEE()
    Q := X ;
    while Q ≠ ∅ do
        W-AC*2001(Q) ;
        DEE(Q) ;
```

the variable i satisfies Equation 4, removing the value r of D_i, then a new pair for i will be checked at the next iteration of Algorithm 2 in the modified procedure AC*-DEE^1 (replacing Algorithm 1 by Algorithm 2 in AC*-DEE).

Notice that DEE^1 is equivalent to DEE on problems with Boolean variables, such as MaxSAT. For problems with non-Boolean domains, DEE^1 is still able to detect and prune several values per variable. Clearly, its time (resp. space) complexity is $O(n^3d^2)$ (resp. $O(n)$ using only one residue per variable), reducing by a factor d^2 the time and space complexity compared to DEE.

Algorithm 2: Enforce DEE^1

Procedure DEE^1((X, W): AC* consistent WCSP)
  $\Delta := \emptyset$;
  foreach $i \in X$ **do**
9 $u := \arg\min_{v \in D_i} E(i_v)$;
10 $r := \arg\max_{v \in D_i} E(i_v)$;
11 **if** $u = r$ /* $\forall v \in D_i, E(i_v) = 0$ */ **then**
      if $u = max(D_i)$ **then**
        $r := min(D_i)$;
      else
        $r := max(D_i)$;
    $R :=$ MultipleDominanceCheck(i, u, r) ;
    if $R = \emptyset$ **then** $R :=$ MultipleDominanceCheck(i, r, u) ;
    $\Delta := \Delta \cup R$;
  foreach $i_r \in \Delta$ **do**
    remove r from D_i ;
    $Q := Q \cup \{i\}$;

/* Check if value u dominates value r and possibly other values */
Function MultipleDominanceCheck(i, u, r): set of dominated values
  $\delta_{ur} := E(i_r) - E(i_u)$;
  if $\delta_{ur} < 0$ **then return** $\emptyset$;
12 $ub_u := E(i_u)$;
  foreach $j \in X \setminus \{i\}$ **do**
    $(\delta, ub) :=$ getDifference-Maximum(j, i, u, r) ;
    $\delta_{ur} := \delta_{ur} + \delta$;
13    $ub_u := ub_u + ub$;
    if $\delta_{ur} < 0$ **then return** $\emptyset$;
14 **if** $ub_u = 0$ **then return** $\{i_v | v \in D_i \setminus \{u\}\}$;
  $R := \{i_r\}$ /* $\delta_{ur} \geq 0$ */ ;
15 **foreach** $v \in D_i \setminus \{u\}$ **do**
16   **if** $(E(i_v) \geq ub_u)$ **then** $R := R \cup \{i_v\}$;
  return R ;

/* Compute smallest cost difference and maximum cost for value u */
Function getDifference-Maximum(j, i, u, r): pair of costs
  $\delta_{ur} := 0$;
  $ub_u := 0$;
  foreach $s \in D_j$ **do**
    if $E(i_r, j_s) + E(i_r) + E(j_s) + E_\varnothing < k$ **then**
      $\delta_{ur} := \min(\delta_{ur}, E(i_r, j_s) - E(i_u, j_s))$;
    $ub_u := \max(ub_u, E(i_u, j_s))$;
  return (δ_{ur}, ub_u) ;

/* Enforce AC* and DEE^1 */
Procedure AC*-DEE^1()
  $Q := X$;
  while $Q \neq \emptyset$ **do**
    W-AC*2001(Q) ;
    DEE^1(Q) ;

4. Computational Protein Design instances

In our initial experiments with CPD in [2], we built 12 designs using the CPD dedicated tool `osprey` 1.0. A new version of `osprey` being available since, we used this new 2.0 version [30] for all computations. Among different changes, this new version uses a modified energy field that includes a new definition of the "reference energy" and a different rotamer library. We therefore rebuilt the 12 instances from [2] and additionally created 35 extra instances from existing published designs, as described in [83]. We must insist on the fact that the 12 rebuilt instances do not define the same energy landscape or search space as the initial [2]'s instances (due to changes in rotamers set).

These designs include protein structures derived from the PDB that were chosen for the high resolution of their 3D-structures, their use in the literature, and their distribution of sizes and types. Diverse sizes of sequence-conformation combinatorial spaces are represented, varying by the number of mutable residues, the number of alternative amino acid types at each position and the number of conformations for each amino acid. The *Penultimate* rotamer library was used [64]. Over these 47 designs, we only report results on the 40 designs for which a GMEC could be identified and proven by one of the tested solvers. All 47 designs are available for download both in native and WCSP formats at `http://genotoul.toulouse.inra.fr/~tschiex/CPD-AIJ`.

Preparation of CPD instances. Missing heavy atoms in crystal structures and hydrogen atoms were added with the *tleap* module of the AMBER9 software package [12]. Each molecular system was then minimized in implicit solvent (Generalized Born model [42]) using the *Sander* program and the all-atom *ff99* force field of AMBER9. All $E_{\varnothing}$, $E(i_r)$, and $E(i_r, j_s)$ energies of rotamers (see Equation 1) were pre-computed using `osprey` 2.0. The energy function consisted of the Amber electrostatic, van der Waals and the solvent terms. Rotamers and rotamer pairs leading to sterical clashes between molecules are associated with huge energies (10^{38}) representing forbidden combinations. For n residues to optimize with d possible (amino acid,conformation) pairs, there are n unary and $\frac{n.(n-1)}{2}$ binary cost functions that can be computed independently.

Translation to WCSP format. The native CPD problems were translated to the WCSP format before any pre-processing. To convert the floating point energies of a given instance to non-negative integer costs, we subtracted the minimum energy to all energies and then multiplied energies by an integer constant M and rounded to the nearest integer. The initial upper bound k is set to the sum, over all cost functions, of the maximum energies (excluding forbidden sterical clashes). High energies corresponding to sterical clashes are represented as costs equal to the upper bound k (the forbidden cost). The resulting WCSP model was used as the basis for all other solvers (except `osprey`). To keep a cost magnitude compatible with all the compared solvers, we used $M = 10^2$. Experiments with a finer discretization ($M = 10^8$) was used in previous experiments [83] with no significant difference in computing efforts.

4.1. A new cost-based variable ordering heuristics

We analyzed the distribution of costs for the CPD problem in order to infer a new variable ordering heuristics. Figure 3-left shows the histogram of a typical binary cost function for one of the CPD instances (1ENH, one of the open instances). Although the distribution has several modes, we chose to collect as an *important feature of a cost function* its median cost, which is less sensitive to extrema than the mean cost.

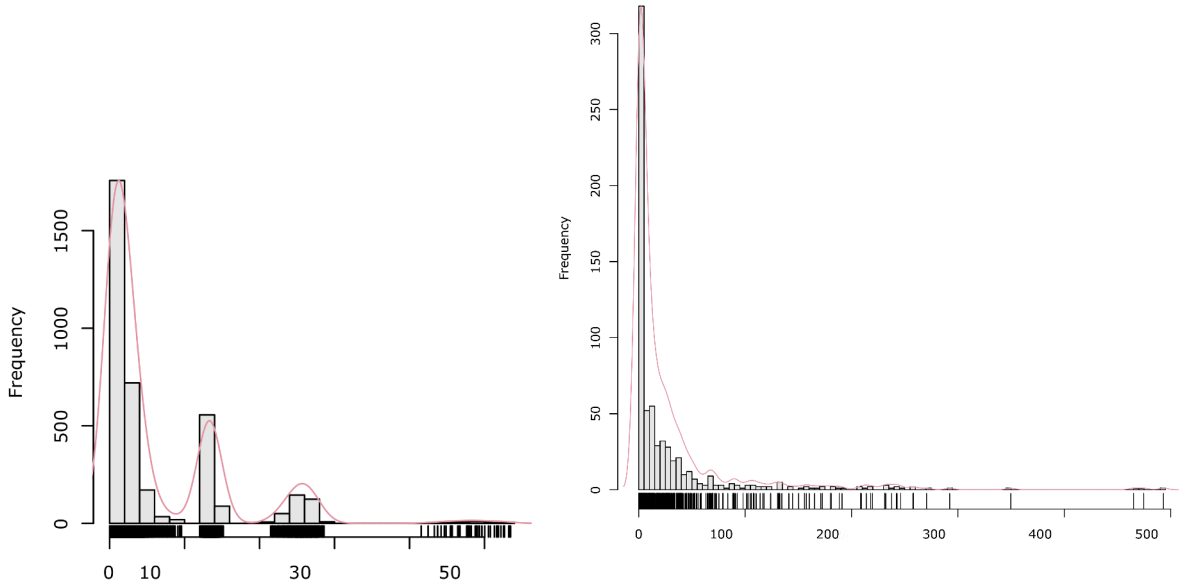


Figure 3: Histograms of $E(1, 12)$ costs (left) and of median costs of all binary cost functions (right) for the 1ENH instance ($M = 10^2$).

Figure 3-right shows the histogram of median costs for this instance. The problem has 666 binary cost functions and we collected the median cost in every cost function. The distribution of median costs has a heavy right tail. This feature can be exploited during search to focus on the most important variables first. For that, we define a new dynamic variable ordering heuristics selecting at each node of the search tree the variable minimizing the ratio of its current domain size divided by the sum of the median costs of all its current cost functions (including its unary cost function). The sum of the median costs gives a rough estimate of the average lower bound increase if we select that variable, relating this heuristics to *strong branching* in Operations Research [62, 1]. In order to save computation time, median costs of binary cost functions are computed only once, just after enforcing EDAC (and DEE), before the search.

5. Alternative models for the CPD

The rigid backbone CPD problem has a simple formulation and can be easily written in a variety of combinatorial optimization frameworks. To evaluate CFN algorithms, the new DEE[1] algorithm and our domain specific heuristics, we compared these different variants with a variety of other solvers, coming from different fields. We present now the different models used in the comparison.

5.1. CPD as a probabilistic graphical model

The notion of *graphical model* has been mostly associated with *probabilistic graphical models*, the most famous examples of these are Markov random fields and Bayesian networks [49]. In those formalisms, a concise description of a joint distribution of probabilities over a set of variables is obtained through a factorization in local terms, involving only few variables. For terms involving at most two variables, if vertices represent variables and edges represent terms, a factorization can be represented as a graph, hence the

name of *graphical models*. The same idea is used for concisely describing set of solutions (relations) in CSP or cost distributions in CFN.

Definition 2. *A discrete Markov random field (MRF) is a pair* (X, Φ) *where* $X = \{1, \ldots, n\}$ *is a set of* n *random variables and* Φ *is a set of potential functions. Each variable* $i \in X$ *has a finite domain* D_i *of values that can be assigned to it. A potential function* $\phi_S \in \Phi$*, with scope* $S \subseteq X$*, is a function* $\phi_S : D_S \mapsto \mathbb{R}$*.*

A discrete Markov random field (MRF) implicitly defines a non-normalized probability distribution over X. For a given tuple t, the probability of t is defined as:

$$P(t) = \frac{\exp(-\sum_{\phi_S \in \Phi} \phi_S(t[S]))}{Z}$$

where Z is a normalizing constant.

From the sole point of view of optimization, the problem of finding an assignment of maximum probability, also known as the maximum a posteriori (MAP) assignment in a MRF or a minimum cost solution of a CFN (the Weighted CSP) are equivalent by monotonicity of the exp() function. Some technical differences remain: CFN are restricted to non-negative costs (and some tools are restricted to integer costs). Being focused on optimization, CFN also emphasizes the possible existence of a finite upper bound k that leads to the use of bounded addition to combine costs instead of plain addition of potentials in MRFs.

The CPD problem can therefore directly be modeled as the MAP problem in a MRF exactly as we have described for CFN before, additive using potentials to capture energies (see for example [86]). Combinations of values with cost k (forbidden) are mapped to an infinite additive potential or a 0 value if multiplicative (exponential) potentials are used.

These models can be solved using MAP-MRF solvers such as `daoopt` [69] (winner of the Pascal Inference Challenge in 2011[4]) or the recent version of the `mplp` [80] solver.

5.2. Integer linear programming model

A 0/1 linear programming (01LP) problem is defined by a linear criterion to optimize over a set of Boolean variables under a conjunction of linear equalities and inequalities. The previous optimization problem over a graphical model can also be represented as a 01LP problem using the encoding proposed in [51].

For every assignment i_r of every variable i, there is a Boolean variable d_{ir} that is equal to 1 iff $i = r$. Additional constraints enforce that exactly one value is selected for each variable. For every pair of values of different variables (i_r, j_s) involved in a binary energy term, there is a Boolean variable p_{irjs} that is equal to 1 iff the pair (i_r, j_s) is used. Constraints enforce that a pair is used iff the corresponding values are used. Then, finding a GMEC reduces to the following ILP:

[4]See `http://www.cs.huji.ac.il/project/PASCAL/`.

$$\min \sum_{\substack{i,r \\ E(i_r)\neq k}} E(i_r).d_{ir} + \sum_{\substack{i,r,j,s \\ j>i, E(i_r,j_s)\neq k}} E(i_r,j_s).p_{irjs}$$

$$\text{s.t.} \quad \sum_r d_{ir} = 1 \quad (\forall i) \tag{5}$$

$$\sum_s p_{irjs} = d_{ir} \quad (\forall i,r,j) \tag{6}$$

$$d_{ir} = 0 \quad (\forall i,r) E(i_r) = k \tag{7}$$

$$p_{irjs} = 0 \quad (\forall i,r,j,s) E(i_r,j_s) = k \tag{8}$$

$$d_{ir} \in \{0,1\} \quad (\forall i,r) \tag{9}$$

$$p_{irjs} \in \{0,1\} \quad (\forall i,r,j,s) \tag{10}$$

This model is also the ILP model IP1 proposed in [48] for side-chain positioning. It has a quadratic number of Boolean variables. Constraints (7) and (8) explicitely forbid values and pairs with cost k (sterical clashes).

This model can be simplified by relaxing the integrality constraint on the p_{irjs}: indeed, if all d_{ir} are set to 0 or 1, the constraints (5) and (6) enforce that the p_{irjs} are set to 0 or 1. The same observation has been previously done for in the context of the linearization of a quadratic optimization model for wind farm design in [87]. In the rest of the paper, except where it is otherwise mentioned, we relax constraint (10). This type of ILP model can be handled by various ILP solvers such as IBM ILOG `cplex`.

5.3. 0/1 quadratic programming model

A 01QP problem is defined by a quadratic criterion to optimize over a set of Boolean variables under a conjunction of linear equality and inequality constraints. A compact encoding of the problem can be obtained using the ability of expressing the product of Boolean variables, getting rid of a quadratic number of p_{irjs} variables of the 01LP model.

For every value i_r, there is again a Boolean variable d_{ir} that is equal to 1 iff $i = r$. Additional linear constraints enforce that exactly one value is selected for each variable. The use of a given pair of rotamers at positions (i_r, j_s) can then be simply captured by the product $d_{ir}.d_{js}$. Then, finding a GMEC reduces to the following compact QP:

$$\min \sum_{i,r} E(i_r).d_{ir} + \sum_{\substack{i,r,j,s \\ j>i}} E(i_r,j_s).d_{ir}.d_{js}$$

$$\text{s.t.} \quad \sum_r d_{ir} = 1 \quad (\forall i)$$

$$d_{ir} \in \{0,1\} \quad (\forall i,r)$$

$$d_{ir} = 0 \quad (\forall i,r) E(i_r) = k \tag{11}$$

$$d_{ir} + d_{js} \leq 1 \quad (\forall i,r,j,s) E(i_r,j_s) = k \tag{12}$$

Values and pairs generating sterical clashes are explicitely forbidden by constraints (11) and (12). This model can be handled by the QP solver of IBM ILOG CPLEX.

5.4. 0/1 quadratic optimization model

Another compact model can be obtained in the more restricted case of pure Boolean Quadratic Optimization (BQO), where a quadratic criterion is optimized but no linear constraints can be expressed.

For every value i_r, there is again a Boolean variable d_{ir} that is equal to 1 iff $i = r$. We must integrate the fact that exactly one value must be selected in each domain in the criterion itself. To capture the fact that there is at most one value selected per domain, we penalize the simultaneous selection of every pair i_r, i_s of rotamers of the same variable i with a sufficiently large penalty M. To guarantee that at least one value will be selected in each domain, we shift all finite energies by a constant negative term N such that all shifted finite energies are strictly negative. If an assignment selects no value in a given domain, then selecting one value can only result in an assignment with a lower cost, by introducing new negative terms in the global energy. An optimal solution must therefore contain exactly one value per domain.

The corresponding model can be written as:

$$\min \sum_{i,r} (E(i_r) - N).d_{ir} + \sum_{\substack{i,r,j,s \\ j>i}} (E(i_r, j_s) - N).d_{ir}.d_{js} + \sum_{\substack{i,r,s \\ s>r}} M.d_{ir}.d_{is}$$

For N,we just use the largest negative integer that is strictly below the opposite of the largest finite energy in a given instance. M must be chosen in such a way that no combination of energy can compensate for the cost M. The selection of one additional value i_r can just contribute to the criterion by the addition of the energy $E(i_r)$ and the energies $E(i_r, j_s)$ for all other variables j and their rotamers j_s. M is therefore set to the opposite of the largest negative integer below the most negative sum of these energies, overall all variables i and rotamers i_r.

The corresponding Boolean quadratic optimization problem can be solved using the semidefinite programming based exact best-first branch-and-bound solver `biqmac` [78].

5.5. Weighted partial MaxSAT

Definition 3. *A weighted partial MaxSAT (WPMS) instance is a set of pair $\langle C, w \rangle$, where C is a clause and w is a number in $\mathbb{N} \cup \{\infty\}$, which is called the* weight *of that clause. A clause is a disjunction of literals. A literal is a Boolean variable or its negation.*

If the weight of a clause is ∞, it is called a *hard* clause, otherwise it is a *soft* clause. The objective is to find an assignment to the variables appearing in the clauses such that all hard clauses are satisfied and the weight of all falsified soft clauses is minimized.

The CPD problem can be encoded into a WPMS instance. We present two encodings, which are based on existing translations of CSP into SAT: the *direct* encoding [5], which is closer to the CFN model, and the *tuple* encoding, which was presented but not named by Bacchus [6] and is quite similar to the ILP model.

Direct encoding. In the direct encoding, we have one proposition d_{ir} for each variable/value pair (i, r), which is true if variable i is assigned the value r. We have hard clauses $(\neg d_{ir} \vee \neg d_{is})$ for all $i \in [1, n]$ and all $r < s$, $r, s \in D_i$, as well as a hard clause $(\bigvee_r d_{ir})$ for all i. These clauses ensure that the propositional encoding of the CFN allows exactly one value for each variable. The cost functions are represented respectively by an empty clause with weight $E_\varnothing$, unit clauses $\neg d_{ir}$ with weight $E(i_r)$ and binary clauses $\neg d_{ir} \vee \neg d_{js}$ with weight $E(i_r, j_s)$.

Tuple encoding. The tuple encoding encodes variable domains the same way as the direct encoding, therefore we have a proposition d_{ir} for each variable/value pair $i = r$, along with clauses that enforce that each variable is assigned exactly one value. The constant and unary energy terms are also respectively represented as an empty soft clause with weight $E_\varnothing$ and soft unit clauses $\neg d_{ir}$ with weight $E(i_r)$.

For all non-zero pairwise energy term $E(i_r, j_s)$, we have a proposition p_{irjs} as well as the soft clause $(\neg p_{irjs})$ with weight $E(i_r, j_s)$. This represents the cost to pay if the corresponding pair (energy term) is used. We also have the hard clauses $(d_{ir} \vee \neg p_{irjs})$ and $(d_{js} \vee \neg p_{irjs})$. These enforce that if a pair is used, the corresponding values must be used. Finally, for all the pairs of variables (i, j) and all the values i_r, hard clauses $(\neg d_{ir} \vee \bigvee_{s \in D_j} p_{irjs})$ enforce that if a value i_r is used, one of the pair $p_{irj\cdot}$ must be used.

This encoding is similar to the 01LP encoding and was originally proposed in the context of SAT encodings for classical CSP [6]. Unit Propagation (UP) on the tuple encoding enforces arc consistency in the original CSP (the set of values that are deleted by enforcing AC have their corresponding literal set to false by UP).

5.6. Constraint programming model

In [72], a generic translation of WCSPs into crisp CSPs with extra cost variables has been proposed. In this transformation, the decision variables remain the same as in the original WCSP and every cost function is reified into a constraint, which applies on the original cost function scope augmented by one extra variable representing the assignment cost. This reification of costs into domain variables transforms a WCSP in a crisp CSP with more variables and augmented arities. Typically, unary and binary cost functions are converted into `table` constraints of arity two and three respectively. Another extra cost variable encodes the global GMEC criterion, related by a `sum` constraint to all the unary and binary cost variables. All the cost variables are positive integer bounded by the same initial upper bound k as in the WCSP format.

The resulting CSP model has been expressed in the `minizinc` [65] constraint programming (CP) language. It can be solved using any CP solvers such as `gecode` or `mistral`.

6. Experimental results

For computing the GMEC, all computations were performed on a single core of an AMD Operon 6176 at 2.3 GHz, 128 GB of RAM, and a 9,000-second time-out. These computations were performed on the GenoToul cluster.

6.1. Solvers tested

The solvers tested have different configurability in terms of parameters. Solvers such as `mplp` offer essentially no tuning, while others offer a large number of options. SAT solvers that participate routinely in the SAT competition have excellent default settings and those settings were kept unmodified. For one solver that explicitly requires tuning, we contacted the author for some advice. There is always a question whether dramatically different results could be obtained by different settings. The situation here corresponds to the situation of a non-naive user faced with several optimization tools.

DEE/A optimization.* The underlying principles of DEE/A* have been described in Section 2.1. To solve the different protein design cases, we used `osprey` version 2.0 (`cs.duke.edu/donaldlab/osprey.php`). The procedure starts by extensive DEE preprocessing (*algOption* = 3, includes simple Goldstein, Magic bullet pairs, 1 and 2-split

positions, Bounds and pairs pruning) followed by A^* search. Only the GMEC conformation is generated by A^* ($initEw$=0).

CFN solver. `toulbar2` is a depth-first branch-and-bound solver using soft local consistencies for bounding and specific variable and value ordering heuristics for efficiency. The default EDAC [55] consistency may simultaneously reformulate all the cost functions involving one variable (a star subgraph). The default variable ordering strategy is based on the Weighted Degree heuristics [8] with Last Conflict [60], while the default value ordering consists in choosing for each variable its *fully supported value* as defined by EDAC.

We used `toulbar2` version 0.9.6 (`mulcyber.toulouse.inra.fr/projects/toulbar2/`) using binary branching and an initial limited discrepancy search phase [41] with discrepancy less than or equal to 1. We tested this vanilla version (options `-d: -l=1 -dee=0`) and incrementally introduced our new cost-based variable ordering heuristics (option `-m`) and different levels of DEE processing: maintaining DEE[1] during search (`-dee=1`), preprocessing with DEE (`-dee=4`), both together (`-dee=2`), or maintaining DEE during search (`-dee=3`).

daoopt *solver.* We decided to include `daoopt` as the winning solver of the 2011 PASCAL probabilistic inference challenge in the ''MAP" category. We downloaded `daoopt` version 1.1.2 from its repository (`https://github.com/lotten/daoopt`) and contacted the author for some advice. The distributed version of `daoopt` is not the same as the PIC challenge version. It lacks the Dual Decomposition bound strengthening component [69] that relies on private code.

This solver relies on Stochastic Local Search for finding initial solutions followed by depth-first AND/OR search [22] and mini-bucket lower bounds [23] for pruning. Mini-bucket lower bounds require space and time in $O(d^i)$ (where i is a user controlled parameter). CPD is certainly not an ideal domain for `daoopt`: the complete graph makes AND/OR search useless and the large maximum domain size d makes mini-buckets space and time intensive. We used the ''1 hour" settings for the PIC challenge from [69], modified to account for the complete graph that makes optimization of the AND/OR decomposition useless. This leads to the parameters `-i 35 --slsX 10 --slsT 6 -lds 1` and tried to allocate different amounts of memory to mini-buckets (option `-m` with 500MB, 5GB or 50GB), the i parameter being then automatically set by the solver to use a maximum amount of memory. We kept only the results for the best tuning (5GB, the worst results being obtained with 50GB). Note that because of large domain sizes, and the $O(d^i)$ space complexity of mini-buckets, a fine tuning of this parameter should have limited influence on the results.

The WCSP instances were transformed into the UAI ''MARKOV" format through the application of an exponential transformation of costs into multiplicative potentials. Costs above the upper bound were translated to zero potentials to preserve pruning. The exponential basis was chosen so that the largest multiplicative potentials are equal to 1.

MPLP MAP-MRF solver. We downloaded the sources for the recent version 2 of the `mplp` (Message Passing Linear Programming) implementation [80, 81] available at `http://cs.nyu.edu/~dsontag/`.

This solver uses a Message Passing based bound and duality theory to identify optimal solutions of a MAP-MRF problem through successive tightening of subsets of variables. The message passing used in `mplp` defines reparametrizations of the underlying MRF. These reparametrizations are similar to the reformulations done by local consistencies in

CFN [79, 18]. The solver is unique in all the solvers considered in that it does not use branching but only increasingly strong inference by applying reparametrizations to set of variables that initially contain only pairwise potentials, reasoning on stars [35], and are incrementally enlarged to include several potentials and strengthen the corresponding bound [81, 80].

All costs were divided by 1,000 and the optimality gap threshold kept to the default of $2 \cdot 10^{-4}$. The solver does not have any parameter.

ILP and QP optimization. We used `cplex` version 12.2 with parameters EPAGAP, EPGAP, and EPINT set to zero to avoid premature stop. No other tuning was done.

Boolean quadratic optimization. We used the `biqmac` [78] solver (`http://biqmac.uni-klu.ac.at/biqmaclib.html`) from sources provided by Angelika Wiegele. `biqmac` is a branch-and-bound solver relying on a strong Semi-Definite Programming (SDP) bound for Boolean quadratic optimization. The SDP framework is known to provide strong bounds for a variety of combinatorial optimization problems among which MaxCut and Max2SAT, with guaranteed approximation ratios [36]. Two solver settings (with branching rule set to 2 or 3, as advised by the author) were tried with no significant difference in the performances.

Weighted partial MaxSAT optimization. The same problems have been translated to WPMS using the two previously described encodings. There are two categories of complete WPMS solvers that we consider here: branch-and-bound (B&B) solvers and sequence-of-SAT solvers.

- Sequence-of-SAT solvers reformulate the WPMS problem as a series of SAT instances that allow us to successively increase the lower bound or decrease the upper bound for the optimal solution of the WPMS instance. A particular technique used by several sequence-of-SAT solvers, such as `WPM1` [4] and `maxhs` [20], is identifying *unsatisfiable cores* of the WPMS instance. An unsatisfiable core is a subset of the soft clauses of the instances which, taken together with the hard clauses of the instances, cannot all be satisfied by any assignment. The sequence of SAT instances then builds a collection of cores. The last SAT instance produces an assignment that violates at least one clause from each core but satisfies all other clauses. This assignment can be shown to be optimal.

- B&B solvers explore a backtracking search tree. At each node of the tree, they compute a lower bound on the cost of the best solution that can be found in the subtree rooted at that node. If that lower bound is higher than the cost of the best solution found so far, the solver backtracks. The solver `minimaxsat` [43] employs a method that is typically used in B&B solvers. In its case, the lower bound computation consists in performing unit propagation over the entire formula, including soft clauses. Unit propagation is able to detect some but not all unsatisfiable cores of the reduced formula at the current node. These cores are collected and used to transform the formula into an equivalent formula with a higher lower bound.

As B&B solvers, we have used `akmaxsat` [52] as it was among the best B&B performers in the latest MaxSAT evaluation and `minimaxsat` [43], which was shown to be one of the best solvers over all the instances of all MaxSAT evaluations in [21]. Among sequence-of-SAT solvers, we have used `bin-c-d`, `wpm1` and `wpm2`, which are among the best performers in recent evaluations, as well as `maxhs`, which was shown to be the best solver for the entire ensemble of instances of MaxSAT evaluations [21].

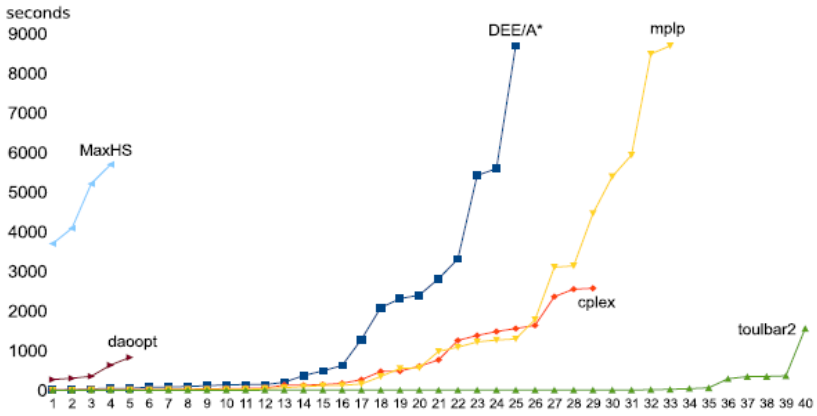


Figure 4: A figure showing the number of problems that can be solved by each approach (X-axis) as a function of cumulative time (Y-axis), assuming that each solver tackles problems in increasing order of cpu-time needed to solve it.

We can observe that there exists a bijection between cores of the direct encoding of an instance and cores of the tuple encoding. However, there exist cores in the tuple encoding that can be detected just by unit propagation, but require a longer refutation in the direct encoding. On the other hand, the tuple encoding is larger and hence unit propagation is slower. Since both B&B and sequence-of-SAT solvers essentially rely on collecting cores of the formula, both types of solver can benefit from the tuple encoding by detecting more cores with less search. However, the overhead of performing unit propagation on a larger formula may not pay off in runtime.

CP solvers. We used `gecode` version 4.2.0 (`http://www.gecode.org/`) and `mistral` version 1.3.40 (using its Python interface `numberjack` at `http://numberjack.ucc.ie/` and `http://github.com/eomahony/Numberjack/tree/fzn`). `mistral` uses a Weighted Degree heuristics [8] and a restart strategy with geometric factor 1.3 and base 256. No tuning was done for `gecode`.

All the Python and C translating scripts used are available together with the CPD instances at `http://genotoul.toulouse.inra.fr/~tschiex/CPD-AIJ`.

6.2. Results

Several solvers were unable to solve any of the instances in the allocated 9,000 seconds per problem. Despite the compact associated models, neither `cplex` for the quadratic programming model, nor `biqmac` for the quadratic Boolean optimization model could solve any single instance in less than 9,000 seconds. Similarly, most of the WPMS solvers failed to solve any instance, in either of the two encoding tested. The only exception to this is the `maxhs` solver when applied to the tuple encoding. Finally, neither `gecode` nor `mistral` could solve any single instance. In Table 1, we therefore only report the results obtained by the WPMS solver `maxhs`, the CPD solver `osprey`, the ILP solver `cplex`, the MAP-MRF solvers `daoopt` and `mplp`, and the CFN solver `toulbar2` in its vanilla version (using the default variable ordering heuristics and no DEE).

The detailed results are given in Table 1. The table shows that the cpu-times are very well correlated across different models and solvers, and show a clear ordering in terms of difficulty of these problems for all solvers, from WPMS/`maxhs`, MAP-MRF/`daoopt`, DEE/A*/`osprey`, ILP/`cplex`, MAP-MRF/`mplp`, and CFN/`toulbar2`.

Table 1: For each instance: protein (PDB id.), number of mutable residues, maximum domain size (maximum number of rotamers), and CPU-time for solving using `maxhs`, `daoopt`, DEE/A*, `cplex`, `mplp`, and `toulbar2`. A '-' indicates that the corresponding solver did not prove optimality within the 9,000-second time-out. A '!' indicates the solver stops with a SEGV signal.

PDB id.	n	d	`maxhs`	`daoopt`	DEE/A*	`cplex`	`mplp`	`toulbar2`
2TRX	11	44	4,086	268.6	31.5	2.6	2.8	**0.1**
1PGB	11	45	5,209	300.4	135.3	3.6	0.5	**0.1**
1HZ5	12	45	5,695	350.2	75.0	7.6	16.7	**0.1**
1UBI	13	45	-	826.9	2,812.6	139.2	37.3	**0.2**
1PGB	11	148	-	-	8,695.2	-	1,291	**4.3**
1HZ5	12	148	-	-	2,398.3	1,555	1,217	**2.4**
1UBI	13	148	-	-	-	-	-	**1,557**
2PCY	18	44	-	-	1,281.1	26.9	14.5	**0.2**
2DHC	14	148	-	-	-	-	5,388	**14.1**
1CM1	17	148	-	-	138.4	473.1	87.5	**3.3**
1MJC	28	182	3,698	631.7	4.6	4.1	0.8	**0.1**
1CSP	30	182	-	-	200.0	1,380	1,264	**0.8**
1BK2	24	182	-	-	93.2	125.0	114.9	**0.6**
1SHG	28	182	-	-	138.0	39.4	!	**0.2**
1CSK	30	49	-	-	41.7	12.5	9.6	**0.1**
1SHF	30	56	-	-	44.3	8.6	3.1	**0.1**
1FYN	23	186	-	-	622.0	2,548	3,136	**2.8**
1PIN	28	194	-	-	-	-	-	**3.7**
1NXB	34	56	-	-	11.1	17.0	4.5	**0.2**
1TEN	39	66	-	-	113.0	45.4	17.1	**0.2**
1POH	46	182	-	-	77.9	29.0	13.1	**0.3**
2DRI	37	186	-	-	-	-	4,458	**42.8**
1FNA	38	48	-	-	3,310	124.9	121.2	**0.5**
1UBI	40	182	-	-	-	2,572	979.4	**2.4**
1C9O	43	182	-	-	2,310	1,635	155.7	**1.8**
1CTF	39	56	-	-	-	263.2	549.2	**0.7**
2PCY	46	56	-	-	2,080	54.0	20.3	**0.4**
1DKT	46	190	-	-	5,420	1,254	3,103	**2.5**
2TRX	61	186	-	-	487.0	765.0	344.1	**0.9**
1CM1	42	186	-	-	-	-	-	**17.4**
1BRS	44	194	-	-	-	-	-	**346.5**
1CDL	40	186	-	-	-	-	-	**341.8**
1LZ1	59	57	-	-	-	601.6	1,084	**1.5**
1GVP	52	182	-	-	-	-	-	**361.8**
1RIS	56	182	-	-	-	-	8,483	**288.4**
2RN2	69	66	-	-	-	480.8	565.2	**1.2**
1CSE	97	183	-	-	367.0	172.9	60.9	**0.7**
1HNG	85	182	-	-	5,590	2,360	5,934	**2.8**
3CHY	74	66	-	-	-	-	8,691	**59.6**
1L63	83	182	-	-	-	1,480	1,779	**2.9**
			4	5	25	29	33	**40**

The variant of the ILP model originally proposed by [51], where the p_{irjs} variables are constrained to be 0/1 variables was also tested. It was overall less efficient than the

relaxed model we used. The ratio in terms of speedup was never very important (between 0.2 and 3.4 with a mean of 1.4 over all the solved instances) showing the robustness of `cplex`. It is often claimed, following [86], that LP technology is not able to deal with large instances of MRF. This experiment, on realistically designed instances of CPD, using state-of-the-art energy functions, including sterical clashes, shows that the recent 12.2 version of `cplex` gives reasonably good results on these problems.

6.3. Non-vanilla `toulbar2`

The results obtained on the same 40 CPD instances using the vanilla `toulbar2`, enhanced with our new variable ordering heuristics and increasingly stronger DEE processing are given in Table 2. None of the 7 open unreported instances could be solved by these new variants.

The new variable ordering heuristics consistently offer improved results. The effect of additional DEE processing is mostly visible on the difficult instances, the most visible and persistent improvements being obtained when using DEE in pre-processing, and for some instances (*e.g.,* 1BRS, 1RIS) also maintaining DEE^1 during search. They offer speedups up to 6 (on 1GVP). Further tests on a variety of CFN benchmarks (`http://costfunction.org`) are reported in [34]. They show that DEE^1 allows to solve more problems and DEE^1 is now a default option of `toulbar2`.

Table 2: For each instance: CPU-time for solving using `toulbar2` and different combinations of options for DVO and DEE. A '-' indicates that the corresponding solver did not prove optimality with the 9,000-second time-out. The `tb2` column gives the results obtained using the vanilla `toulbar2` for reference. The DVO corresponds to the activation of the new variable ordering heuristics described in Section 4.1. This option is kept activated in all the remaining columns. These columns correspond respectively to additionally maintaining DEE^1 during search, pre-processing using DEE, doing both, and maintaining DEE during search. The last line reports the number of times a method was faster than the others.

PDB	n	d	`tb2`	DVO	DEE^1	DEE$_{pre}$	DEE$_{pre}$ +DEE^1	DEE
2TRX	11	44	**0.1**	**0.1**	**0.1**	**0.1**	**0.1**	**0.1**
1PGB	11	45	**0.1**	**0.1**	**0.1**	**0.1**	**0.1**	**0.1**
1HZ5	12	45	**0.1**	**0.1**	**0.1**	**0.1**	**0.1**	**0.1**
1UBI	13	45	**0.2**	**0.2**	**0.2**	**0.2**	0.3	0.5
1PGB	11	148	4.3	3.8	3.4	**3.1**	8.6	15.1
1HZ5	12	148	2.4	2.4	2.3	**2.2**	3.1	3.5
1UBI	13	148	1,557	**1,068**	1,736	1,133	1,162	-
2PCY	18	44	**0.2**	**0.2**	**0.2**	**0.2**	**0.2**	**0.2**
2DHC	14	148	14.1	8.0	**7.0**	**7.0**	14.5	52.0
1CM1	17	148	3.3	**3.1**	3.2	**3.1**	**3.1**	**3.1**
1MJC	28	182	**0.1**	**0.1**	**0.1**	**0.1**	**0.1**	**0.1**
1CSP	30	182	0.8	0.6	**0.5**	0.7	0.7	0.8
1BK2	24	182	0.6	0.6	0.6	**0.5**	0.7	**0.5**
1SHG	28	182	**0.2**	**0.2**	**0.2**	**0.2**	**0.2**	**0.2**
1CSK	30	49	**0.1**	**0.1**	**0.1**	**0.1**	**0.1**	**0.1**
1SHF	30	56	**0.1**	**0.1**	**0.1**	**0.1**	**0.1**	**0.1**
1FYN	23	186	2.8	2.9	**2.6**	3.0	3.2	3.8
1PIN	28	194	3.7	**3.0**	**3.0**	4.8	6.2	12.0
1NXB	34	56	**0.2**	**0.2**	**0.2**	**0.2**	**0.2**	**0.2**
1TEN	39	66	**0.2**	**0.2**	**0.2**	**0.2**	**0.2**	**0.2**
1POH	46	182	**0.3**	**0.3**	**0.3**	0.4	0.4	0.4
2DRI	37	186	42.8	16.4	37.7	**9.6**	15.5	51.2
1FNA	38	48	0.5	0.4	**0.3**	0.4	0.4	0.5
1UBI	40	182	2.4	1.0	**0.7**	0.9	0.9	1.3
1C9O	43	182	1.8	**1.5**	1.7	2.3	2.4	3.6
1CTF	39	56	0.7	0.9	**0.6**	**0.6**	0.7	0.8
2PCY	46	56	**0.4**	**0.4**	**0.4**	**0.4**	**0.4**	**0.4**
1DKT	46	190	2.5	2.8	**2.4**	2.6	2.7	3.9
2TRX	61	186	**0.9**	**0.9**	**0.9**	1.8	1.7	1.9
1CM1	42	186	17.4	11.8	13.2	**8.6**	11.6	20.0
1BRS	44	194	346.5	241.4	135.4	70.4	**60.1**	129.0
1CDL	40	186	341.8	198.1	159.0	**79.6**	128.8	286.4
1LZ1	59	57	1.5	1.1	1.0	**0.9**	1.0	1.1
1GVP	52	182	361.8	248.5	408.2	**38.3**	66.8	163.5
1RIS	56	182	288.4	147.4	77.9	37.8	**28.8**	122.8
2RN2	69	66	1.2	**1.1**	**1.1**	1.2	**1.1**	1.2
1CSE	97	183	0.7	0.8	**0.6**	**0.6**	**0.6**	**0.6**
1HNG	85	182	2.8	2.4	**2.3**	3.1	2.8	3.6
3CHY	74	66	59.6	27.9	10.7	**10.6**	14.9	20.3
1L63	83	182	2.9	2.8	**2.3**	2.4	2.5	2.7
			14	19	26	25	16	14

6.4. Analysis of results

It is unusual to apply such a wide range of NP-complete solving methods on a common set of benchmarks. Most comparisons are usually performed on a closely related family of solvers, sharing a common modeling language (SAT, CSP, MRF...).

Solvers are complex systems involving various mechanisms. The effect of their interactions during solving is hard to predict. Therefore, explaining the differences in efficiency observed between the different approaches is not straightforward. However, given the simplicity of our encodings, the fact that these instances are challenging for some approaches while at the same time being simpler to solve for other approaches should provide a source of inspiration for solver designers.

Quadratic programming and Quadratic optimization. One of the first surprising results is the difficulty of these instances for quadratic programming with `cplex`. The quadratic model is very dense with nd Boolean variables only. `cplex` is a totally closed-source black box but the behavior of the solver provides some information on its weak spot here. On the simplest problems, QP/`cplex` consumes memory very quickly and grows a very large node file. On the simple 2TRX problem ($n = 11$, first line of Table 1), QP/`cplex` solver explored $51,003,970$ nodes and was interrupted by the time-out with an optimality gap of 774%. This indicates a poor lower bound that leads to memory intensive best first search. On bigger problems, the number of nodes is never large because each node takes quite a time to explore. On the 1UBI problem ($n = 13, d = 148$), it explored only 5,233 nodes with an unbounded gap. It is therefore reasonable to assume that the lower bound used by `cplex` is too slow to compute on these problems and does not provide the additional strength that would compensate for the computing cost.

The model we devised for BQO using `biqmac` is compact, with the same $n.d$ 0/1 variables. `biqmac` uses a semidefinite programming lower bound that is known to provide among the strongest polynomial time lower bounds for a variety of optimization problems [76]. Despite this, even the smallest CPD instances could not be solved. We tried to extend the 9,000-second deadline for the simplest instance. After several hours of computing, `biqmac` stopped and reported that only a few nodes had been explored. The SDP technology used in `biqmac` may provide excellent bounds, but the time needed to compute them is currently too large to offer a viable alternative for CPD. The `biqmac` library at `http://biqmac.uni-klu.ac.at/biqmaclib.html` contains a variety of QP and (closely related) MaxCut problems that can be used for benchmarking. We tested `toulbar2` on the 10 `beasley` instances of size $n = 100$. They are solved in less than 1 second each by `toulbar2`, whereas `biqmac` took around 1 minute each, as reported in [78].

Integer linear programming. Considering 01LP, it is known that the continuous LP relaxation of the 0/1 linear programming model we used in Section 5.2 is the dual of the LP problem encoded by Optimal Soft Arc Consistency (OSAC) [17, 14] when the upper bound k used in CFN is infinite. OSAC is known to be stronger than any other soft arc consistency level, including EDAC and Virtual Arc Consistency (VAC) [16]. However, as soon as the upper bound k used for pruning in CFN decreases to a finite value, soft local consistencies may prune values and EDAC becomes incomparable with the dual of these relaxed LPs. To better evaluate the pruning power of `cplex`, we compared the number of nodes it explored with those explored by `toulbar2` in its vanilla mode or with the new heuristics and DEE pre-processing. Table 3 shows that among the 28 instances solved, 18 are solved by `cplex` before search starts, 7 are solved by the non-vanilla version of `toulbar2` w/o backtracks. For the remaining less trivial problems, the number of nodes

explored by `cplex` and `toulbar2` are often similar with no clear winner. Overall, these results show comparable pruning power. It also shows that the problems solved by `cplex` are relatively simple problems but the computation of the lower bound is quite expensive in `cplex`. It typically develops from 1 to 50 nodes per minute while `toulbar2` develops from 1 to 40 thousand nodes per minute. Note that the problems that are not solved by `cplex` are much harder, requiring more than 120,000 nodes to explore for the hardest solved problem (not shown in the Table).

Table 3: For each instance solved by both `cplex` and `toulbar2`, we report the number of nodes explored by each solver (with the number of backtracks in parentheses when available) and the number of nodes per minute developed. `toulbar2` is the vanilla version, `toulbar2`$^+$ uses the new variable ordering heuristics and DEE as pre-processing.

PDB id.	n	d	`cplex`		`toulbar2`		`toulbar2`$^+$	
			nodes	nd/min	nodes (bt)	nd/min	nodes (bt)	nd/min
2TRX	11	44	0	-	8 (0)	6857	10 (1)	7500
1PGB	11	45	0	-	17 (1)	11333	16 (1)	10667
1HZ5	12	45	0	-	25 (5)	15000	29 (7)	17400
1UBI	13	45	51	22.0	143 (61)	39000	82 (31)	24600
1HZ5	12	148	0	-	89 (34)	2225	54 (16)	1453
2PCY	18	44	0	-	53 (6)	13826	40 (9)	10909
1CM1	17	148	0	-	14 (0)	258	0 (0)	0
1MJC	28	182	0	-	22 (0)	14667	2 (0)	1714
1CSP	30	182	547	23.8	540 (245)	42078	42 (16)	3877
1BK2	24	182	3	1.4	28 (3)	2800	19 (3)	2375
1SHG	28	182	214	326	268 (101)	69913	51 (16)	19125
1CSK	30	49	0	-	38 (5)	20727	14 (3)	7636
1SHF	30	56	0	-	35 (4)	17500	12 (0)	6000
1FYN	23	186	0	-	84 (20)	1819	43 (8)	863
1NXB	34	56	0	-	30 (0)	9474	14 (0)	4667
1TEN	39	66	0	-	75 (6)	20455	22 (5)	6600
1POH	46	182	0	-	111 (5)	21484	15 (0)	2195
1FNA	38	48	0	-	189 (47)	25200	84 (28)	12600
1UBI	40	182	287	6.7	1,669 (766)	41900	539 (228)	36337
1C9O	43	182	49	1.8	222 (57)	7525	82 (16)	2112
1CTF	39	56	94	21.4	294 (95)	24845	110 (33)	10820
2PCY	46	56	0	-	62 (5)	10629	20 (0)	3158
1DKT	46	190	0	-	210 (36)	5122	134 (24)	3045
2TRX	61	186	6	0.5	111 (14)	7239	85 (16)	2818
1LZ1	59	57	735	73.5	807 (308)	31855	178 (53)	11609
2RN2	69	66	0	-	105 (17)	5385	110 (17)	5500
1CSE	97	183	0	-	94 (0)	8418	9 (0)	915
1HNG	85	182	48	1.2	411 (110)	8745	96 (21)	1870
1L63	83	182	0	-	196 (17)	4055	58 (3)	1468

Markov Random Field MAP. The relaxed LP is also equivalent, in the pairwise case, to the LP relaxation of MRFs in the so-called *local polytope* [81]. In its original version [35], `mplp` is only guaranteed to produce this LP bound if domains are Boolean. It is therefore weaker than OSAC for CFN and comparable to Virtual AC [14]. With the recent additions

described in [81, 80], `mplp` has the ability to incrementally tighten its bound by performing local inference on several potentials (or cost functions) organized in cyclic structures. The strength of this lower bound is such that `mplp` version 2 is often able to prove optimality based just on this bound and the cost of the assignment that optimizes unary reparameterized potentials. Still, weaker but faster CFN lower bounds combined with search apparently offer a better solution on these realistic CPD instances.

In the MRF community, the inefficiency of pure LP on large MRF instances is well-known [86]. These experiments show that, combined with branching, the incrementality of the LP bound allows 01LP to get very decent results on these problems. However, the quadratic size of the 01LP model probably explains the better efficiency of `mplp`.

MaxSAT. The most surprising result is probably the difficulty of these problems for MaxSAT solvers, either branch-and-bound based or core-based. To analyze branch and bound based algorithm behavior, we instrumented the two solvers MiniMaxSat and `akmaxsat` to report the best upper bound found and the number of nodes explored. Additionally, `akmaxsat` reports the lower bound computed at the root node of the search tree. In the direct encoding, MiniMaxSat is fast and may explore up to 36 thousand nodes per second (two orders of magnitude faster than `toulbar2`). In 15 problems, it was able to identify sub-optimal solutions ending up with a non-trivial upper bound (within 3.2% to 0.26% of the optimum) but never started the final optimality proof, showing a weak lower bound. Indeed, the lower bound computed by `akmaxsat` at the root of the search tree is never higher than 27% of the optimum. In contrast, the lower bound computed by `toulbar2` at the root was often 99% of the optimum and never less than 97%. We conclude that the direct encoding does not allow for strong propagation and lower bouds.

The tuple encoding was chosen to put WPMS solvers in a situation where UP applied to a hardened version of the formula would be able to detect more unsatisfiable cores. Since this operation is at the heart of the lower bounding procedures of such solvers, it should allow the derivation of a stronger lower bound. Additionally, unit propagation on the hardened version of the tuple encoding is equivalent to enforcing arc consistency on the hardened CFN. In CFN, VAC precisely identifies subproblems whose hardened version is arc inconsistent (and therefore define inconsistent cores) to increase the lower bound. VAC is known to be capable of producing stronger lower bounds than the default local consistency EDAC [55] used in `toulbar2`. Hence, the tuple encoding provides enough information to give better lower bounds than what EDAC computes. The empirical results verify that the lower bound computed at the root is much stronger with the tuple encoding than with the direct encoding. For several instances `akmaxsat` computes a lower bound that is 92% of the optimum. However, this is still far from the lower bound computed by `toulbar2`. But the more important problem with this encoding is that with a quadratic number of extra variables, both minimaxsat and `akmaxsat` were extremely slow, exploring at most 2 nodes before the 9,000 second time-out and in several instances timed out before even finishing the lower bound computation at the root node. They never produced a single incumbent assignment.

On the other hand, the `maxhs` core-based solver is able to exploit the stronger tuple encoding, being able to solve 4 problems to optimality. Analyzing the behaviour of `maxhs` on these instances reveals that the solver spends almost all of its time trying to reduce the size of the cores it finds, using a greedy minimization algorithm. This is because these protein design instances contain some very large cores (tens of thousands of clauses) which can not be significantly reduced in size. Such cores arise, for example, from the binary cost functions in the CFN model. In this case, the core expresses the condition that the

cost of at least one tuple in the cost function will be incurred, and the core therefore contains as many clauses as there are tuples in the originating cost function. We observed that as a result, `maxhs` is usually unable to complete its initial disjoint core phase within the timeout. Given this observation, we also experimented with running `maxhs` with the core minimization option turned off, on the set of 12 instances which update those in [2], using the tuple encoding. With core minimization turned off, `maxhs` was able to complete the disjoint core phase on all 12 instances. This allowed us to compare the lower bound produced by the disjoint core phase of `maxhs` with the lower bound produced at the root node by `toulbar2`. Over the 12 instances, the lower bound produced by `maxhs` was between 84% and 98% of the lower bound produced by `toulbar2`, and it was calculated within only 35 seconds except for two cases. Note that this bound calculated by `maxhs` differs from that of `toulbar2` in that `maxhs` uses a complete SAT solver to find the cores, and the cores are strictly disjoint. Based on these observations, we believe the potential to improve the performance of the `maxhs` approach on these instances is very promising.

For other core-based solvers, either because of the quadratic number of variables or because of a different exploitation of non-AC/UP cores, these CPD instances remain very hard. Whether it is a fundamental, technical, or implementation difference, identifying the cause of this difference should allow to improve the existing WPMS technology.

Constraint programming. The generic translation of WCSPs into crisp CSPs suffers from the large magnitude of costs, resulting in large domains for the extra cost variables with very slow arc consistency propagation of `table` constraints, `mistral` develops approx. 215 nodes per minute. It also requires huge memory space for expressing the `table` constraints. Only 4 instances among the 42 could fit into 128 GB during `minizinc` to `flatzinc` translation. By dividing all costs by 100 (*i.e.,* $M = 1$), `mistral` was able to solve 4 instances (21 unsolved due to memory errors), including 2TRX ($n = 11$) in 47.3 seconds and 28,057 nodes, and `gecode` solved only one: 2TRX in 2,234 seconds and 213,423 nodes (20 unsolved due to memory errors). The difference in performances between the two solvers might come from the different search strategies, `mistral` used restarts, but not `gecode`.

DEE/A.* The DEE/A* combination uses strong polynomial time dominance analysis using several variants of dead-end elimination. This pre-processing is followed by best-first search relying on an obsolete lower bound instead of the stronger lower bounds offered by soft local consistencies such as EDAC [55], or the LP relaxation bound. To confirm this, we computed the number of nodes explored by `osprey` during A* search. Except for simple problems where DEE alone could solve the problem, `osprey` explored trees larger than those explored by ILP or CFN by several orders of magnitude. On problem 1DKT, it explored more than 10^7 nodes while ILP/`cplex` solved the problem without search and `toulbar2` explored 134 nodes. This confirms the weakness of current bounds in exact CPD algorithms. Otherwise, `osprey` is quite fast and can develop more than 110,000 nodes per minute. Despite the exploration of huge trees, no DEE/A* execution led to memory exhaustion before time-out. With an extended time-out of 100 ours [83], only 2 instances ultimately led to memory exhaustion. Iterative alternatives to A^* such as IDA* [50] would therefore probably have little influence on the results of DEE/A*.

7. Conclusions

The simplest formal optimization problem underlying CPD looks for a Global Minimum Energy Conformation (GMEC) over a rigid backbone and altered side-chains (iden-

tity and conformation). In computational biology, exact methods for solving the CPD problem combine dominance analysis (DEE) and an A^* search.

The CPD problem can also be directly formulated as a Cost Function Network, with a very dense graph and relatively large domains. We have shown how DEE can be integrated with local consistency with a reasonable time complexity.

The CPD can also be easily reduced to optimization in MRF, 01LP, 01QP, weighted partial MaxSAT, and Boolean quadratic optimization, offering an ideal benchmark for a large cross-technology comparison.

On a variety of real instances, we have shown that state-of-the-art optimization algorithms on graphical models exploiting bounds based on the reformulation (or reparametrization) of the graphical model, but also 01LP algorithms, give important speedups compared to usual CPD algorithms combining dead-end elimination with $A*$. Among all the tested solvers, `toulbar2` was the most efficient solver and its efficiency was further improved by the use of DEE during search.

We also showed that these CPD problems define challenging benchmarks for a variety of solvers, including weighted partial MaxSAT solvers, either branch-and-bound or core-based, and quadratic programming or quadratic optimization solvers, including semidefinite programming based solvers.

In practice, it must be stressed that just finding the GMEC is not a final answer to real CPD problems. CDP energies functions represent an approximation of the real physics of proteins and optimizing a target score based on them (such as stability, affinity,...) is not a guarantee of finding a successful design. Indeed, some designs may be so stable that they are unable to accomplish the intended biological function. The usual approach is therefore to design a large library of proteins whose sequences are extracted from all solutions within a small threshold of energy of the GMEC. This problem is also efficiently solved by `toulbar2` [83].

Although it is easy to formulate as a discrete optimization problem, another important limitation of the rigid backbone/rotamer CPD problem lies in the restrictions generated by these two assumptions. In practice, rotamers offer a continuous range of rotations along dihedral angles and backbones also have degrees of flexibility. Several approaches have been proposed and introduced in `osprey` in the last few years that relax either or both of these two assumptions while still offering a guarantee of optimality [40, 29, 31]. When flexibility counts, `osprey` is therefore a reference tool. All these approaches ultimately require to solve the very same type of optimization problems involving a sum of precomputed pairwise *lower bounds* on energy terms. In this context, it becomes crucial to be able to enumerate all the solutions within a threshold of the optimum. These approaches should therefore ultimately also benefit from algorithmic improvements in GMEC optimization, as far as exhaustively enumerating all the solutions within a threshold of the optimum is feasible.

Acknowledgements

This work has been partly funded by the "Agence nationale de la Recherche" (ANR-10-BLA-0214 and ANR-12-MONU-0015-03), the INRA and the Region Midi-Pyrnes. We would like to thank Damien Leroux for his help in the generation of encodings using Python. We thank the Computing Center of Region Midi-Pyrénées (CALMIP, Toulouse, France) and the GenoToul Bioinformatics Platform of INRA-Toulouse for providing computing resources and support.

References

[1] Achterberg, T., Koch, T., Martin, A., 2005. Branching rules revisited. Operations Research Letters 33, 42–54.

[2] Allouche, D., Traoré, S., André, I., de Givry, S., Katsirelos, G., Barbe, S., Schiex, T., 2012. Computational protein design as a cost function network optimization problem, in: Principles and Practice of Constraint Programming, Springer. pp. 840–849.

[3] Anfinsen, C., 1973. Principles that govern the folding of protein chains. Science 181, 223–253.

[4] Ansótegui, C., Bonet, M.L., Levy, J., 2009. Solving (weighted) partial maxsat through satisfiability testing, in: Theory and Applications of Satisfiability Testing-SAT 2009, Springer. pp. 427–440.

[5] Argelich, J., Cabiscol, A., Lynce, I., Manyà, F., 2008. Encoding Max-CSP into partial Max-SAT, in: Multiple Valued Logic, 2008. ISMVL 2008. 38th International Symposium on, IEEE. pp. 106–111.

[6] Bacchus, F., 2007. GAC via unit propagation, in: Principles and Practice of Constraint Programming–CP 2007, Springer. pp. 133–147.

[7] Boas, F.E., Harbury, P.B., 2007. Potential energy functions for protein design. Current opinion in structural biology 17, 199–204. URL: `http://www.ncbi.nlm.nih.gov/pubmed/17387014`, doi:`10.1016/j.sbi.2007.03.006`.

[8] Boussemart, F., Hemery, F., Lecoutre, C., Sais, L., 2004. Boosting systematic search by weighting constraints, in: ECAI, p. 146.

[9] Bowie, J.U., Luthy, R., Eisenberg, D., 1991. A method to identify protein sequences that fold into a known three-dimensional structure. Science 253, 164–170.

[10] Campeotto, F., Dal Pal, A., Dovier, A., Fioretto, F., Pontelli, E., 1991. A constraint solver for flexible protein models. Science 253, 164–170.

[11] Carothers, J.M., Goler, J.A., Keasling, J.D., 2009. Chemical synthesis using synthetic biology. Current opinion in biotechnology 20, 498–503.

[12] Case, D., Darden, T., Cheatham III, T., Simmerling, C., Wang, J., Duke, R., Luo, R., Merz, K., Pearlman, D., Crowley, M., Walker, R., Zhang, W., Wang, B., Hayik, S., Roitberg, A., Seabra, G., Wong, K., Paesani, F., Wu, X., Brozell, S., Tsui, V., Gohlke, H., Yang, L., Tan, C., Mongan, J., Hornak, V., Cui, G., Beroza, P., Mathews, D., Schafmeister, C., Ross, W., Kollman, P., 2006. Amber 9. Technical Report. University of California. San Francisco.

[13] Champion, E., André, I., Moulis, C., Boutet, J., Descroix, K., Morel, S., Monsan, P., Mulard, L.A., Remaud-Siméon, M., 2009. Design of α-transglucosidases of controlled specificity for programmed chemoenzymatic synthesis of antigenic oligosaccharides. Journal of the American Chemical Society 131, 7379–7389.

[14] Cooper, M., de Givry, S., Sanchez, M., Schiex, T., Zytnicki, M., Werner, T., 2010. Soft arc consistency revisited. Artificial Intelligence 174, 449–478.

[15] Cooper, M.C., 2005. High-order consistency in Valued Constraint Satisfaction. Constraints 10, 283–305.

[16] Cooper, M.C., de Givry, S., Sánchez, M., Schiex, T., Zytnicki, M., 2008. Virtual arc consistency for weighted CSP., in: Proc. of AAAI'08, pp. 253–258.

[17] Cooper, M.C., de Givry, S., Schiex, T., 2007. Optimal soft arc consistency, in: Proc. of IJCAI'2007, Hyderabad, India. pp. 68–73.

[18] Cooper, M.C., Schiex, T., 2004. Arc consistency for soft constraints. Artificial Intelligence 154, 199–227.

[19] Dahiyat, B.I., Mayo, S.L., 1996. Protein design automation. Protein science : a publication of the Protein Society 5, 895–903. URL: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2143401\&tool=pmcentrez\&rendertype=abstract, doi:10.1002/pro.5560050511.

[20] Davies, J., Bacchus, F., 2011. Solving MAXSAT by solving a sequence of simpler SAT instances, in: Principles and Practice of Constraint Programming–CP 2011. Springer, pp. 225–239.

[21] Davies, J., Bacchus, F., 2013. Exploiting the power of MIP solvers in MaxSAT, in: Theory and Applications of Satisfiability Testing–SAT 2013, Springer. pp. 166–181.

[22] Dechter, R., Mateescu, R., 2007. AND/OR search spaces for graphical models. Artificial intelligence 171, 73–106.

[23] Dechter, R., Rish, I., 2003. Mini-buckets: A general scheme for bounded inference. Journal of the ACM (JACM) 50, 107–153.

[24] Desmet, J., De Maeyer, M., Hazes, B., Lasters, I., 1992. The dead-end elimination theorem and its use in protein side-chain positioning. Nature 356, 539–42. URL: http://www.ncbi.nlm.nih.gov/pubmed/21488406.

[25] Desmet, J., Spriet, J., Lasters, I., 2002. Fast and accurate side-chain topology and energy refinement (FASTER) as a new method for protein structure optimization. Proteins 48, 31–43. URL: http://www.ncbi.nlm.nih.gov/pubmed/12012335, doi:10.1002/prot.10131.

[26] Fersht, A., 1999. Structure and mechanism in protein science: a guide to enzyme catalysis and protein folding. WH. Freemean and Co., New York.

[27] Freuder, E.C., 1991. Eliminating interchangeable values in constraint satisfaction problems, in: Proc. of AAAI'91, Anaheim, CA. pp. 227–233.

[28] Fritz, B.R., Timmerman, L.E., Daringer, N.M., Leonard, J.N., Jewett, M.C., 2010. Biology by design: from top to bottom and back. BioMed Research International 2010.

[29] Gainza, P., Roberts, K.E., Donald, B.R., 2012a. Protein design using continuous rotamers. PLoS computational biology 8, e1002335.

[30] Gainza, P., Roberts, K.E., Georgiev, I., Lilien, R.H., Keedy, D.A., Chen, C.Y., Reza, F., Anderson, A.C., Richardson, D.C., Richardson, J.S., et al., 2012b. Osprey: Protein design with ensembles, flexibility, and provable algorithms. Methods Enzymol .

[31] Georgiev, I., Keedy, D., Richardson, J.S., Richardson, D.C., Donald, B.R., 2008a. Algorithm for backrub motions in protein design. Bioinformatics 24, i196–i204.

[32] Georgiev, I., Lilien, R.H., Donald, B.R., 2006. Improved Pruning algorithms and Divide-and-Conquer strategies for Dead-End Elimination, with application to protein design. Bioinformatics (Oxford, England) 22, e174–83. URL: http://www.ncbi.nlm.nih.gov/pubmed/16873469, doi:10.1093/bioinformatics/btl220.

[33] Georgiev, I., Lilien, R.H., Donald, B.R., 2008b. The minimized dead-end elimination criterion and its application to protein redesign in a hybrid scoring and search algorithm for computing partition functions over molecular ensembles. Journal of computational chemistry 29, 1527–42. URL: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3263346\&tool=pmcentrez\&rendertype=abstract, doi:10.1002/jcc.20909.

[34] de Givry, S., Prestwich, S., O'Sullivan, B., 2013. Dead-end elimination for weighted CSP, in: Springer (Ed.), Principles and Practice of Constraint Programming–CP 2013.

[35] Globerson, A., Jaakkola, T.S., 2007. Fixing max-product: Convergent message passing algorithms for map lp-relaxations, in: Advances in neural information processing systems, pp. 553–560.

[36] Goemans, M.X., Williamson, D.P., 1995. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. Journal of the ACM (JACM) 42, 1115–1145.

[37] Goldstein, R.F., 1994. Efficient rotamer elimination applied to protein side-chains and related spin glasses. Biophysical journal 66, 1335–40. URL: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=1275854\&tool=pmcentrez\&rendertype=abstract, doi:10.1016/S0006-3495(94)80923-3.

[38] Gront, D., Kulp, D.W., Vernon, R.M., Strauss, C.E., Baker, D., 2011. Generalized fragment picking in rosetta: design, protocols and applications. PloS one 6, e23294.

[39] Grunwald, I., Rischka, K., Kast, S.M., Scheibel, T., Bargel, H., 2009. Mimicking biopolymers on a molecular scale: nano(bio)technology based on engineered proteins. Philosophical transactions. Series A, Mathematical, physical, and engineering sciences 367, 1727–47. URL: http://www.ncbi.nlm.nih.gov/pubmed/19376768, doi:10.1098/rsta.2009.0012.

[40] Hallen, M.A., Keedy, D.A., Donald, B.R., 2013. Dead-end elimination with perturbations (deeper): A provable protein design algorithm with continuous sidechain and backbone flexibility. Proteins: Structure, Function, and Bioinformatics 81, 18–39.

[41] Harvey, W.D., Ginsberg, M.L., 1995. Limited discrepancy search, in: Proc. of the 14^{th} IJCAI, Montréal, Canada.

[42] Hawkins, G., Cramer, C., Truhlar, D., 1996. Parametrized models of aqueous free energies of solvation based on pairwise descreening of solute atomic charges from a dielectric medium. The Journal of Physical Chemistry 100, 19824–19839.

[43] Heras, F., Larrosa, J., Oliveras, A., 2008. Minimaxsat: An efficient weighted Max-SAT solver. J. Artif. Intell. Res.(JAIR) 31, 1–32.

[44] Janin, J., Wodak, S., Levitt, M., Maigret, B., 1978. Conformation of amino acid side-chains in proteins. Journal of molecular biology 125, 357–386.

[45] Khalil, A.S., Collins, J.J., 2010. Synthetic biology: applications come of age. Nature Reviews Genetics 11, 367–379.

[46] Khare, S.D., Kipnis, Y., Greisen, P., Takeuchi, R., Ashani, Y., Goldsmith, M., Song, Y., Gallaher, J.L., Silman, I., Leader, H., Sussman, J.L., Stoddard, B.L., Tawfik, D.S., Baker, D., 2012. Computational redesign of a mononuclear zinc metalloenzyme for organophosphate hydrolysis. Nature chemical biology 8, 294–300. URL: http://www.ncbi.nlm.nih.gov/pubmed/22306579, doi:10.1038/nchembio.777.

[47] Khoury, G.A., Smadbeck, J., Kieslich, C.A., Floudas, C.A., 2014. Protein folding and *de novo* protein design for biotechnological applications. Trends in biotechnology 32, 99–109.

[48] Kingsford, C.L., Chazelle, B., Singh, M., 2005. Solving and analyzing side-chain positioning problems using linear and integer programming. Bioinformatics (Oxford, England) 21, 1028–36. URL: http://www.ncbi.nlm.nih.gov/pubmed/15546935, doi:10.1093/bioinformatics/bti144.

[49] Koller, D., Friedman, N., 2009. Probabilistic graphical models: principles and techniques. The MIT Press.

[50] Korf, R.E., 1985. Depth first iterative deepening : An optimal admissible tree search. Artificial Intelligence 27, 97–109.

[51] Koster, A., van Hoesel, S., Kolen, A., 1999. Solving Frequency Assignment Problems via Tree-Decomposition. Technical Report RM/99/011. Universiteit Maastricht. Maastricht, The Netherlands.

[52] Kuegel, A., 2010. Improved exact solver for the weighted Max-SAT problem, in: Workshop Pragmatics of SAT.

[53] Kuhlman, B., Baker, D., 2000. Native protein sequences are close to optimal for their structures. Proceedings of the National Academy of Sciences of the United States of America 97, 10383–8. URL: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=27033\&tool=pmcentrez\&rendertype=abstract.

[54] Larrosa, J., 2002. On arc and node consistency in weighted CSP, in: Proc. AAAI'02, Edmondton, (CA). pp. 48–53.

[55] Larrosa, J., de Givry, S., Heras, F., Zytnicki, M., 2005. Existential arc consistency: getting closer to full arc consistency in weighted CSPs, in: Proc. of the 19^{th} IJCAI, Edinburgh, Scotland. pp. 84–89.

[56] Larrosa, J., Meseguer, P., Schiex, T., Verfaillie, G., 1998. Reversible DAC and other improvements for solving max-CSP, in: Proc. of AAAI'98, Madison, WI.

[57] Larrosa, J., Schiex, T., 2004. Solving weighted CSP by maintaining arc consistency. Artif. Intell. 159, 1–26.

[58] Leach, A.R., Lemon, A.P., 1998. Exploring the conformational space of protein side chains using dead-end elimination and the A* algorithm. Proteins 33, 227–39. URL: http://www.ncbi.nlm.nih.gov/pubmed/9779790.

[59] Lecoutre, C., Roussel, O., Dehani, D.E., 2012. WCSP integration of soft neighborhood substitutability, in: Principles and Practice of Constraint Programming, Springer. pp. 406–421.

[60] Lecoutre, C., Saïs, L., Tabary, S., Vidal, V., 2009. Reasoning from last conflict(s) in constraint programming. Artificial Intelligence 173, 1592,1614.

[61] Lewis, J.C., Bastian, S., Bennett, C.S., Fu, Y., Mitsuda, Y., Chen, M.M., Greenberg, W.A., Wong, C.H., Arnold, F.H., 2009. Chemoenzymatic elaboration of monosaccharides using engineered cytochrome p450bm3 demethylases. Proceedings of the National Academy of Sciences 106, 16550–16555.

[62] Linderoth, J., Savelsbergh, M., 1999. A computational study of search strategies for mixed integer programming. INFORMS Journal on Computing 11, 173–187.

[63] Looger, L.L., Hellinga, H.W., 2001. Generalized dead-end elimination algorithms make large-scale protein side-chain structure prediction tractable: implications for protein design and structural genomics. Journal of molecular biology 307, 429–45. URL: http://www.ncbi.nlm.nih.gov/pubmed/11243829, doi:10.1006/jmbi.2000.4424.

[64] Lovell, S.C., Word, J.M., Richardson, J.S., Richardson, D.C., 2000. The penultimate rotamer library. Proteins 40, 389–408. URL: http://www.ncbi.nlm.nih.gov/pubmed/10861930.

[65] Marriott, K., Nethercote, N., Rafeh, R., Stuckey, P., de la Banda, M.G., Wallace, M., 2008. The design of the zinc modelling language. Constraints 13, 229–267.

[66] Martin, V.J., Pitera, D.J., Withers, S.T., Newman, J.D., Keasling, J.D., 2003. Engineering a mevalonate pathway in escherichia coli for production of terpenoids. Nature biotechnology 21, 796–802.

[67] Nestl, B.M., Nebel, B.A., Hauer, B., 2011. Recent progress in industrial biocatalysis. Current Opinion in Chemical Biology 15, 187–193. URL: http://www.ncbi.nlm.nih.gov/pubmed/21195018, doi:10.1016/j.cbpa.2010.11.019.

[68] Niedermeier, R., Rossmanith, P., 2000. New upper bounds for maximum satisfiability. J. Algorithms 36, 63–88.

[69] Otten, L., Ihler, A., Kask, K., Dechter, R., 2012. Winning the pascal 2011 map challenge with enhanced AND/OR branch-and-bound, in: DISCML'12 Workshop, at NIPS'12, Lake Tahoe, NV, USA.

[70] Pabo, C., 1983. Molecular technology. Designing proteins and peptides. Nature 301, 200. URL: http://www.ncbi.nlm.nih.gov/pubmed/6823300.

[71] Peisajovich, S.G., Tawfik, D.S., 2007. Protein engineers turned evolutionists. Nature methods 4, 991–4. URL: http://www.ncbi.nlm.nih.gov/pubmed/18049465, doi:10.1038/nmeth1207-991.

[72] Petit, T., Régin, J., Bessière, C., 2000. Meta constraints on violations for over constrained problems, in: Proceedings of IEEE ICTAI'2000, Vancouver, BC, Canada. pp. 358–365.

[73] Pierce, N., Spriet, J., Desmet, J., Mayo, S., 2000. Conformational splitting: A more powerful criterion for dead-end elimination. Journal of computational chemistry 21, 999–1009.

[74] Pierce, N.A., Winfree, E., 2002. Protein design is NP-hard. Protein engineering 15, 779–82. URL: http://www.ncbi.nlm.nih.gov/pubmed/12468711.

[75] Pleiss, J., 2011. Protein design in metabolic engineering and synthetic biology. Current opinion in biotechnology 22, 611–7. URL: http://www.ncbi.nlm.nih.gov/pubmed/21514140, doi:10.1016/j.copbio.2011.03.004.

[76] Raghavendra, P., 2008. Optimal algorithms and inapproximability results for every CSP?, in: Proceedings of the 40th annual ACM symposium on Theory of computing, ACM. pp. 245–254.

[77] Raha, K., Wollacott, A.M., Italia, M.J., Desjarlais, J.R., 2000. Prediction of amino acid sequence from structure. Protein science : a publication of the Protein Society 9, 1106–19. URL: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2144664\&tool=pmcentrez\&rendertype=abstract, doi:10.1110/ps.9.6.1106.

[78] Rendl, F., Rinaldi, G., Wiegele, A., 2010. Solving Max-Cut to optimality by intersecting semidefinite and polyhedral relaxations. Math. Programming 121, 307.

[79] Schiex, T., 2000. Arc consistency for soft constraints, in: Principles and Practice of Constraint Programming - CP 2000, Singapore. pp. 411–424.

[80] Sontag, D., Choe, D.K., Li, Y., 2012. Efficiently searching for frustrated cycles in MAP inference, in: Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence (UAI-12), AUAI Press, Corvallis, Oregon. pp. 795–804.

[81] Sontag, D., Meltzer, T., Globerson, A., Weiss, Y., Jaakkola, T., 2008. Tightening LP relaxations for MAP using message-passing, in: 24th Conference in Uncertainty in Artificial Intelligence, AUAI Press. pp. 503–510.

[82] Swain, M., Kemp, G., 2001. A CLP approach to the protein side-chain placement problem, in: Principles and Practice of Constraint Programming–CP 2001, Springer. pp. 479–493.

[83] Traoré, S., Allouche, D., André, I., de Givry, S., Katsirelos, G., Schiex, T., Barbe, S., 2013. A new framework for computational protein design through cost function network optimization. Bioinformatics 29, 2129–2136.

[84] Voigt, C.A., Gordon, D.B., Mayo, S.L., 2000. Trading accuracy for speed: A quantitative comparison of search algorithms in protein sequence design. Journal of molecular biology 299, 789–803. URL: http://www.ncbi.nlm.nih.gov/pubmed/10835284, doi:10.1006/jmbi.2000.3758.

[85] Wallace, R., 1995. Directed arc consistency preprocessing, in: Meyer, M. (Ed.), Selected papers from the ECAI-94 Workshop on Constraint Processing. Springer, Berlin. number 923 in LNCS, pp. 121–137.

[86] Yanover, C., Meltzer, T., Weiss, Y., 2006. Linear programming relaxations and belief propagation–an empirical study. The Journal of Machine Learning Research 7, 1887–1907.

[87] Zhang, P.Y., Romero, D.A., Beck, J.C., Amon, C.H., 2013. Solving wind farm layout optimization with mixed integer programming and constraint programming, in: Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CP-AI-OR), Springer. pp. 284–299.

3.3 A new framework for computational protein design through cost function network optimization

Next, we addressed another challenge of CPD which is the enumeration of a gap free list of suboptimal solutions. The generation of large ensembles of suboptimal solutions enables to compensate approximations introduced in the modelling of the design problem. While the task of finding a set of near-optimal models proved to be an insurmountable computational hurdle for DEE/A*, the CFN-based methods successfully solved 30 out of the 35 design cases tested.

Combining the CFN solver *toulbar2* with the CPD-dedicated software *osprey*, we conceived a framework which goes from the modeling of the CPD system to the combinatorial optimization phase based on the last CFN developments. We also introduced new selection criteria for defining the allowed amino acids at each variable position taking into account its location in the 3D structure. This methodology relies on a more precise measurement of the burial of residues compared to other methods described in the literature which are rather based on the measurement of the Solvent Accessible Surface Area of residues [3]–[5].

This work was described in an article published in the *Bioinformatics* journal [6] and enclosed hereafter.

A New Framework for Computational Protein Design through Cost Function Network Optimization

Seydou Traoré [1-3#], David Allouche[4#], Isabelle André[1-3], Simon de Givry[4], George Katsirelos[4], Thomas Schiex[4*], Sophie Barbe[1-3*]

[1]Université de Toulouse;INSA,UPS,INP; LISBP, 135 Avenue de Rangueil, F-31077 Toulouse, France
[2]INRA, UMR792, Ingénierie des Systèmes Biologiques et des Procédés, F-31400 Toulouse, France
[3]CNRS, UMR5504, F-31400 Toulouse, France
[4]Unité de Biométrie et Intelligence Artificielle, UR 875, INRA, F-31320 Castanet Tolosan, France

ABSTRACT

Motivation: The main challenge for computational structure-based protein design (CPD) remains the combinatorial nature of the search space. Even in its simplest fixed-backbone formulation, CPD encompasses a computationally difficult NP-hard problem that prevents the exact exploration of complex systems defining large sequence-conformation spaces.

Results: We present here a CPD framework, based on Cost Function Network (CFN) solving, a recent exact combinatorial optimization technique, to efficiently handle highly complex combinatorial spaces encountered in various protein design problems. We show that the CFN-based approach is able to solve to optimality a variety of complex designs that could often not be solved using a usual CPD-dedicated tool or state-of-the-art exact Operations Research tools. Beyond the identification of the optimal solution, the global minimum energy conformation (GMEC), the CFN-based method is also able to quickly enumerate large ensembles of sub-optimal solutions of interest to rationally build experimental enzyme mutant libraries.

Availability: The combined pipeline used to generate energetic models (based on a patched version of the open source solver osprey 2.0), the conversion to CFN models (based on Perl scripts) and CFN solving (based on the open source solver toulbar2) are all available at http://snp.toulouse.inra.fr/~tschiex/SpeedUp.tgz.

Contacts: Sophie.Barbe@insa-toulouse.fr and Thomas.Schiex@toulouse.inra.fr

Supplementary information: Available at Bioinformatics online.

1 INTRODUCTION

The engineering of tailored proteins with desired properties holds great interest for applications ranging from medicine, biotechnology (Nestl et al., 2011), and synthetic biology (Pleiss, 2011) to nanotechnologies (Grunwald et al., 2009). Although, directed evolution techniques coupled with high-throughput automated procedures have met with some success, they do not provide structural design principles to guide the rational design of novel proteins. The development of generic and effective protein engineering methodologies, both experimental and computational, is thus of utmost interest to speed-up the design of tailored proteins having the desired properties. Structure-based computational protein design (CPD) approaches have demonstrated their potential to adequately capture fundamental aspects of molecular recognition and interactions which have already enabled the successful (re)design of several enzymes for various purposes (Hellinga and Richards, 1991; Dahiyat and Mayo, 1997; Looger et al., 2003; Khare et al., 2012). Despite these outstanding results, the efficiency, predictability and reliability of CPD methods have shown that they still need to mature.

* To whom correspondence should be addressed.
# These authors contributed equally to this work.

CPD is faced with several challenges. The first lies in the exponential size of the conformational and protein sequence space that has to be explored which rapidly grows out of reach of computational approaches. Another obstacle to overcome is the unsolved issue of accurate structure prediction for a given sequence. Therefore, the design problem is usually approached as an inverse folding problem (Pabo, 1983), in order to reduce the problem to the identification of an amino acid sequence that can fold into a target protein 3D-scaffold which matches the design objective (Bowie et al., 1991). This paradigm typically assumes a fixed protein backbone and, for each type of amino acid considered at a given position, allows the side-chains to move only among a set of discrete and low-energy conformations, called rotamers (Janin et al., 1978). CPD is thus formulated as an optimization problem which consists in choosing combinations of rotamers at designable specified positions such that the fold is stabilized and the desired property is achieved. In order to solve this problem, we need a computationally tractable energetic model to evaluate the energy of any combination of rotamers. We also require computational optimization techniques that can efficiently explore the sequence-conformation space to find the sequence-conformation model of global minimum energy (GMEC: Global Minimum-Energy Conformation) or an ensemble of low-energy sequence-conformation models. Indeed, several reasons can motivate the generation of multiple near-optimal solutions. First, the sequence-conformation model with the lowest predicted energy may not fold into the targeted protein scaffold due to inaccuracies in the modeling of protein energetics (Kuhlman et al., 2003). Secondly, the GMEC solution may be so stabilized that it can lack the flexibility required to operate the protein biological function (Arnold, 2001). Such sub-optimal ensembles can then be analyzed in order to rationally guide the experimental construction of protein libraries while enhancing the chances of success to identify a protein hit.

The protein design problem modeled with a rigid backbone, a discrete set of rotamers, and pairwise energy functions, has been proven to be NP-hard (Pierce and Winfree, 2002). Hence several meta-heuristic methods have been applied to it, including Monte-Carlo with simulated annealing (Kuhlman and Baker, 2000; Voigt *et al.*, 2000), genetic algorithms (Desjarlais and Handel, 1995; Raha et al., 2000), and other methods (Wernisch et al., 2000; Desmet et al., 2002; Allen and Mayo, 2006). These approaches can usually find a relatively low-energy model fairly quickly but without any guarantees of completeness or accuracy. Indeed, these stochastic optimization routines may end up trapped in local minima and miss the GMEC with no indication.

Conversely, there exist methods which solve the GMEC exactly, such as approaches based on the Dead-End Elimination (DEE) theorem (Desmet et al., 1992), on the Branch-and-Bound algorithm (Gordon and Mayo, 1999; Wernisch et al., 2000; Hong et al., 2009), on integer linear programming (Althaus *et al.*, 2002; C. L. Kingsford *et al.*, 2005) or on dynamic programming (Leaver-Fay et al., 2005). These exact methods offer several advantages. First, they ensure that discrepancies between CPD predictions and experimental results come exclusively from the inadequacies of the biophysical model and not from the algorithm. Next, because provable methods can determine that the optimum is reached, they may actually stop before meta-heuristic approaches. Finally, empirical studies on solving the GMEC problem reported that the accuracy of meta-heuristic approaches tend to degrade as the problem size increases (Voigt *et al.*, 2000).

In this paper, we modeled the CPD problem as either a binary Cost Function Network (CFN) or an Integer Linear Programming (ILP) problem (Section 2.4). We compared the performance of the open source CFN solver *toulbar2* and the IBM™ ILOG ILP solver *cplex* against that of the combined DEE/A* approach as implemented in the dedicated CPD software *osprey* (Section 3.2),

for design problems (Section 3.1). The CFN-based method outperformed by several orders of magnitude the other methods both in identifying the GMEC but also in producing a set of low-energy sequence-conformation models. This second step was not attainable in most of the study cases using DEE/A* (Section 3.4). Therefore, on the basis of the CFN approach, we propose a new CPD framework (Fig. 1). Our methodology, which we describe in Section 2, is well-adapted to solving exactly macromolecular design problems of large sequence-conformation spaces. It also has the potential to improve methods that integrate flexibility to a larger extent in protein design, as this considerably expands the size of the search space or may require solving a large number of GMEC instances (Humphris and Kortemme, 2008; Hallen *et al.*, 2013). These aspects are highly relevant to CPD and we shall address them here.

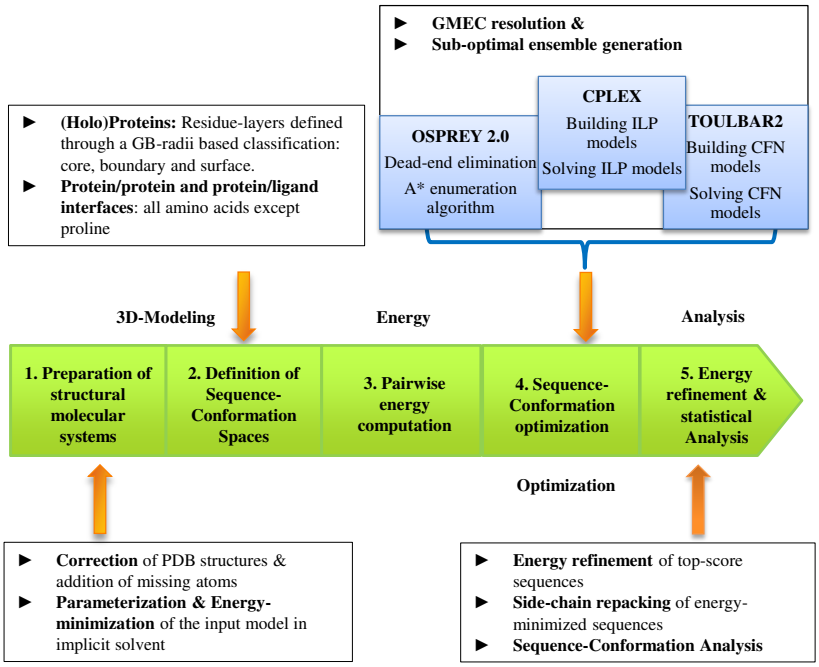


Fig. 1 Flow chart of the CPD framework

2 METHODS

The CPD strategy introduced in this work (Fig. 1) is composed of five main stages discussed in more details in the following sub-sections. The whole CPD framework and the approaches used to handle the sequence-conformation combinatorial optimization problem were assessed for the design of more stable proteins and cofactor-bound proteins as well as protein-ligand and protein-protein interfaces.

2.1 Preparation of structural molecular systems

Three-dimensional models of proteins in free and complex states were derived from their respective X-ray structures deposited in the PDB (Protein Data Bank) (Bernstein *et al.*, 1977) (Table S1). Missing heavy atoms in crystal structures as well as hydrogen atoms were added using the *tleap* module of the *Amber 9* software package (Case *et al.*, 2006). Cofactors as well as crystallographic water molecules specified in *SITE* and *LINK* records of PDB files were kept in structural models. Histidine protonation states and disulfide bonds were assigned using the *tleap*

module. For multimeric proteins, the transformation matrix specified in the PDB file was applied to reproduce missing chains. Parameters for non-amino acid type ligands and co-factors were generated with the *Antechamber* module of *Amber 9* (Wang *et al.*, 2006). The molecular all-atom *ff99SB* (Hornak *et al.*, 2006), *Glycam06* (Kirschner *et al.*, 2008) and *gaff* force fields (Wang *et al.*, 2004) were used for the proteins, carbohydrates and other non-standard molecules, respectively. Each molecular system was then subjected to 500 steps of minimization with the *Sander* module of *Amber 9,* using the Generalized Born/Surface Area implicit solvent model (Hawkins *et al.*, 1996).

2.2 Definition of sequence-conformation spaces

The residues of each protein were classified into three layers (labeled core, boundary or surface) according to their burial in the 3D-model (Fig. S1). This burial of residues was defined by calculating their solvation radius from atomic solvation radii, as defined by (Archontis and Simonson, 2005). The solvation radius B_R of residue R is given by:

$$B_R = \frac{\sum_{i \in R} q_i^2}{\sum_{i \in R} \frac{q_i^2}{b_i}} \quad (1)$$

where b_i and q_i are the atomic solvation radius of atom i from residue R and its partial charge, respectively. From these calculations, three layers of decreasing residue-solvation radius from the core to the surface of the protein were defined. The set of amino acid types considered at each mutable residue (i.e., candidate positions for redesign) of the proteins (in their apo form or bound to a cofactor) depends on the layer to which the residue belongs to. Further details regarding the selection of designed positions and the allowed amino acids can be found in the Supplementary data.

2.3 Computation of pairwise energies

The total energy (E_{total}) of a sequence-conformation model defined by the selection of one specific amino acid associated with a given conformation (rotamer) for each variable amino acid type is assessed as follows:

$$E_{\text{total}} = E_c + \sum_i E(i_r) + \sum_{i,j} E(i_r, j_s) \quad (2)$$

where E_c is a constant energy contribution capturing interactions between fixed parts of the model, $E(i_r) = E_{self}(i_r) - E_{\text{ref}}(i_r)$ is the difference between the self energy of rotamer r at position i capturing internal interactions or interactions with fixed regions $E_{self}(i_r)$ and its reference energy $E_{\text{ref}}(i_r)$ which corresponds to the lowest computed intra-rotamer energy for each amino acid type by variable residue position (Lippow *et al.*, 2007) and $E(i_r, j_s)$ is the pairwise interaction energy between rotamer r at position i and rotamer s at position j. In this formulation, the conformations (*i.e.*, rotamers) of amino acid-type ligands are processed as rotamers of amino acid side-chains.

All pairwise energy terms were pre-computed and stored using *osprey 2.0* (Gainza *et al.*, 2013). These calculations were based on the *Amber* all-atom *ff94* parameters implemented in *osprey 2.0* as well as on additional force field parameters generated from *Glycam06* and *gaff* force fields using the *Antechamber* module of *Amber 9*. These parameters were added in the parameter files

of *osprey 2.0* (Chen *et al.*, 2009) and were used for modeling carbohydrates and other non-standard molecules. The energy functions consisted in the sum of the *Amber* electrostatic terms (with a distance-dependent dielectric constant), van der Waals and dihedral energy terms and the EEF1 implicit solvation energy term. No cut-off was used for non-bonded interactions.

2.4 Sequence-Conformation optimization

The problem of finding the set of rotamers that will optimize the total energy (E_{total}) was modeled as either a binary Cost Function Network (CFN) or an Integer Linear Programming (ILP) problems. The complete interaction graph and large tree-width excluded the use of dynamic programming (Leaver-Fay et al., 2005). The performance of CFN and ILP was compared against that of the combined DEE/A* CPD-dedicated approach.

Cost Function Network model (CFN)

A *Cost Function Network*, or Weighted Constraint Satisfaction Problem P, is composed of a set of local cost functions, each involving a set of specific variables (Schiex *et al.*, 1995). CFNs have been used as a modeling framework for representing and solving various combinatorial optimization problems in bioinformatics and resource allocation (Cabon *et al.*, 1999; de Givry *et al.*, 2006; Zytnicki *et al.*, 2008).

Formally, a CFN P is a triple $P = (X, D, C)$ where $X = \{1,2,..,n\}$ is a set of n variables. Each variable $i \in X$ has a discrete domain $d_i \in \mathrm{D}$. C is a set of local cost functions. Each cost function $c_S \in C$ is defined over a subset of variables $S \subseteq X$ (called its scope), has domain $\prod d_i$ and takes its values in $\mathbb{N} \cup \{+\infty\}$. Cost functions must be non negative but are otherwise totally arbitrary and are often described by cost tables. The infinite cost is used to represent hard constraints. An assignment A is a mapping from variables to values from their domains. The cost of an assignment A for a local cost function is the value of the cost function for the projection of A to the scope of the function. The global cost of A is the sum of the costs of A over all local cost functions. It is usually assumed that C contains one constant cost function, with an empty scope, denoted as $c_\emptyset$. A CFN P defines a joint cost distribution over all the variables X defined by the cost of the assignments. Since all cost functions in a CFN are non negative, the constant cost function $c_\emptyset \in C$ defines a lower bound on this joint cost distribution. Solving a CFN consists in finding an assignment that minimizes the joint cost distribution.

Modeling a CPD problem as a CFN is straightforward. Each mutable or flexible residue i is represented by a variable i. The set of rotamers available to the residue defines the domain d_i of the variable i. Finally, each interaction energy term in E_{total} can be represented as a cost function. The constant term E_c is captured as a constant cost function with empty scope. The terms $E(i_r)$, which depend on one residue only, are captured as unary cost functions involving one variable i each. Finally, interaction terms $E(i_r, j_s)$ can be captured as binary cost functions involving variables i and j. To enforce the non negativity requirement on cost functions, one can simply subtract the minimum of every cost function from its cost table. The joint cost distribution defined by the corresponding CFN is then equal to the energy, up to a known constant shift. The optimal solution of the CFN is an assignment that corresponds to a GMEC for the CPD problem. When the maximum number of available rotamers over all residues is d, the resulting binary CFN takes space in $O(n^2 d^2)$.

Solving a CFN consists in finding a combination of values for all the variables in X that minimizes the joint cost distribution. Such an optimal solution defines a GMEC for the CPD

problem. Existing exact algorithms for solving CFN are usually Depth-First Branch and Bound (DFBB) algorithms integrating strong incrementally computed polynomial time lower bounds. The use of depth-first search algorithms avoids the worst-case exponential space of the A* algorithm used in the DEE/A* approach. The incremental lower bounds are produced by algorithms that enforce so-called local consistencies (Schiex, 2000; Larrosa and Schiex, 2004). Enforcing a local consistency on a given CFN P transforms it to an *equivalent* CFN P' (defining the same joint cost distribution) with a possibly increased constant cost function $c_\emptyset$, providing an improved lower bound. This lower bound is used to prune the search tree and to delete rotamers.
To identify the GMEC of all CPD instances, we used the open source *toulbar2*[1] solver (release 0.9.7.0) with options `-d: -l=3 -m` and no initial upper bound. By default, *toulbar2* maintains Existential Directional Arc Consistency (de Givry *et al.*, 2005) for incremental lower bounding, dynamic value ordering (based on minimum unary cost) and a variable ordering heuristics (based on the median energy of terms involving a given residue following preprocessing) combined with last conflict heuristics (Lecoutre *et al.*, 2009)). The counterpoint to the improved space complexity of a DFBB search instead of A* is that DFBB search cannot directly provide a sorted list of sub-optimal solutions. To enumerate all sub-optimal solutions within $E_{\text{cut}} = 2.0\ \text{kcal}.mol^{-1}$ of the GMEC, we first computed the GMEC and its associated energy E_{GMEC} as above and performed a second exhaustive search for all solutions with energy below $E_{\text{GMEC}} + E_{\text{cut}}$ (options `-a` and `-ub` in *toulbar2* to respectively produce all solutions and set a global upper bound).

Integer Linear Programming model (ILP)
We also modeled the CPD problem as an integer linear program (01LP) problem using the usual translation from CFN to ILP initially proposed in (Koster et al., 1999). An ILP is defined by a linear criteria and a set of linear constraints on integer variables. For every value/rotamer i_r of the variable/residue i, we introduced one Boolean variable d_{i_r} that indicates whether the rotamer i_r is used ($d_{i_r} = 1$) or not ($d_{i_r} = 0$). In order to enable the expression of the energy as a linear function of variables, we introduced an extra Boolean variable $p_{\mathrm{i_r j_s}}$ for every pair of rotamers (i_r, j_s), capturing the fact that this pair of rotamers is used. The energy can then be expressed directly as the linear function to be minimized (the constant term can be ignored as it cannot change the optimal solution):

$$\sum_{i,r} E(i_r).\, d_{i_r} + \sum_{i,r,j,s} E(i_r, j_s).\, p_{\mathrm{i_r j_s}} \quad \textbf{(3)}$$

Additional constraints enforce that exactly one rotamer is selected for each variable position and that a pair is used if and only if the corresponding values are used. Then, finding a GMEC reduces to the following ILP:

$$min \sum_{i,r} E(i_r).\, d_{i_r} + \sum_{i,r,j,s} E(i_r, j_s).\, p_{i_r j_s} \quad \textbf{(4)}$$

such that:

$$\sum_r d_{i_r} = 1(\forall i)$$

$$\sum_s p_{i_r j_s} = d_{i_r}(\forall i, r, j)$$

[1] Toulbar2 is an international collaborative CFN solver development. All sources are available on our software forge at http://mulcyber.toulouse.inra.fr/projects/toulbar2.

The resulting ILP contains $O(n^2d^2)$ variables and $O(n^2d)$ constraints. It is equivalent to the model proposed for CPD in (Kingsford *et al.*, 2005). Note that since the objective function is non-linear, it is fundamentally impossible to express it in ILP without introducing a quadratic number of variables. Hence, this ILP model cannot be improved significantly in size.
To identify the GMEC, we used the IBM™ ILOG ILP solver *cplex* (version 12.4.0.0) on this ILP with parameters EPAGAP, EPGAP and EPINT set to zero to avoid premature stop.

Dead-End-Elimination / A combined approach (DEE/A*)*
The DEE algorithm iteratively eliminates rotamers and pairs of rotamers which cannot possibly be part of the GMEC (Desmet *et al.*, 1992) by using a dominance criterion. The original DEE single elimination criterion removes a rotamer r at position i if there exists another rotamer u at the same position such that:

$$E(i_r) - E(i_u) + \sum_{j \neq i} min_s E(i_r, j_s) - \sum_{j \neq i} max_s E(i_u, j_s) > 0 \quad (5)$$

In this case, r can be removed in this case because any conformation using r can be transformed into a lower energy conformation by substituting u for r. The pruning criterion is applied until a single solution remains (*i.e.*, the GMEC) or all solutions outside an energy window of E_{cut} have been pruned or otherwise when no more pruning is identified during a given round. The DEE pruning step is followed by an A* Branch-and-Bound like search which uses the remaining rotamers (Leach *et al.*, 1998) to identify the GMEC or produces a sorted list of all models whose energy is within a specified energy E_{cut} of the GMEC energy. The A* algorithm is a worst-case exponential space and exponential time algorithm whose efficiency is tightly linked to the quality of the heuristic admissible evaluation function used to decide which node to explore next. Interestingly, the heuristic used in the A* approach applied in this study is equivalent to the CFN heuristic used in the PFC-DAC algorithm (Wallace, 1996). This lower bound as well as an improved variant of it (Larrosa *et al.*, 1998) have been obsoleted by the incremental local consistencies introduced in (Schiex, 2000).

In this study, we used *osprey 2.0* to perform the DEE/A* procedure in order to find the GMEC and sub-optimal models within a $E_{\text{cut}} = 2.0$ kcal. mol^{-1} of the GMEC energy (option $initEw = 2$). The procedure starts by extensive DEE ($algoOption = 3$ which enables simple Goldstein, Magic bullet pairs, 1 and 2-split positions, Bounds and pairs pruning) and is followed by the A* search. We also optionally applied a procedure including a pre-filtering step before the DEE which eliminates rotamers i_r such that $E(i_r) > 30$ kcal.mol^{-1} and pairs (i_r, j_s) such that $E(i_r, j_s) > 100$ kcal.mol^{-1} ($pruning$ and $stericE$ parameters).

All computations (*toulbar2*, *cplex* and *osprey*) were performed on one core of an AMD Opteron™ Processor 6176@2.3 GHz. We used 128GB of RAM and a 100 hours timeout.

2.5 Analysis of top-score models

For 4 design cases (1TEN, 1UBI, 2PCY, 1CSK), the 3D structure of the best conformation of each unique sequence found within a 2 kcal.mol^{-1} window of the GMEC energy was built using *osprey 2*. These 3D structure models were then subjected to 1000 steps of minimization with the *Sander* module of *Amber 9,* using the Generalized Born/Surface Area implicit solvent model.

During these minimizations, heavy atom positions were restrained using a harmonic potential (force constant = 1 kcal.mol^{-1}.Å^{-2}). The score of minimized structures was computed with *osprey 2.0*. Finally, the conformational variability of the minimized models was assessed by carrying out an optimization step allowing all variable amino acids (mutable and flexible) to repack. This step was performed using the CFN-based approach with a E_{cut} of 0.2 kcal.mol^{-1} and involved the pre-computing of pairwise energy terms for each minimized model using *osprey 2.*

3 RESULTS AND DISCUSSION

3.1 Benchmark set

A tailored benchmark set was produced in order to assess the performance of combinatorial optimization algorithms on sequence-conformation spaces of various sizes and complexity as well as the potential of the CPD methodology proposed herein (Fig. 1) for tackling the redesign of diverse structural systems involving free proteins or proteins bound to a cofactor, a ligand or a protein. The studied systems have all been extracted from previously published papers about protein engineering, *in silico* protein design or protein structural studies (see references Table S1). A detailed description of our benchmark preparation protocol is given in Supplementary data.

In our benchmark set, the number of mutable residues varies from 3 to 119 (Table S1). They are located either in the core of (holo)proteins (except when data is available in the literature) or at the protein-protein or protein-ligand interfaces. We then defined a set of flexible residues (from 1 to 93) (Table S1) that surrounds mutable positions and mainly occupies the core and the boundary regions of proteins. The resulting number of variable residues ranges then from 23 to 120 and, given the *penultimate* rotamer library used, from 3 to 194 amino acid rotamers were considered at each variable position. Our resulting benchmark set covers thus a wide range of combinatorial spaces (from 4.10^{26} to 2.10^{249}) and allows us to evaluate different combinatorial optimization problems of varying complexity.

3.2 GMEC-based design

First, we evaluated the performance of CFN and ILP methods for solving the GMEC identification problem exactly and compared them against the exact CPD-specific method DEE/A*. We compared the CFN solver *toulbar2,* the ILP solver *cplex* and the DEE/A* implemented in the *osprey* software on the benchmark set of 35 design cases (Table S1) and present the results in Table 1.

Out of the 35 design cases, the CFN solver toulbar2 and the ILP solver cplex solved respectively 30 and 27 cases within the 100-hour time-out whereas the DEE/A* CPD-dedicated approach identified only 18 GMEC (Table 1). The A* step tends to be the time consuming step of the combined DEE/A* approach. Table S3 gives fractions of times spent during DEE and A* for each instances (the GMEC problem). The DEE step converges in 30 cases while spending at worst 22.5% of the runtime in 29 cases, leaving A* with the remaining 77.5% run-time. Application of the CFN approach after DEE leads to shorter runtime and all the remaining GMECs are finally solved (Table S3).

The DEE/A* method managed to find the 18 GMEC with CPU times ranging from a few minutes to over one hour. Only four cases (1MJC, 1CSK, 1SHF and 1NXB) corresponding to the exploration of small combinatorial spaces on small proteins took less than one minute to be solved. The DEE step converged to a single solution in only 3 instances (1MJC, 1NXB, 1CSE)

(Table S3). The A* search successfully identified 15 further GMEC using the sub-set of rotamers remaining after the DEE step. It failed for 12 instances due to time limit (10 cases) or memory limit (2 cases). In these cases, the size of the combinatorial search space of non-pruned rotamers after the DEE step ranged ~ 10^{19} to 10^{36}. This was not a sufficient reduction to enable A* to extract the GMEC from the sub-set of remaining rotamers with the available computational resources. Finally, the DEE step failed to complete within the 100 hours time-out for 5 tests covering initial combinatorial spaces from ~ 10^{61} to 10^{249} and the highest number of variable residues at the surface of the proteins. One can expect the surface residues to be more flexible than the buried residues and such flexibility may then lead to a high density of conformations with similar energy and a corresponding increase in the complexity of the combinatorial space to be explored. These results clearly underline the limits of the DEE-based approach to handle large and complex problems.

Table 1: CPU-time for solving the GMEC using DEE/A* (*osprey*), ILP (*cplex*) and CFN (*toulbar2*).

PDB	Sequence Conformation Space Size	Times (s)		
		DEE/A*	ILP	CFN
1MJC	4.36e+26	4.57.	3.94	0.08
1CSP	5.02e+30	200.00	360.00	0.84
1BK2	1.18e+32	93.20	303.00	0.65
1SHG	2.13e+32	138.00	42.30	0.25
1CSK	4.09e+32	41.70	24.90	0.15
1SHF	1.05e+34	44.30	11.10	0.17
1FYN	5.04e+36	622.00	2.26e+03	3.79
1PIN	5.32e+39	9.54e+03	3.00e+03	3.99
1NXB	2.61e+41	11.10	21.20	0.24
1TEN	6.17e+43	113.00	81.70	0.33
1POH	8.02e+43	77.90	31.80	0.45
2DRI	1.16e+47	T_{A*}	2.92e+5	24.5
1FNA	3.02e+47	3.31e+03	419.00	0.73
1UBI	2.43e+49	T_{A*}	704.00	2.14
1C9O	3.77e+49	2.31e+03	1.40e+03	2.20
1CTF	3.95e+51	T_{A*}	580.40	1.23
2PCY	2.34e+52	2.08e+03	76.70	0.51
1DKT	3.94e+58	5.42e+03	1.85e+03	3.95
2TRX	9.02e+59	487.00	1.34e+03	1.7
1PGB	5.10e+61	T_{DEE}	T	T
1CM1	3.73e+63	T_{A*}	2.65e+04	19.2
1BRS	1.67e+64	T_{A*}	2.39e+05	426.0
1ENH	6.65e+64	T_{DEE}	T	T
1CDL	5.68e+65	T_{A*}	T	191.1
1LZ1	1.04e+72	T_{A*}	1.25e+03	2.11
2CI2	7.26e+75	T_{DEE}	T	T
1GVP	1.51e+78	T_{A*}	T	593.6
1RIS	1.23e+80	T_{A*}	T	501.00
2RN2	3.68e+80	M_{A*}	1.14e+03	2.77
1CSE	8.35e+82	367.00	205.93	1.36
1HNG	3.70e+88	5.59e+03	4.15e+03	6.97
3CHY	2.36e+92	T_{A*}	2.91e+04	171.00
1L63	2.17e+94	M_{A*}	2.82e+03	6.41
3HHR	2.98e+171	T_{DEE}	M	T
1STN	2.00e+249	T_{DEE}	M	M
# Nb of cases solved in 10 min		11	13	30
# Nb of cases solved in 100 h		18	27	30

A 'M' indicates an exceeded memory size (128G) and a 'T' indicates an exceeded computation time (100h). For the DEE/A* approach, the A* and the DEE associated with M or T indicate the step during which occurred the exceeding of memory or computation time.

In order to improve the efficiency of the DEE-based approach, a pre-processing is usually applied to eliminate rotamers and pairs of rotamers of high energy which are not expected to appear in the GMEC. We then performed such pre-filtering using a 30 kcal.mol^{-1} threshold for rotamers

and a 100 kcal.mol^{-1} threshold for pairs of rotamers. Using these parameters in one instance (1CSE), the optimal solution after preprocessing did not match the GMEC obtained in the absence of this pre-processing step. However, when we increased the threshold for rotamers 30 to 50 kcal.mol^{-1}, the optimal solution remained the same with and without preprocessing. Besides losing the guarantee of optimality, the preprocessing step did not improve the performance of DEE/A* in terms of the number of instances that were solved (18 GMEC identified out of the 35 cases) (Table S3). Furthermore, 7 instances required more CPU time for identifying the GMEC when preprocessing was used.

The ILP solver cplex solved 9 additional instances from our benchmark set (2DRI, 1UBI, 1CTF, 1CM1, 1BRS, 1LZ1, 2RN2, 3CHY, 1L63) compared to DEE/A* (Table 1). In all these cases, DEE/A* timed out during the A* search. Among these 9 design cases, only 2 (3CHY and 1L63) cover combinatorial spaces of greater size (~ 1092 and 1094) than the largest combinatorial problem (1HNG ~ 1088) solved by DEE/A* (Table 1). The other 18 instances solved by the *cplex* are the ones which were successfully solved by DEE/A*. Although the ILP solver was faster in 13 of these 18 instances, the time is overall similar for both methods. Nevertheless, the total number of solved instances shows that the ILP solvers can be more efficient for several design cases, as previously reported (Allouche et al., 2012). More concise Quadratic Programming (cplex QP solver) and Partial Weighted MaxSAT (akmaxsat, MaxHS and Bin-Core-Dis solvers) models were also tried, to no avail.

The CFN solver *toulbar2* solved respectively 12 and 3 more cases compared to the DEE/A* and *cplex* (Table 1). Therefore, CFN only failed on 5 instances (1PGB, 1ENH, 2CI2, 3HHR, 1STN) out of the 35 handled. These instances correspond to vast combinatorial spaces (from about 10^{61} to 10^{249}) which mostly include variable residues scattered over the three layers of the proteins (Table S1). There are no cases solved by DEE/A* or ILP that CFN could not solve. Moreover, CFN outperformed the two other approaches by an impressive margin in terms of speed. Among the 30 instances successfully handled by CFN and including large combinatorial spaces, 11 cases were solved in less than one second, 23 in less than 10 seconds and only 5 instances required a few minutes. Given its running time performance and its success rate for handling large protein design problems, the CFN approach appears as an appealing alternative to current exact CPD-dedicated methods, especially for solving highly complex GMEC-based design problems.

There is no simple explanation for the performance advantage of the CFN solver *toulbar2* over the ILP solver *cplex* and the DEE/A* implemented in the *osprey*. Indeed, solvers are complex systems involving various mechanisms. The effect of their interactions during solving is hard to predict. Moreover, the IBM™ ILOG ILP solver *cplex* is a totally closed-source industrial black box solver.

Compared to the CFN solver *toulbar2, osprey* uses an obsolete lower bound instead of the more recent incremental and stronger lower bounds offered by soft local consistencies such as EDAC (Larrosa *et al.*, 2005). This, together with the associated informed value ordering provided by these local consistencies, may explain why the CFN approach outperforms the DEE/A* method. Considering ILP, it is known that the LP relaxation lower bound used in ILP is (by duality) the same as the Optimal Soft AC (Cooper *et al.*, 2010) lower bound when no upper bounding occurs. Since OSAC dominates all other local consistencies at the arc level, this provides an explanation for the efficiency of *cplex* compared to *osprey*. Finally, compared to ILP, the formulation of the deeply non linear problem is more direct in CFN. This probably contributes, together with the

upper bounding, variable and value ordering heuristics of *toulbar2*, to the efficiency of the CFN approach compared to ILP.

3.3 Sub-optimal ensemble generation

We also performed computational design tests to assess the ability of the CFN method to generate an ensemble of provably near-optimal sequence-conformation models in addition to the GMEC. The performance of the CFN approach was compared against DEE/A*. For this purpose, the 35 design cases of the benchmark set (Table S1) were again investigated using the CFN *toulbar2* solver and the DEE/A* implemented in *osprey* software to access the set of sequence-conformation models comprised within an energy window of 2 kcal.mol^{-1} of the GMEC energy.
Out of the 35 design cases, the CFN solver *toulbar2* managed to produce the sub-optimal ensembles of solutions for 30 design cases whereas the DEE/A* approach only successfully handled one instance (Table 2).

Table 2: CPU-time for generating the ensemble of sub-optimal models (E *cut* =2 kcal/mol) using DEE/A* (*osprey*) and CFN (*toulbar2*).

PDB	Times (s)		Number of sequence-conformation solutions	Number of different sequences
	DEE/A*	CFN		
1MJC	T_{A*}	41.99	2.11e+06	91
1CSP	M_{A*}	5.91	1.18e+05	794
1BK2	M_{A*}	6.89	3.18e+05	2.19e+03
1SHG	M_{A*}	13.38	6.46e+05	542
1CSK	M_{A*}	9.04	4.48e+05	199
1SHF	1.20e+05	0.92	3.07e+04	26
1FYN	T_{DEE}	14.40	4.05e+05	7.02e+03
1PIN	T_{DEE}	8.99	1.62e+04	310
1NXB	T_{A*}	1015.3	4.67e+07	526
1TEN	T_{A*}	135.7	6.42e+06	294
1POH	T_{A*}	1.74e+04	7.56e+08	177
2DRI	T_{DEE}	209.0	4.94e+06	340
1FNA	T_{A*}	2.19e+03	9.87e+07	3.94e+03
1UBI	T_{A*}	11.10	3.04e+05	194
1C9O	T_{DEE}	83.12	3.37e+06	1.65e+03
1CTF	T_{A*}	439.5	1.96e+07	3.06e+04
2PCY	T_{A*}	6.15	2.28e+05	144
1DKT	T_{A*}	2.37e+04	1.00e+09	2.83e+04
2TRX	T_{A*}	376.0	1.01e+09	132
1PGB	T_{DEE}	T	n.d	n.d
1CM1	T_{DEE}	117.8	3.01e+06	5.18e+03
1BRS	T_{DEE}	1.57e+03	2.10e+06	2.09e+04
1ENH	T_{DEE}	T	n.d	n.d
1CDL	T_{DEE}	669.4	1.22e+05	2.54e+03
1LZ1	T_{A*}	263.8	9.87e+06	546
2CI2	T_{DEE}	T	n.d	n.d
1GVP	T_{DEE}	1.66e+04	4.32e+08	3.13e+05
1RIS	T_{DEE}	3.01e+03	1.11e+08	1.24e+04
2RN2	T_{A*}	638.8	2.32e+07	4.00e+03
1CSE	T_{A*}	2.75e+03	8.00e+07	61
1HNG	T_{A*}	3.39e+03	1.08e+08	1.3e+03
3CHY	T_{A*}	370.0	3.52e+06	8.56e+03
1L63	T_{A*}	2.46e+04	8.00e+08	6.03e+03
3HHR	T_{DEE}	M	n.d	n.d
1STN	T_{DEE}	M	n.d	n.d

A 'M' indicates an exceeded memory size (128G) and a 'T' indicates an exceeded computation time (100h). For the DEE/A* approach, the A* and the DEE associated with M or T indicate the step during which occurred the exceeding of memory or computation time.

The DEE/A* approach failed for 34 instances due to time (30 cases) or memory (4 cases) limits. It only identified the set of near-optimal models for one instance (1SHF) among the 18 successfully handled for the GMEC problem (Table 1). Although this solved instance

corresponds to one of the smallest investigated combinatorial spaces (~10^{34}) (Table S1), ~ 37 hours of computation were needed to find the ensemble of low-energy models compared to less than one second for the CFN approach. This running time is even larger than the ~7 hours required by the CFN method in the worst case (1L63) including an important combinatorial space (~10^{94}) and a large number of sub-optimal solutions ($8x10^8$).
Given the failure rate of the DEE/A* approach, we attempted to provide as input the information derived from the GMEC-based design tests carried out with the DEE/A* and CFN approaches to facilitate generation of the near-optimal ensembles. Despite having prior knowledge of the GMEC solution, the DEE/A* was not able to produce the set of sub-optimal models for the unsolved instances (data not shown).

The CFN method only failed to identify the near-optimal ensemble on the 5 instances (1PGB, 1ENH, 2CI2, 3HHR, 1STN) for which the GMEC problem was also unsolved (Table 1). In addition to the high success rate achieved by CFN, the method was also very efficient: 10 cases were solved by CFN in less than one minute, 18 required several minutes and only 4 instances needed several hours (Table 2).

While the task of finding a set of low-energy sequence-conformation models proved to be an insurmountable computational hurdle for DEE/A* as implemented in *osprey*, the CFN solver *toulbar2* successfully solved most of the design cases tested. Moreover, the CFN approach gave access to sets of provably sub-optimal solutions with outstanding running time performances.

The CFN approach efficiently uses the knowledge of the GMEC solution in the enumeration procedure of the near-optimal models. The GMEC defines an upper bound corresponding to the energy of the GMEC + 2 kcal.mol^{-1}. In CFN, this upper bound is systematically compared to the lower bound provided by local consistency enforcing. The DEE/A* implemented in *osprey* uses the same upper bound (parameter pruningE) but exploits a weaker lower bound. This likely explains the performance gap compared to the *toulbar2* CFN solver.

3.4 Sub-optimal ensemble analysis

First, we analyzed the sequence and conformational variability of the near-optimal models obtained for 4 design cases (1CSK, 1TEN, 1UBI, 2PCY) of proteins adopting distinct structural folds (Table S2). These instances include from 9 to 16 mutable residues and from 21 to 30 flexible residues (Table S1).

Within a window of 2 kcal.mol^{-1} of the GMEC, the CFN-based approach produced over 105 sequence-conformation models for each of the 4 design cases (Table 3). The score of these models is lower by as much as ~ 20 kcal.mol^{-1} than that of the wild-type model (Fig. S2c-S5c). In these ensembles of models, 144, 194, 199 and 294 unique sequences were found, respectively, for 2PCY, 1UBI, 1CSK and 1TEN design cases.

Only few unique sequences were then generated compared to the high number of enumerated models within a small energy window of 2 kcal.mol^{-1} of the GMEC energy. However, when the experimental construction of the protein library is considered, it is important to have access to a larger ensemble of distinct sequences. For this purpose, the outstanding performances of the CFN solver (Table 2) could be harnessed to provably predict sub-optimal models distributed on a wider energy window of the GMEC and thus attempt to generate more diverse sequence ensembles.

For the 4 design cases, the wild-type amino acid was the most often either substituted by an amino acid of slightly larger size or conserved (Fig. S2a-S5a). Nevertheless, the entire wild-type sequence of the protein was never found within the sub-optimal ensembles. It is noteworthy that the mutable residues of glycine type were not found substituted by another amino acid type (Fig. S2a, S3a, S4a). The number of conformations adopted by each sequence decreases gradually as the energy value of the sequence becomes more unfavorable (Fig. S2d-S5d). The superposition of the best conformation of each unique sequence showed that each flexible residue adopts almost the same rotamer in all best conformations (Fig. S2b-S5b). Moreover, the orientation of mutable residue side-chains is similar in all the best models regardless the assigned amino-acid type.

For the protein design problems studied here, we expected that mutations would favor the introduction of bulkier amino acids in order to fill up the free space available in the core of proteins. However, changes in amino acid sizes were subtle. A visual inspection of the 3D structures of mutants suggests that some slight adjustments of side-chains and/or backbone of surrounding residues could enable accommodation of larger side-chains which were here assigned with high interaction energies. This lack of conformational relaxation seems also to be at the origin of the observed conservation of glycine amino acid types. Therefore, the sequence selection may be biased and restricted by the lack of flexibility of surrounding residues. These results highlight the key role of the local molecular flexibility to extend the accessible sequence space, as shown by previous work (Bordner and Abagyan, 2004). We then subjected each unique sequence of the 4 sub-optimal ensembles to energy minimization in order to assess the effect of the relaxation of side-chains and backbone freedom degrees on the energy ranking of the sequence ensembles. Overall, the minimization decreased the energy values of these models from ~ 20 to 60 kcal.mol^{-1} depending on the design case (Fig. S2c-S5c). Nonetheless, the superposition of the structures before and after minimization only showed slight rearrangements of protein side chains and backbone. This clearly indicates that slight geometrical adjustments can significantly lower model energies. The conformational variability of these low energy sequences were further investigated by carrying out an optimization step (with a E_{cut} of 0.2 kcal.mol^{-1}) which enables all variable amino acids (mutable and flexible) to be repacked. Despite an E_{cut} value which is 10 times smaller, the number of conformations adopted by each unique sequence was found extremely high whatever the energy ranking of the sequence (Fig. S2d-S5d). Therefore, the significant differences observed among mutants before minimization step, is probably the result of the discretization of conformational freedom degrees. The minimization step thus allows us to extend the accessible conformational space. Current trends in CPD refine the exact DEE/A* approach along various directions, allowing respectively for continuous rotamers (I. Georgiev *et al.*, 2008; Gainza *et al.*, 2012), for continuous (Ivelin Georgiev and Bruce R. Donald, 2007) or discrete (I. Georgiev *et al.*, 2008) backbone conformation adjustments or both (Hallen *et al.*, 2013). The CFN approach can still be extended to handle such flexibilities descriptions.

In addition to lowering the energy and increasing the conformational variability of models, the geometry relaxation step re-ranks the sequence ensemble. The GMEC obtained after minimization (refined-GMEC) does not match with the original GMEC. The energy values of the minimized sub-optimal models are spread within an energy window up to ~ 6 kcal.mol^{-1} of the refined-GMEC (Fig. S2c-S5c). Therefore, with a E_{cut} of 2 kcal.mol^{-1}, from 92 to 185 unique sequences depending on the design case would be removed of these minimized ensembles. Even on a small energy window, these results demonstrate the advantage of the post-minimization to re-rank and post-screen the most promising candidate sequences to evaluate experimentally.

4 CONCLUSION

In this paper, we have formulated a novel open-source based computational framework to provably identify the GMEC as well as a set of low-energy protein sequences within the context of atomic protein design. This CFN-based approach provides remarkable speedups, allowing us to explore vast sequence-conformational spaces more efficiently than the DEE/A* algorithm or state-of-the-art ILP algorithms. Despite the significant change in terms of problem complexity, it is surprising to see that this efficiency extends to the generation of gap-free sets of sub-optimal solutions. This paper and the companion open source computational tools we offer will therefore facilitate the optimization of new CPD systems, without requiring expensive computational resources.

Ultimately, we hope that CFN technology will allow complex CPD problems, mixing optimization of flexible systems and discrete integration (capturing entropic effects and affinity) to be directly tackled.

ACKNOWLEDGEMENTS

This work has been funded by the "Agence Nationale de la Recherche", references ANR 10-BLA-0214 and ANR-12-MONU-0015-03. We thank the Computing Center of Region Midi-Pyrénées (CALMIP, Toulouse, France) and the GenoToul Bioinformatics Platform of INRA-Toulouse for providing computing resources and support. S. Traoré was supported by a grant from the INRA and the Region Midi-Pyrénées.

REFERENCES

Allen,B.D. and Mayo,S.L. (2006) Dramatic performance enhancements for the FASTER optimization algorithm. *Journal of computational chemistry*, **27**, 1071–1075.
Allouche,D. *et al.* (2012) Computational Protein Design as a Cost Function Network Optimization Problem. In, *CP 2012*.
Althaus,E. *et al.* (2002) A combinatorial approach to protein docking with flexible side chains. *J. Comput. Biol.*, **9**, 597–612.
Archontis,G. and Simonson,T. (2005) A residue-pairwise Generalized Born scheme suitable for protein design calculations. *J. Phys. Chem. B*, **109**, 22667–22673.
Arnold,F.H. (2001) Combinatorial and computational challenges for biocatalyst design. *Nature*, **409**, 253–257.
Baldwin,E. *et al.* (1993) The role of backbone flexibility in the accommodation of variants that repack the core of T4 lysozyme. *Science*, **262**, 1715–1718.
Bernstein,F.C. *et al.* (1977) The Protein Data Bank. A computer-based archival file for macromolecular structures. *Eur. J. Biochem.*, **80**, 319–324.
Bordner,A.J. and Abagyan,R.A. (2004) Large-scale prediction of protein geometry and stability changes for arbitrary single point mutations. *Proteins*, **57**, 400–413.
Bowie,J.U. *et al.* (1991) A method to identify protein sequences that fold into a known 3-dimensional structure. *Science*, **253**, 164–170.
Cabon,B. *et al.* (1999) Radio link frequency assignment. *Constraints*, **4**, 79–89.
Case, D. A. *et al.* (2006) AMBER 9 University of California, San Francisco.
Chazelle,B. *et al.* (2004) A Semidefinite Programming Approach to Side Chain Positioning with New Rounding Strategies. *Iinforms J. on Computing*, **16**, 380–392.
Chen,C.-Y. *et al.* (2009) Computational structure-based redesign of enzyme activity. *Proceedings of the National Academy of Sciences*, **106**, 3764–3769.
Cooper,M.C. *et al.* (2010) Soft arc consistency revisited. *Artif. Intell.*, **174**, 449–478.
Desjarlais,J.R. and Handel,T.M. (1995) De novo design of the hydrophobic cores of proteins. *Protein Science*, **4**, 2006–2018.
Desmet,J. *et al.* (2002) Fast and accurate side-chain topology and energy refinement (FASTER) as a new method for protein structure optimization. *Proteins: Structure, Function, and Bioinformatics*, **48**, 31–43.
Desmet,J. *et al.* (1992) The dead-end elimination theorem and its use in protein sidechain positioning. *Nature*, **356**, 539–542.
Gainza,P. *et al.* (2012) Protein Design Using Continuous Rotamers. *PLoS Comput Biol*, **8**, e1002335.
Georgiev,I. *et al.* (2008) The minimized dead-end elimination criterion and its application to protein redesign in a hybrid scoring and search algorithm for computing partition functions over molecular ensembles. *Journal of Computational Chemistry*, **29**, 1527–1542.
Georgiev,Ivelin and Donald,Bruce R. (2007) Dead-End Elimination with Backbone Flexibility. *Bioinformatics*, **23**, i185–i194.
De Givry,S. *et al.* (2006) Mendelsoft: Mendelian error detection in complex pedigree using weighted constraint satisfaction techniques. *8th World Congress on Genetics Applied to Livestock Production, page 2p., Belo Horizonte, Brazil*.
De Givry,Simon *et al.* (2005) Existential arc consistency: getting closer to full arc consistency in weighted CSPs. In, *Proceedings of the 19th international joint conference on Artificial intelligence*, IJCAI'05. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 84–89.
Gordon,D. B. and Mayo,S. L. (1999) Branch-and-terminate: a combinatorial optimization algorithm for protein design. *Structure*, **7**, 1089–1098.
Grunwald,I. *et al.* (2009) Mimicking biopolymers on a molecular scale: nano(bio)technology based on engineered proteins. *Philos Transact A Math Phys Eng Sci*, **367**, 1727–1747.

Hallen,M.A. *et al.* (2013) Dead-end elimination with perturbations (DEEPer): a provable protein design algorithm with continuous sidechain and backbone flexibility. *Proteins*, **81**, 18–39.
Hawkins,G.D. *et al.* (1996) Parametrized models of aqueous free energies of solvation based on pairwise descreening of solute atomic charges from a dielectric medium. *The Journal of Physical Chemistry*, **100**, 19824–19839.
Hellinga,H.W. and Richards,Frederic M. (1991) Construction of new ligand binding sites in proteins of known structure: I. Computer-aided modeling of sites with pre-defined geometry. *Journal of Molecular Biology*, **222**, 763 – 785.
Hong,E.-J. *et al.* (2009) Rotamer optimization for protein design through MAP estimation and problem-size reduction. *J Comput Chem*, **30**, 1923–1945.
Hornak,Viktor *et al.* (2006) Comparison of multiple Amber force fields and development of improved protein backbone parameters. *Proteins*, **65**, 712–725.
Humphris,E.L. and Kortemme,T. (2008) Prediction of protein-protein interface sequence diversity using flexible backbone computational protein design. *Structure*, **16**, 1777–1788.
Janin,J. *et al.* (1978) Conformation of amino acid sidechains in proteins. *J. Mol. Biol.*, **125**, 357–386.
Jaramillo,A. *et al.* (2002) Folding free energy function selects native-like protein sequences in the core but not on the surface. *Proc. Natl. Acad. Sci. USA*, **99**, 13554–13559.
Johnson,E.C. *et al.* (1999) Solution structure and dynamics of a designed hydrophobic core variant of ubiquitin. *Structure*, **7**, 967–976.
Khare,S.D. *et al.* (2012) Computational redesign of a mononuclear zinc metalloenzyme for organophosphate hydrolysis. *Nat. Chem. Biol.*, **8**, 294–300.
Kingsford,C.L. *et al.* (2005) Solving and analyzing side-chain positioning problems using linear and integer programming. *Bioinformatics*, **21**, 1028–1036.
Kirschner,K.N. *et al.* (2008) GLYCAM06: a generalizable biomolecular force field. Carbohydrates. *J Comput Chem*, **29**, 622–655.
Koster,A.M.C.. *et al.* (1999) Solving Frequency Assignment Problems via Tree-Decomposition. *Electronic Notes in Discrete Mathematics*, **3**, 102–105.
Kuhlman,B. *et al.* (2003) Design of a novel globular protein fold with atomic-level accuracy. *Science*, **302**, 1364–1368.
Kuhlman,B. and Baker,D. (2000) Native protein sequences are close to optimal for their structures. *Proc. Natl. Acad. Sci. USA*, **97**, 10383–10388.
Larrosa,J. *et al.* (2005) Existential arc consistency: getting closer to full arc consistency in weighted CSPs. 84–89.
Larrosa,J. and Schiex,T. (2004) Solving weighted CSP by maintaining arc consistency. *Artificial Intelligence*, **159**, 1–26.
Larrosa,Javier *et al.* (1998) Reversible DAC and other improvements for solving Max-CSP. In, *Proceedings of the national conference on artificial intelligence*., pp. 347–352.
Leach,A.R. *et al.* (1998) Exploring the conformational space of protein side chains using dead-end elimination and the A* algorithm. *Proteins Structure Function and Genetics*, **33**, 227–239.
Leaver-Fay,A. *et al.* (2005) An adaptive dynamic programming algorithm for the side chain placement problem. *Pac Symp Biocomput*, 16–27.
Lecoutre,C. *et al.* (2009) Reasoning from last conflict (s) in constraint programming. *Artificial Intelligence*, **173**, 1592–1614.
Lim,W.A. *et al.* (1994) The crystal structure of a mutant protein with altered but improved hydrophobic core packing. *Proc. Natl. Acad. Sci. U.S.A.*, **91**, 423–427.
Lippow,S. M. *et al.* (2007) Computational design of antibody affinity improvement beyond in vitro maturation. *Nature Biotech.*, **25**, 1171–1176.
Lovell,S.C. *et al.* (2000) The penultimate rotamer library. *Proteins*, **40**, 389–408.
Nestl,B.M. *et al.* (2011) Recent progress in industrial biocatalysis. *Curr Opin Chem Biol*, **15**, 187–193.
Ogata,K. *et al.* (2003) Automatic Sequence Design of MHC Class-I Binding Peptides Impairing CD8+ T Cell Recognition. *J. Biol. Chem.*, **278**.
Pabo,C. (1983) Molecular technology: designing proteins and peptides. *Nature*, **301**, 200–200.
Pierce,N.A. and Winfree,E. (2002) Protein Design is NP-hard. *Protein Engineering*, **15**, 779–782.
Pleiss,J. (2011) Protein design in metabolic engineering and synthetic biology. *Curr. Opin. Biotechnol.*, **22**, 611–617.
Raha,K. *et al.* (2000) Prediction of amino acid sequence from structure. *Prot. Sci.*, **9**, 1106–1119.
Saunders,C.T. and Baker,D. (2005) Recapitulation of protein family divergence using flexible backbone protein design. *J. Mol. Biol.*, **346**, 631–644.
Schiex,T. (2000) Arc consistency for soft constraints. *Principles and Practice of Constraint Programming–CP 2000*, 411–425.
Schiex,T. *et al.* (1995) Valued constraint satisfaction problems: Hard and easy problems. *International Joint Conference on Artificial Intelligence*, **14**, 631–639.
Voigt,C.A. *et al.* (2000) Trading accuracy for speed: A quantitative comparison of search algorithms in protein sequence design. *Journal of Molecular Biology*, **299**, 789–803.
Wallace,R.J. (1996) Enhancements of branch and bound methods for the maximal constraint satisfaction problem. In, *Enhancements of branch and bound methods for the maximal constraint satisfaction problem*., pp. 188–195.
Wang,J. *et al.* (2006) Automatic atom type and bond type perception in molecular mechanical calculations. *J. Molec. Graph. Model.*, **25**, 247260.
Wang,J. *et al.* (2004) Development and testing of a general AMBER force field. *J. Comp. Chem.*, **25**, 1157–1174.
Wernisch,L. *et al.* (2000) Automatic protein design with all atom force fields by exact and heuristic optimization. *J. Mol. Biol.*, **301**, 713–736.
Zytnicki,M. *et al.* (2008) DARN! A weighted constraint solver for RNA motif localization. *Constraints*, **13**, 91–109.

SUPPLEMENTARY INFORMATION

A New Framework for Computational Protein Design through Cost Function Network Optimization

Seydou Traoré [1-3#], David Allouche[4#], Isabelle André[1-3], Simon de Givry[4], George Katsirelos[4], Thomas Schiex[4*], Sophie Barbe[1-3*]
[1]Université de Toulouse;INSA,UPS,INP; LISBP, 135 Avenue de Rangueil, F-31077 Toulouse, France
[2]INRA, UMR792, Ingénierie des Systèmes Biologiques et des Procédés, F-31400 Toulouse, France
[3]CNRS, UMR5504, F-31400 Toulouse, France
[4]Unité de Biométrie et Intelligence Artificielle, UR 875, INRA, F-31320 Castanet Tolosan, France

Supplementary data: Preparation of benchmark set

The macromolecular diversity was generated on the basis of 35 high-quality crystal structures of proteins and complexes taken from the PDB (with a resolution of 2.9 Å or lower) which include proteins of various sizes (from 54 to 274 residues) and from different structural classes and folds, according to the SCOP classification (Table S2). Next, in order to explore sequence-conformation spaces relevant to the type of protein design targeted, a strategy was set up to define the set of mutable residues, the set of amino acid types allowed at each mutable position and finally the set of residues to repack. Hydrophobic amino acids (V,L,I,F,M,Y,W) were allowed at the core, hydrophilic amino acids (S,T,D,N,E,Q,H,K,R) at the surface and a combination of both sets in the boundary region. The alanine amino-acid was permitted at all three layers. The amino acid present at mutable positions in the wild-type protein was always allowed. For the redesign of protein-protein and protein-ligand interfaces, all amino acid types, except proline, were considered at each mutable position. The set of mutable residues was derived from experimental and computational literature (see references in Table S1) or defined according to the design target: (i) all residues of the protein core were allowed to mutate for the design of proteins (either free or bound to a cofactor) ; (ii) the residues that have their Cβ within a distance up to 8 Å of the center of mass of the ligand were set to mutable for the redesign of protein-ligand interactions; (iii) the residues from one protein chain that have their Cβ within a distance up to 8 Å of Cβ from any residue belonging to the other protein chain were set to mutable for the redesign of protein-protein interfaces. In addition to mutable residues, a set of residues enabling repacking (*i.e.*, flexible residues) was also defined depending on the design target: (i) all non-mutable residues of the core and the boundary regions were considered flexible for the protein design instances; (ii) non-mutable residues of the core and boundary layers which have their Cβ within a distance of 12 Å from the center of mass of the ensemble of the mutable residues of the designed chain were set to repackable for the redesign of protein-ligand and protein-protein interfaces. The protein backbone, the side-chains of non-variable residues (*i.e.*, residues not set to mutable or repackable) and the non-amino acid type partners were kept fixed.

This strategy seeks the best compromise between the size of each of these three sets that together determine the combinatorial complexity of the search space. The selection applied on these sets is highly dependent on the targeted design which is here the enhancement of the stability of proteins, holoproteins, as well as protein-ligand and protein-protein interfaces. Protein stability is associated with the ability of proteins to fold properly under given conditions and to maintain this folding.

Invariably, water-soluble proteins fold into structures that extensively bury hydrophobic residues into the core of the structure while simultaneously exposing polar amino acid side-chains at the surface to form favorable electrostatic interactions with aqueous solvent (Kauzmann, 1959). Nevertheless, the burial of hydrophilic residues within protein structures is not always thermodynamically unfavorable. Indeed, polar side-chains located inside proteins as well as at the interface between two proteins or a protein and a small molecule can play a structural role in the stabilization of the folded state and the intermolecular complexes (Bolon and Mayo, 2001; Bolon et al., 2003), in addition to their significant contribution to the specificity definition. However, the benefit of this burial which depends on molecular interactions and desolvation, appears to be extremely difficult to predict using current CPD energy functions. Therefore, in order to better reflect the distribution of amino acids types into well-folded native protein structures and to account for the limited capability of CPD energy functions to predict the effect of burying polar groups, we have adopted a restrictive approach widely used in CPD (Bolon et al., 2003). This approach confines protein core positions to hydrophobic amino acids, surface positions to hydrophilic amino acids and in contrast allows both hydrophobic and hydrophilic amino acids at boundary as well as protein-protein and protein-ligand interface regions. An automated way to stratify protein structures into three layers (core, boundary and surface) is then needed. For this purpose, previous work used either the exposed surface area of residues (Koga et al., 2012) or the Cα distances to the nearest exposed surface area residue along the Cα-Cβ vector (Dahiyat and Mayo, 1997). Herein, we implemented a residue classification method which aims to determine more precisely the burial of residues on the basis of the solvation radii of residues (Fig. S1).

In addition to maximizing the integration of structure-based knowledge into CPD, the approach results in a reduction of the combinatorial complexity compared to the consideration of all amino acid types at each mutable position. This reduction on the set of amino acid types may also allow us to handle more mutable positions or/and flexible residues.

Table S1: Benchmark set

PDB	*N*	*N-variable (mutable, flexible)*	*N-variable core*[a] *(mutable, flexible)*	*N-variable boundary*[b] *(mutable, flexible)*	*N-variable surface*[c] *(mutable, flexible)*	*N-rotamer range*[d] *min-max*	*Sequence-Conformation Space Size*	*Partners (Cofactor, Ligand, Protein)*	*References*
1MJC	69	28 (3, 25)	11 (2, 9)	17 (1, 16)	0 (0, 0)	3-182	4.36e+26	-	(Hillier *et al.*, 1998)
1CSP	67	30 (6, 24)	11 (0, 11)	16 (3, 13)	3 (3, 0)	3-182	5.02e+30	-	(Perl and Schmid, 2001)
1BK2	57	24 (11, 13)	8 (6, 2)	16 (5, 11)	0 (0, 0)	3-182	1.18e+32	-	(Pokala and Handel, 2005)
1SHG	57	28 (9, 19)	10 (8, 2)	18 (1, 17)	0 (0, 0)	3-182	2.13e+32	-	(Ventura *et al.*, 2002)
1CSK	58	30 (9, 21)	9 (9, 0)	21 (0, 21)	0 (0, 0)	3-49	4.09e+32	-	(Busch *et al.*, 2008)
1SHF	59	30 (9, 21)	9 (9, 0)	21(0, 21)	0 (0, 0)	3-56	1.05e+34	-	(Northey *et al.*, 2002)
1FYN	62	23 (13, 10)	4 (1, 3)	13(8, 5)	6 (4, 0)	3-186	5.04e+36	Peptide-ligand	(Schweiker *et al.*, 2007)
1PIN	154	28 (11, 17)	13 (3, 10)	13(6, 7)	2 (2, 0)	3-194	5.32e+39	Peptide-ligand, SO_4^{2-}	(Kraemer-Pecore *et al.*, 2003)
1NXB	62	34 (9, 25)	9 (9, 0)	25 (0, 25)	0 (0, 0)	3-56	2.61e+41	SO_4^{2-}	(Z. Liu *et al.*, 2001)
1TEN	90	39 (11, 28)	11 (11, 0)	28 (0, 28)	0 (0, 0)	3-66	6.17e+43	-	(Dantas *et al.*, 2003)
1POH	85	46 (3, 43)	14 (0, 14)	32 (3, 29)	0 (0, 0)	3-182	8.02e+43	SO_4^{2-}	(Pokala and Handel, 2005)
2DRI	271	37(10,27)	20 (8, 12)	17(2, 15)	0 (0, 0)	3-186	1.16e+47	Ribose	(Boas and Harbury, 2008)
1FNA	91	38 (15, 23)	15 (15, 0)	23 (0, 23)	0 (0, 0)	3-48	3.02e+47	-	(Hamill *et al.*, 2000)
1UBI	76	40 (14, 26)	18 (13, 5)	22 (1, 21)	0 (0, 0)	3-182	2.43e+49	-	(Ermolenko *et al.*, 2003)
1C9O	66	43 (11, 32)	14 (1, 13)	25 (6, 19)	4 (4, 0)	3-182	3.77e+49	Na^+	(Perl *et al.*, 2000; Perl and Schmid, 2001)
1CTF	68	39 (20, 19)	20 (20, 0)	19 (0, 19)	0 (0, 0)	3-56	3.95e+51	SO_4^{2-}	(Busch *et al.*, 2008)
2PCY	99	46 (16, 30)	16 (16, 0)	30 (0, 30)	0 (0, 0)	3-56	2.34e+52	-	(Gordon *et al.*, 2003)
1DKT	71	46 (14, 32)	14 (5, 9)	32 (9, 23)	0 (0, 0)	3-190	3.94e+58	-	(Seeliger *et al.*, 2003)
2TRX	108	61 (6, 55)	21 (3,18)	40 (3, 37)	0 (0, 0)	3-186	9.02e+59	Cu^{2+}	(D. Bolon *et al.*, 2003)
1PGB	56	31 (28, 3)	6 (5, 1)	13 (11, 2)	12 (12, 0)	5-182	5.10e+61	-	(Lassila *et al.*, 2002)
1CM1	143	42(18,24)	23 (10, 13)	16(7, 9)	3 (1, 0)	3-186	3.73e+63	Calmodulin-dependant protein Kinase II alpha chain, Ca^{2+}	(Yosef *et al.*, 2009)
1BRS	108	44 (19, 25)	13 (2, 11)	23 (10, 13)	8 (7, 0)	3-194	1.67e+64	Barstar protein chain-D	(Pokala and Handel, 2005)
1ENH	54	36 (27, 9)	7 (5, 2)	11 (4, 7)	18 (18, 0)	5-182	6.65e+64	-	(Marshall *et al.*, 2002)
1CDL	142	40 (21, 19)	14 (8, 6)	22(12, 10)	4 (1, 0)	3-186	5.68e+65	Calmodulin-dependant protein Kinase II alpha chain, Ca^{2+}	(Fromer and Yanover, 2009)
1LZ1	130	59 (22, 37)	22 (22, 0)	37 (0, 37)	0 (0, 0)	3-57	1.04e+72	-	(Baldwin *et al.*, 1993)
2CI2	65	51 (27, 24)	16 (8, 8)	29 (13, 16)	6 (6, 0)	3-183	7.26e+75	-	(Fersht, 1995)
1GVP	87	52 (28, 24)	17 (9, 8)	28 (12, 16)	7 (7, 0)	3-182	1.51e+78	-	(Sandberg and Terwilliger, 1993),
1RIS	97	56 (27, 29)	17 (14, 3)	37 (11, 26)	2 (2, 0)	3-182	1.23e+80	-	(Dantas *et al.*, 2003)
2RN2	155	69 (27, 42)	27 (27, 0)	42 (0, 42)	0 (0, 0)	3-66	3.68e+80	-	(Pokala and Handel, 2005)
1CSE	274	97 (4, 93)	30 (0, 30)	65 (2, 63)	2 (2, 0)	3-183	8.35e+82	Ca^{2+}	(Yi *et al.*, 2003)
1HNG	175	85 (13, 72)	19 (4, 15)	63 (6, 57)	3 (3, 0)	3-182	3.70e+88	-	(Lorch *et al.*, 1999; Poso *et al.*, 2000)
3CHY	128	74 (31, 43)	31 (31, 0)	43 (0, 43)	0 (0, 0)	3-66	2.36e+92	SO_4^{2-}	(Pokala and Handel, 2005)
1L63	162	83 (21, 62)	21 (15, 6)	61 (5, 56)	1 (1, 0)	3-182	2.17e+94	Cl^-	(Baldwin *et al.*, 1993; Blaber *et al.*, 1994)
3HHR	185	115 (45, 70)	46 (29, 17)	69 (16, 53)	0 (0, 0)	3-186	2.98e+171	-	(Filikov *et al.*, 2002)
1STN	136	120 (119, 1)	27 (27, 0)	52 (51, 1)	41 (41, 0)	7-190	2.00e+249	-	(Schwehm *et al.*, 1998; Holder *et al.*, 2001)

For each instance: system reference PDB id (PDB), number of residues (N), number of designable residues of the whole system (N-variable), number of designable residues (N-variable) at each layer of the mutable chain ([a]core, [b]boundary, [c]surface), [d]minimal and maximal number of rotamers per variable position, and references.

Table S2: Description of input models

System name	*PDB*	*R(Å)*	*N*	*Class*	*Fold*
E.coli Cold-shock protein A	1MJC	2.00	69	All beta	OB-fold
B. subtilis Cold-shock protein B	1CSP	2.45	67	All beta	OB-fold
A-Spectrin SH3 domain mutant	1BK2	2.01	57	All beta	SH3-like barrel
A-Spectrin SH3 domain	1SHG	1.80	57	All beta	SH3-like barrel
C-SRC SH3 domain	1CSK	2.50	58	All beta	SH3-like barrel
Fyn Tyrosine Kinase SH3 domain	1SHF	1.90	59	All beta	SH3-like barrel
Phosphotransferase Fyn	1FYN	2.30	62	All beta	SH3-like barrel
Peptidyl-prolyl cis-trans isomerase	1PIN	1.35	154	All beta; (a+b)	WW domain-like; FKBP-like
Neurotoxin B	1NXB	1.38	62	Small proteins	Snake toxin-like
Tenascin fibronectin type III domain	1TEN	1.80	90	All beta	Immunoglobulin-like beta-sandwich
Histidine phosphorelay protein	1POH	2.00	85	(a+b)	HPr-like
D-ribose-binding protein	2DRI	1.60	271	(a/b)	Periplasmic binding protein-like I
Fibronectin Cell-adhesion module type III	1FNA	1.80	91	All beta	Immunoglobulin-like beta-sandwich
Ubiquitin	1UBI	1.80	76	(a+b)	beta-Grasp (ubiquitin-like)
B caldolyticus Cold-shock protein B	1C9O	1.17	66	All beta	OB-fold
Ribosomal protein L7 /L12 C-Ter domain	1CTF	1.70	68	(a+b)	ClpS-like
Apoplastocyanin	2PCY	1.80	99	All beta	Cupredoxin-like
Human cyclin dependent kinase subunit	1DKT	2.90	71	(a+b)	Cell cycle regulatory proteins
Thioredoxine	2TRX	1.68	108	(a/b)	Thioredoxin fold
Protein G domain B1	1PGB	1.92	56	(a+b)	beta-Grasp (ubiquitin-like)
Calmodulin	1CM1	2.00	143	All alpha	EF Hand-like
Barnase	1BRS	2.00	108	(a+b)	Microbial ribonucleases
Engrailed homeodomain	1ENH	2.10	54	All alpha	DNA/RNA-binding 3-helical bundle
Calmodulin	1CDL	2.00	142	All alpha	EF Hand-like
Human Lyzozyme	1LZ1	1.50	130	(a+b)	Lysozyme-like
Chymotrypsin inhibitor 2	2CI2	2.00	65	(a+b)	CI-2 family of serine protease inhibitors
Gene V DNA binding protein	1GVP	1.60	87	All beta	OB-fold
Ribosomal protein S6	1RIS	2.00	97	(a+b)	Ferredoxin-like
Ribonuclease H	2RN2	1.48	155	(a/b)	Ribonuclease H-like motif
Subtilisin Carlsberg	1CSE	1.20	274	(a/b)	Subtilisin-like
Cell Adhesion Molecule CD2	1HNG	2.80	175	All beta	Immunoglobulin-like beta-sandwich
Signal transduction protein CheY	3CHY	1.66	128	(a/b)	Flavodoxin-like
Phage T4 lysozyme	1L63	1.75	162	(a+b)	Lysozyme-like
Human growth hormone	3HHR	2.80	185	All alpha	4-helical cytokines
Staphylococcal nuclease	1STN	1.70	136	All beta	OB-fold

For each instance: system name, reference PDB id (PDB), crystallographic resolution, number of residues (N), SCOP structural classification (Class and Fold).

Table S3: Solving of the GMEC using DEE/A* without pre-processing (I) and with pre-processing (II) and DEE/CFN without pre-processing (I).

PDB	Space size			Times(s)						
	Input[a]	Output[b] (I)	Output[b] (II)	DEE (I)	DEE (II)	CFN (I)	A*(I)	A* (II)	DEE/A*(I)	DEE/A*(II)
1MJC	4.36e+26	1.00	1.00	4.57	5.01	-	-	-	4.57	5.01
1CSP	5.02e+30	4.44e+05	4.44e+05	186.00	103.00	0.01	13.70	7.58	200.00	111.00
1BK2	1.18e+32	1.94e+06	4.42e+06	87.80	598.00	0.01	5.42	5.64	93.20	603.00
1SHG	2.13e+32	6.21e+09	6.21e+09	77.30	79.80	0.02	60.50	61.20	138.00	141.00
1CSK	4.09e+32	9.58e+06	9.58e+06	35.80	57.90	0.00	5.96	7.52	41.70	65.40
1SHF	1.05e+34	3.51e+07	3.51e+07	38.40	131.00	0.01	5.89	93.30	44.30	224.00
1FYN	5.04e+36	8.00	8.00	613.00	658.00	0.00	8.48	8.69	622.00	667.00
1PIN	5.32e+39	1.02e+14	6.22e+05	8.48e+03	3.65e+03	0.00	1.06e+03	1.39e+03	9.54e+03	5.04e+03
1NXB	2.61e+41	1.00	1.00	11.10	7.95	-	-	-	11.10	7.95
1TEN	6.17e+43	2.58e+07	2.58e+07	94.50	98.10	0.01	18.60	21.20	1130	119.00
1POH	8.02e+43	768.00	768.00	61.10	67.10	0.00	16.80	18.10	77.90	85.20
2DRI	1.16e+47	1.78e+21	1.78e+21	7.89e+04	7.87e+04	2.1	T	T	T	T
1FNA	3.02e+47	3.77e+15	3.77e+15	973.00	882.00	0.05	2.33e+03	2.34e+03	3.31e+03	3.22e+03
1UBI	2.43e+49	1.51e+20	1.51e+20	3.20e+03	3.64e+03	0.38	T	T	T	T
1C9O	3.77e+49	4.42e+07	8.00	2.28e+03	636.00	0.00	27.00	10.80	2.31e+03	647.00
1CTF	3.95e+51	6.68e+20	6.68e+20	1.88e+03	1.89e+03	0.18	T	T	T	T
2PCY	2.34e+52	2.11e+15	2.11e+15	675.00	795.00	0.03	1.40e+03	1.40e+03	2.08e+03	2.20e+03
1DKT	3.94e+58	2.06e+13	5.50e+13	1.41e+03	1.00e+03	0.03	4.01e+03	1.04e+04	5.42e+03	1.14e+04
2TRX	9.02e+59	8.49e+07	8.49e+07	426.00	275.00	0.02	61.30	51.70	487.00	326.00
1PGB	5.10e+61	T	T	T	T	Not reached	Not reached	Not reached	T	T
1CM1	3.73e+63	2.05e+19	2.05e+19	4.28e+04	4.18e+04	0.07	T	T	T	T
1BRS	1.67e+64	2.34e+29	2.34e+29	8.10e+04	7.65e+04	5.85	T	T	T	T
1ENH	6.65e+64	T	T	T	T	Not reached	Not reached	Not reached	T	T
1CDL	5.68e+65	1.45e+29	5.94e+26	1.94e+05	1.04e+05	0.87	T	T	T	T
1LZ1	1.04e+72	9.77e+22	6.91e+21	3.77e+03	3.28e+03	0.18	T	T	T	T
2CI2	7.26e+75	T	T	T	T	Not reached	Not reached	Not reached	T	T
1GVP	1.51e+78	8.49e+28	8.49e+28	2.31e+04	2.27e+04	0.32	T	T	T	T
1RIS	1.23e+80	6.40e+39	6.40e+39	6.90e+04	6.89e+04	2.77	T	T	T	T
2RN2	3.68e+80	1.32e+22	1.55e+19	4.76e+03	3.78e+03	0.12	M	T	M	T
1CSE	8.35e+82	1.00	1.00	367.00	282.00	-	-	-	367.00	282.00
1HNG	3.70e+88	6.91e+03	6.91e+03	5.21e+03	4.60e+03	0.01	377.00	233.00	5.59e+03	4.83e+03
3CHY	2.36e+92	9.66e+36	7.87e+35	1.85e+04	1.78e+04	8.51	T	T	T	T
1L63	2.17e+94	6.92e+23	4.18e+21	5.03e+03	4.18e+03	0.08	M	T	M	T
3HHR	2.98e+171	T	T	T	T	Not reached	Not reached	Not reached	T	T
1STN	2.00e+249	T	T	T	T	Not reached	Not reached	Not reached	T	T

For each instance are indicated the [a]input and [b]output (after DEE) sequence-conformation space size. A 'M' indicates an exceeded memory size (128G) and a 'T' indicates an exceeded computation time (100h). '-' indicates that the GMEC was identified during the DEE step.

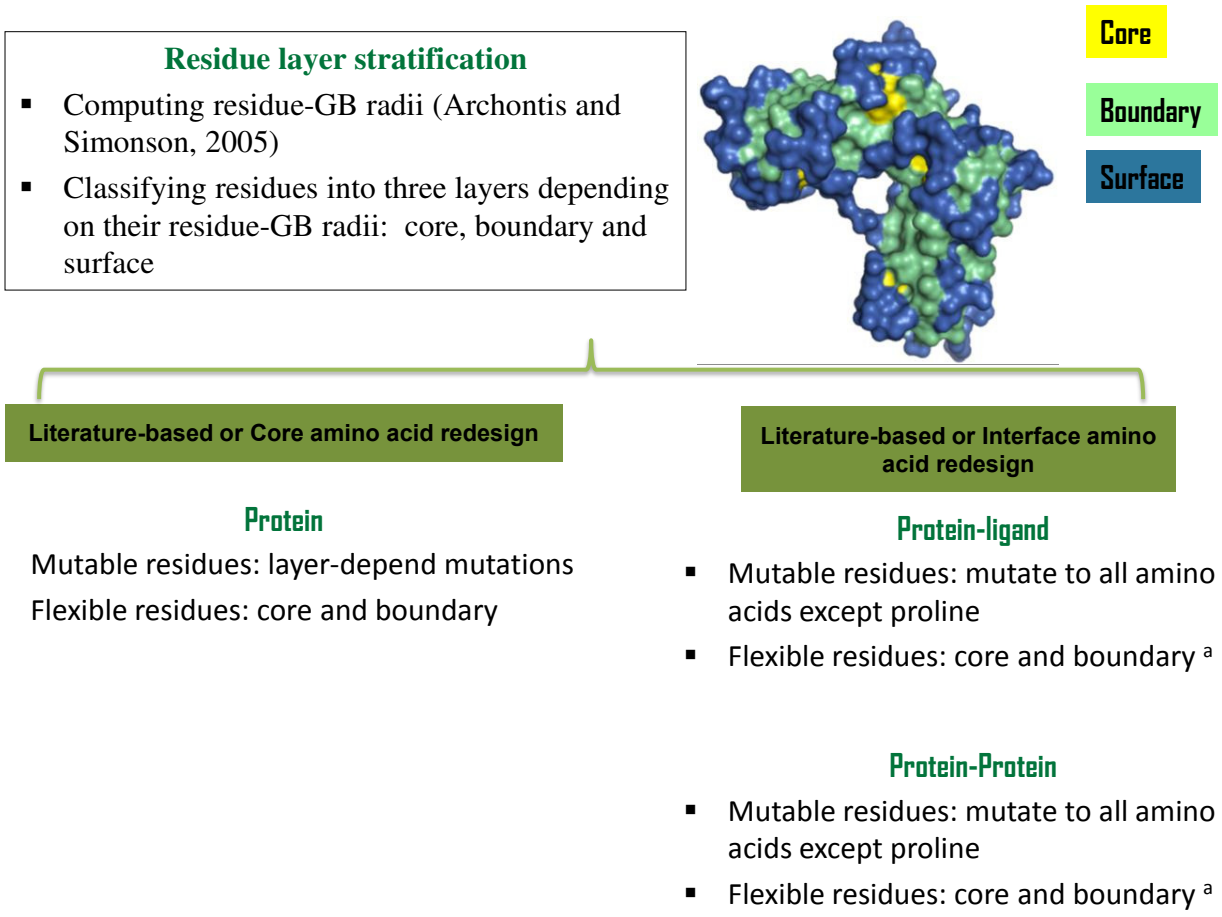


Fig. S1 Definition of Sequence-Conformation Spaces. [a]The non-mutable residues of the core and boundary layers which have their Cβ within a distance of 12 Å from the center of mass of the ensemble of the mutable residues were set to repackable for the redesign of protein-ligand and protein-protein interfaces.

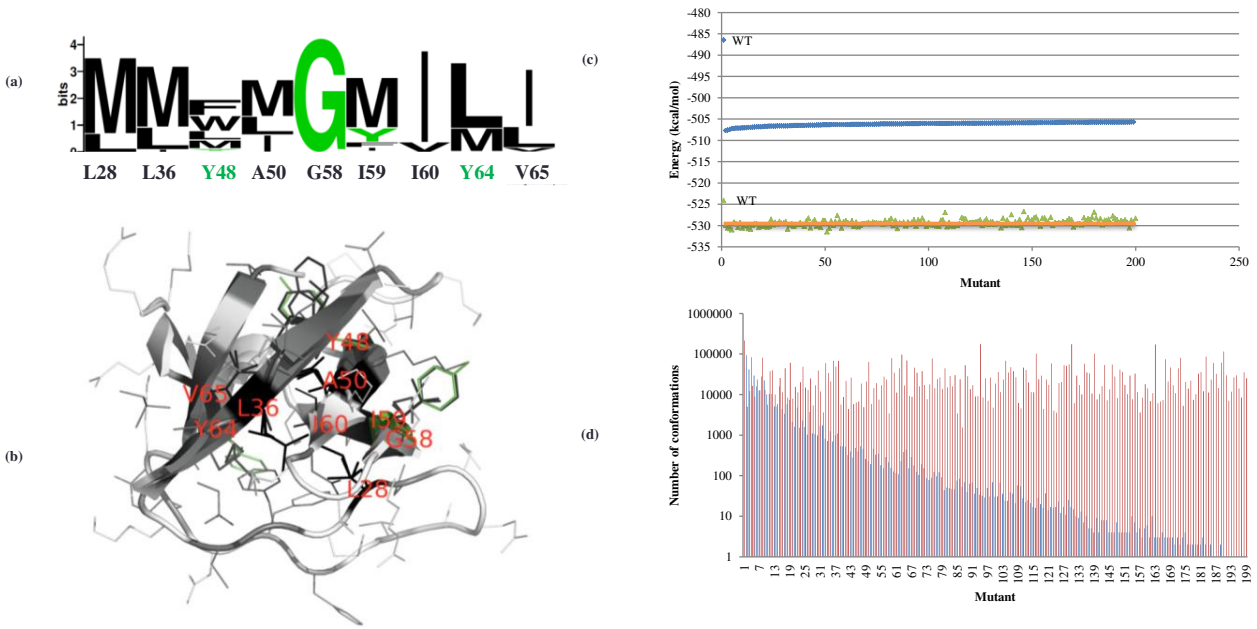


Fig. S1 Score refinement and conformational analysis for 1CSK. (a) Comparison of near-optimal sequences generated by CFN-based approach, using *Weblogo* (Crooks *et al.*, 2004). For each design position (horizontal axis), the total height quantifies the conservation (information content) at that position, where taller positions are more conserved; the relative height of each amino acid denotes its frequency in the design. Polar amino acids and glycine (G,S,T,Y,C) are green, amide purple (N,Q), basic (K,R,H) blue, acidic (D,E) red and hydrophobic (A,V,L,I,P,W,F,M) are black. The wild-type amino acids are given underneath the horizontal axis**.** The rate of mutations of the sequence ensemble, generated by CFN-based approach ranges from 55.56 to 88.89% for 1CSK **(b)** Superimposition of minimized structures of all unique sequences (best conformation per sequence) using *pymol* (Schrödinger, 2010). Mutable residues are colored as in (a), flexible residues are shown in grey**. (c)** Energy refinement: the scores of structures of all unique sequences are shown in blue, and those of the minimized corresponding structures in green. The wild-type model is added to the two sets. The horizontal orange line indicates the subsequent cutoff E'_{cut} of 2 kcal.mol^{-1}, from the scores of minimized structures**. (d)** Distribution of the number of conformations: comparison between the number of conformations generated for each sequence, during the enumeration of the sub-optimal set (blue) and during the repacking of refined structures (red). A cutoff of 0.2 kcal.mol^{-1} (with respect to the sequence-specific minimum score) was used for repacking enumeration (red).

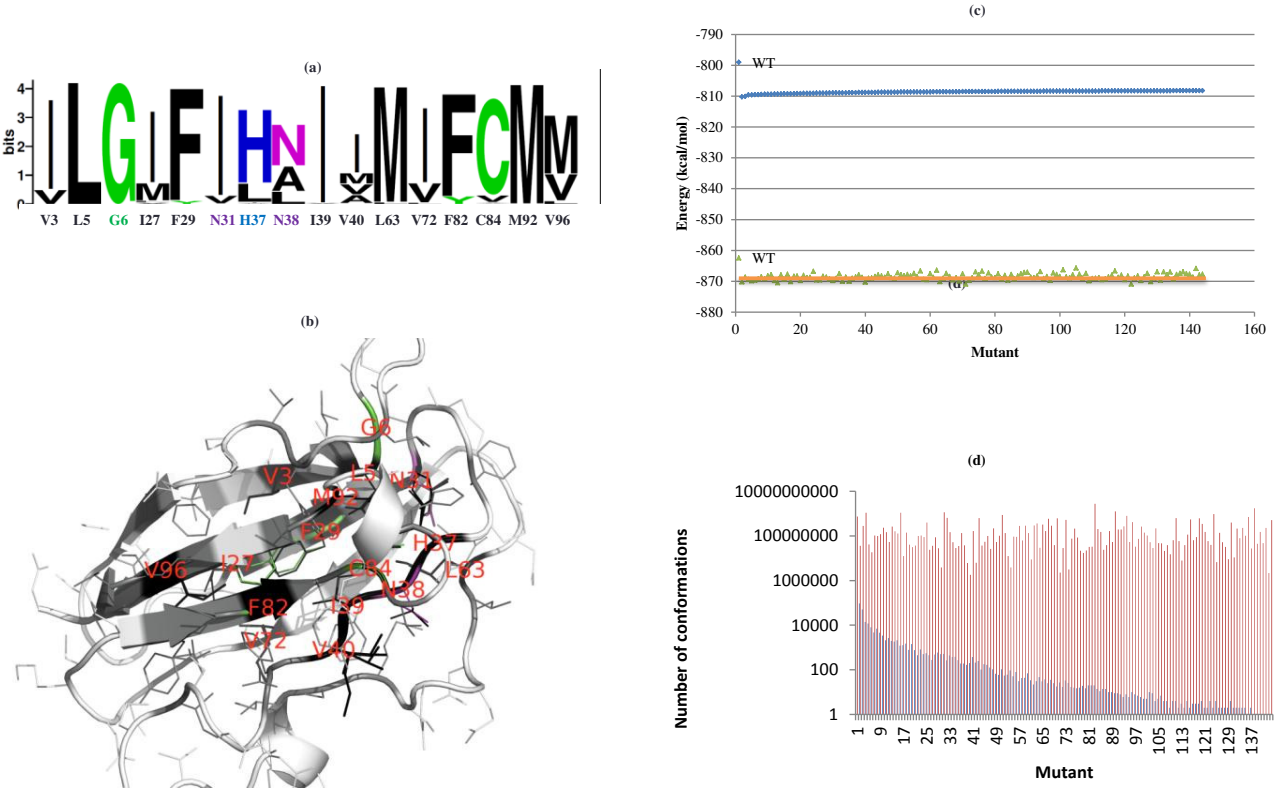


Fig. S3 Score refinement and conformational analysis for 2PCY. (a) Comparison of near-optimal sequences generated by CFN-based approach, using *Weblogo* (Crooks *et al.*, 2004). For each design position (horizontal axis), the total height quantifies the conservation (information content) at that position, where taller positions are more conserved; the relative height of each amino acid denotes its frequency in the design. Polar amino acids and glycine (G,S,T,Y,C) are green, amide purple (N,Q), basic (K,R,H) blue, acidic (D,E) red and hydrophobic (A,V,L,I,P,W,F,M) are black. The wild-type amino acids are given underneath the horizontal axis**.** The rate of mutations of the sequence ensemble, generated by CFN-based approach ranges from 25 to 56.25% for 2PCY**. (b)** Superimposition of minimized structures of all unique sequences (best conformation per sequence) using *pymol* (Schrödinger, 2010). Mutable residues are colored as in (a), flexible residues are shown in grey**. (c)** Energy refinement: the scores of structures of all unique sequences are shown in blue, and those of the minimized corresponding structures in green. The wild-type model is added to the two sets. The horizontal orange line indicates the subsequent cutoff E'_{cut} of 2 kcal.mol^{-1}, from the scores of minimized structures**. (d)** Distribution of the number of conformations: comparison between the number of conformations generated for each sequence, during the enumeration of the sub-optimal set (blue) and during the repacking of refined structures (red). A cutoff of 0.2 kcal.mol^{-1} (with respect to the sequence-specific minimum score) was used for repacking enumeration (red).

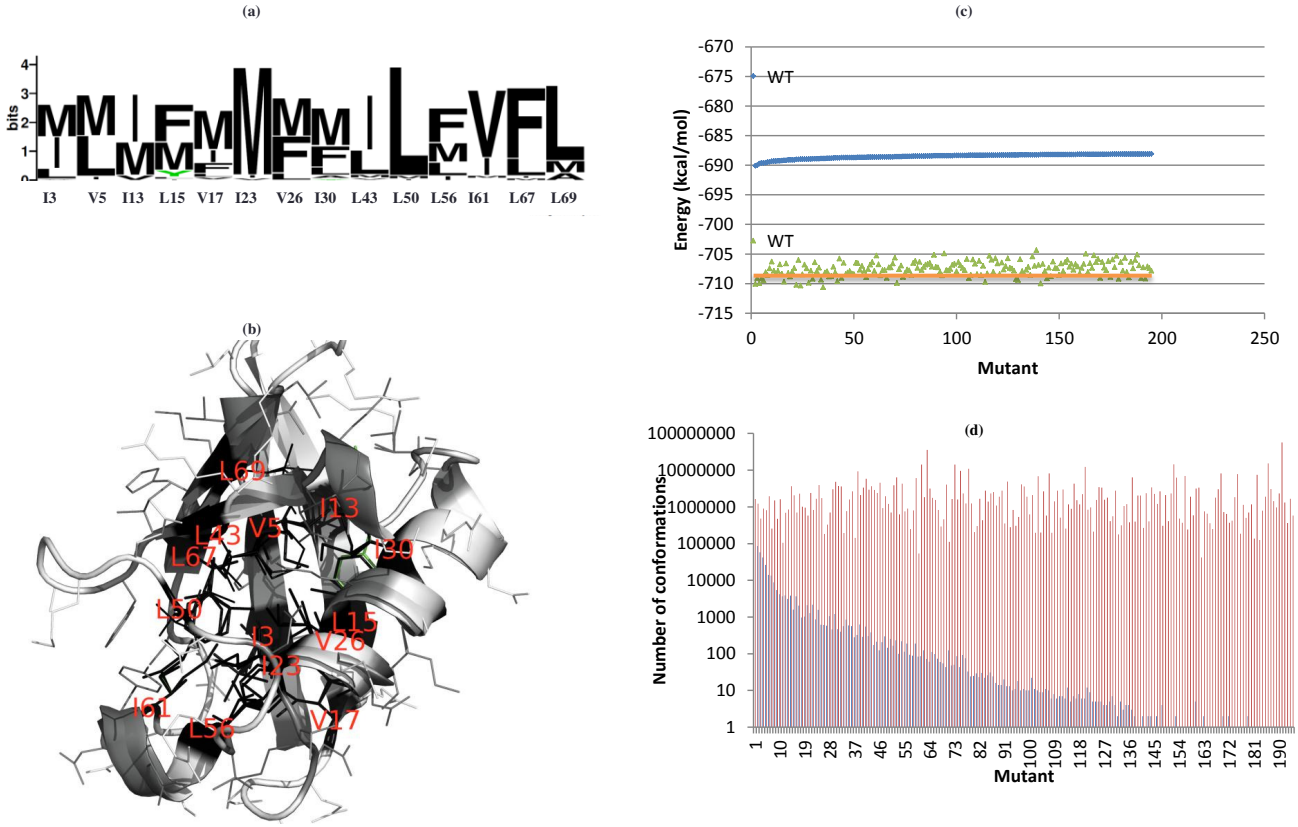


Fig. S4 Score refinement and conformational analysis for 1UBI. (a) Comparison of near-optimal sequences generated by CFN-based approach, using *Weblogo* (Crooks *et al.*, 2004). For each design position (horizontalaxis), the total height quantifies the conservation (information content) at that position, where taller positions are more conserved; the relative height of each amino acid denotes its frequency in the design. Polar amino acids and glycine (G,S,T,Y,C) are green, amide purple (N,Q), basic (K,R,H) blue, acidic (D,E) red and hydrophobic (A,V,L,I,P,W,F,M) are black. The wild-type amino acids are given underneath the horizontal axis. The rate of mutations of the sequence ensemble, generated by CFN-based approach ranges from 64.29 to 85.71% for 1UBI. **(b)** Superimposition of minimized structures of all unique sequences (best conformation per sequence) using *pymol* (Schrödinger, 2010). Mutable residues are colored as in (a), flexible residues are shown in grey. **(c)** Energy refinement: the scores of structures of all unique sequences are shown in blue, and those of the minimized corresponding structures in green. The wild-type model is added to the two sets. The horizontal orange line indicates the subsequent cutoff E'_{cut} of 2 kcal.mol^{-1}, from the scores of minimized structures. **(d)** Distribution of the number of conformations: comparison between the number of conformations generated for each sequence, during the enumeration of the sub-optimal set (blue) and during the repacking of refined structures (red). A cutoff of 0.2 kcal.mol^{-1} (with respect to the sequence-specific minimum score) was used for repacking enumeration (red).

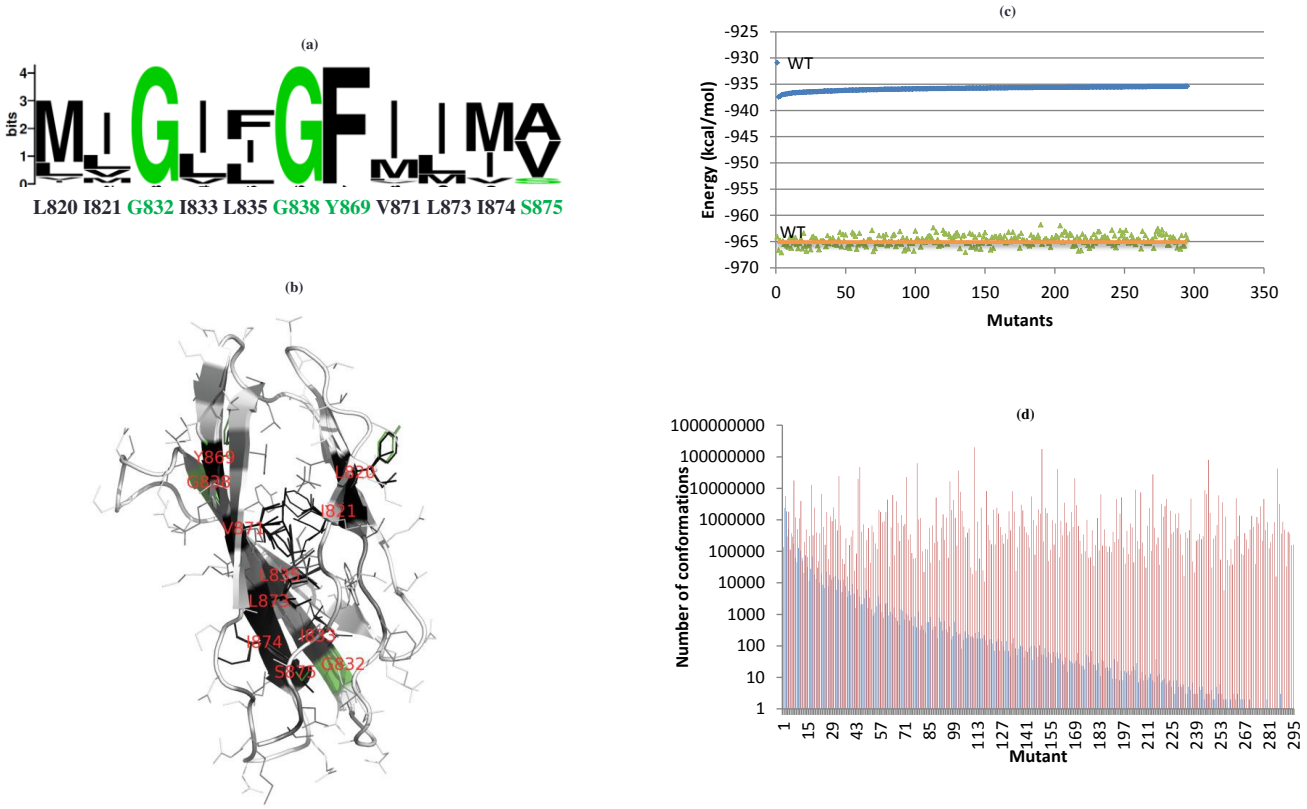


Fig. S5 Score refinement and conformational analysis for 1TEN. **(a)** Comparison of near-optimal sequences generated by CFN-based approach, using *Weblogo* (Crooks *et al.*, 2004). For each design position (horizontal axis), the total height quantifies the conservation (information content) at that position, where taller positions are more conserved; the relative height of each amino acid denotes its frequency in the design. Polar amino acids and glycine (G,S,T,Y,C) are green, amide purple (N,Q), basic (K,R,H) blue, acidic (D,E) red and hydrophobic (A,V,L,I,P,W,F,M) are black. The wild-type amino acids are given underneath the horizontal axis. The rate of mutations of the sequence ensemble, generated by CFN-based approach ranges from 36.36 to 81.82% for 1TEN. **(b)** Superimposition of minimized structures of all unique sequences (best conformation per sequence) using *pymol* (Schrödinger, 2010). Mutable residues are colored as in (a), flexible residues are shown in grey. **(c)** Energy refinement: the scores of structures of all unique sequences are shown in blue, and those of the minimized corresponding structures in green. The wild-type model is added to the two sets. The horizontal orange line indicates the subsequent cutoff E'_{cut} of 2 kcal.mol^{-1}, from the scores of minimized structures. **(d)** Distribution of the number of conformations: comparison between the number of conformations generated for each sequence, during the enumeration of the sub-optimal set (blue) and during the repacking of refined structures (red). A cutoff of 0.2 kcal.mol^{-1} (with respect to the sequence-specific minimum score) was used for repacking enumeration (red).

References

Baldwin,E. *et al.* (1993) The role of backbone flexibility in the accommodation of variants that repack the core of T4 lysozyme. *Science*, **262**, 1715–1718.

Blaber,M. *et al.* (1994) Determination of alpha-Helix Propensity within the Context of a Folded Protein: Sites 44 and 131 in Bacteriophage T4 Lysozyme. *Journal of Molecular Biology*, **235**, 600 – 624.

Boas,F.E. and Harbury,P.B. (2008) Design of protein-ligand binding based on the molecular mechanics energy model. *J. Mol. Biol.*, **380**, 415–424.

Bolon,D N and Mayo,S L (2001) Polar residues in the protein core of Escherichia coli thioredoxin are important for fold specificity. *Biochemistry*, **40**, 10047–10053.

Bolon,D. *et al.* (2003) Enzyme-like proteins by computational design. *J.\ Mol.\ Biol.*, **329**, 611–622.

Bolon,Daniel N *et al.* (2003) Prudent modeling of core polar residues in computational protein design. *J. Mol. Biol.*, **329**, 611–622.

Busch,M.S. am *et al.* (2008) Computational protein design: software implementation, parameter optimization, and performance of a simple model. *J. Comp. Chem.*, **29**, 1092–1102.

Crooks,G.E. *et al.* (2004) WebLogo: a sequence logo generator. *Genome Res.*, **14**, 1188–1190.

Dahiyat,B.I. and Mayo,S. L. (1997) De novo protein design: fully automated sequence selection. *Science*, **278**, 82–87.

Dantas,G. *et al.* (2003) A Large Test of Computational Protein Design: Folding and Stability of Nine Completely Redesigned Globular Proteins. *J. Mol. Biol.*, **332**, 449–460.

Ermolenko,D.N. *et al.* (2003) Noncharged amino acid residues at the solvent-exposed positions in the middle and at the C terminus of the alpha-helix have the same helical propensity. *Protein Science*, **12**, 1169–1176.

Fersht,A.R. (1995) Mapping the Structures of Transition States and Intermediates in Folding: Delineation of Pathways at High Resolution. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, **348**, 11–15.

Filikov,A.V. *et al.* (2002) Computational stabilization of human growth hormone. *Prot. Sci.*, **278**, 1452–1461.

Fromer,M. and Yanover,C. (2009) Accurate prediction for atomic-level protein design and its application in diversifying the near-optimal sequence space. *Proteins: Structure, Function, and Bioinformatics*, **75**, 682–705.

Gordon,D.B. *et al.* (2003) Exact rotamer optimization for protein design. *Journal of computational chemistry*, **24**, 232–243.

Hamill,S.J. *et al.* (2000) Conservation of folding and stability within a protein family: the tyrosine corner as an evolutionary cul-de-sac. *Journal of molecular biology*, **295**, 641–649.

Holder,J.B. *et al.* (2001) Energetics of Side Chain Packing in Staphylococcal Nuclease Assessed by Exchange of Valines, Isoleucines, and Leucines†. *Biochemistry*, **40**, 13998–14003.

Kauzmann,W. (1959) Some factors in the interpretation of protein denaturation. *Adv.\ Prot.\ Chem.*, **14**, 1–63.

Koga,N. *et al.* (2012) Principles for designing ideal protein structures. *Nature*, **491**, 222–227.

Kraemer-Pecore,C.M. *et al.* (2003) A de novo redesign of the WW domain. *Protein Science*, **12**, 2194–2205.

Lassila,K.S. *et al.* (2002) Evaluation of the energetic contribution of an ionic network to beta-sheet stability. *Protein Science*, **11**, 688–690.

Liu,Z. *et al.* (2001) Modeling the third loop of short-chain snake venom neurotoxins: roles of the short-range and long-range interactions. *Proteins*, **42**, 6–16.

Lorch,M. *et al.* (1999) Effects of Core Mutations on the Folding of a beta-Sheet Protein: Implications for Backbone Organization in the I-State†. *Biochemistry*, **38**, 1377–1385.

Marshall,S.A. *et al.* (2002) Electrostatics significantly affect the stability of designed homeodomain variants. *Journal of Molecular Biology*, **316**, 189 – 199.

Northey,J.G.B. *et al.* (2002) Hydrophobic core packing in the SH3 domain folding transition state. *Nature Structural & Molecular Biology*, **9**, 126–130.

Perl,D. *et al.* (2000) Two exposed amino acid residues confer thermostability on a cold shock protein. *Nat Struct Mol Biol*, **7**, 380–383.

Perl,D. and Schmid,F.X. (2001) Electrostatic stabilization of a thermophilic cold shock protein. *Journal of Molecular Biology*, **313**, 343 – 357.

Pokala,N. and Handel,T.M. (2005) Energy functions for protein design: Adjustement with protein-protein complex affinities, models for the unfolded state, and negative design of solubility and specificity. *J. Mol. Biol.*, **347**, 203–227.

Poso,D. *et al.* (2000) Progressive Stabilization of Intermediate and Transition States in Protein Folding Reactions by Introducing Surface Hydrophobic Residues. *Journal of Biological Chemistry*, **275**, 35723–35726.

Sandberg,W.S. and Terwilliger,T.C. (1993) Engineering multiple properties of a protein by combinatorial mutagenesis. *Proceedings of the National Academy of Sciences*, **90**, 8367–8371.

Schrödinger,L. (2010) The PyMOL Molecular Graphics System, Version 1.3r1.

Schwehm,J.M. *et al.* (1998) Stability Effects of Increasing the Hydrophobicity of Solvent-Exposed Side Chains in Staphylococcal Nuclease. *Biochemistry*, **37**, 6939–6948.

Schweiker,K.L. *et al.* (2007) Computational design of the Fyn SH3 domain with increased stability through optimization of surface charge charge interactions. *Protein Sci.*, **16**, 2694–2702.

Seeliger,M.A. *et al.* (2003) Weak Cooperativity in the Core Causes a Switch in Folding Mechanism Between Two Proteins of the cks Family. *Journal of Molecular Biology*, **325**, 189 – 199.

Ventura,S. *et al.* (2002) Conformational strain in the hydrophobic core and its implications for protein folding and design. *Nat. Struct. Biol.*, **9**, 485–493.

Yi,F. *et al.* (2003) Testing Hypotheses about Determinants of Protein Structure with High-Precision, High-Throughput Stability Measurements and Statistical Modeling†. *Biochemistry*, **42**, 7594–7603.

Yosef,E. *et al.* (2009) Computational Design of Calmodulin Mutants with up to 900-Fold Increase in Binding Specificity. *Journal of Molecular Biology*, **385**, 1470 – 1480.

3.4 **Fast search algorithms for Computational Protein Design**

In order to render more accessible the CFN-based approaches to the scientific community, we implemented them into the *osprey* software during the three months that I spent in the group of Bruce Donald (University of Duke, North Carolina, USA). Another motivation for integrating the CFN methods into this software was to take advantage of the developments already carried out by Donald's group, especially to handle the molecular flexibility in CPD. In addition, we proposed novel methodological advances derived from CFN methods and new variable- and value-ordering heuristics to improve further the performances of the combinatorial optimization phase.

This work is presented hereafter and it should be submitted shortly for publication. This is a joined work with Bruce Donald's group and Thomas Schiex's group.

Fast search algorithms for Computational Protein Design

ABSTRACT

In previous work, we demonstrated the efficiency of Cost Function Network optimization on various Computational Protein Design problems against a broad range of combinatorial optimization technologies including 0/1 Linear Programming, 0/1 Quadratic Programming, 0/1 Quadratic Optimization, Weighted Partial MaxSAT, Graphical Model optimization problems and the well establish CPD dedicated framework Dead-End Elimination/A* (DEE/A*). Along this line, the aim of the work disclosed herein was to develop new CPD-dedicated search algorithms and heuristics and to implement them into the well-established CPD package *osprey*.

New Best-First methods using the CFN lower bound as heuristic are introduced here and they led to an important speedup compared to the well-known DEE/A^* framework. A new CPD-dedicated CFN upper bound tuning heuristic has also been developed in order to improve the CFN-based search. This led to an important increase in the pruning efficiency at root node. Finally, we designed new cost-based ordering heuristics which gave comparable performance to the well-established heuristics.

INTRODUCTION

Computational Protein Design (CPD) has become a valuable tool for tailoring proteins with desired biophysical and functional properties and for assessing our understanding of protein sequence-structure-function relationships. By combining physico-chemical models governing relations between protein amino-acid composition and the protein three-dimensional structure with advanced computational algorithms, CPD seeks to identify one or a set of amino-acid sequences that fold into a given 3D-scaffold and that will bestow the re-designed protein with targeted properties. Such rational protein design approaches have been successfully applied to alter intrinsic properties (stability, binding affinity…) of existing proteins or to endow them new functionalities, thus leading to the generation of novel enzymatic catalysts, binding pairs of proteins, protein inhibitors, and large oligomeric ensembles[1]–[6]. The field of application of this technology is broad, ranging from medicine, biotechnology, and synthetic biology to nanotechnologies[7]–[9].

Despite its notable results, substantial methodological advances are still needed to improve CPD performances and extend its effective application. The success of CPD predictions actually depends on several elements, among which, the biologically meaningful modeling of the design problem, the accuracy of the energy and objective functions used to assess fitness of the predicted sequence-structures, and the efficiency of the search algorithms to find solutions in a timely manner. However, CPD approaches have to strike a compromise between speed and accuracy to face the exponential size of the search space defined by the composition of protein sequences and conformations. Most of the CPD methods rely thus on: 1) a coarse-graining of the structure as a sequence of discrete side-chains rotamers, 2) an assumption of minimal backbone conformational flexibility, where a fixed backbone or a set of possible backbones are used, and 3) an approximation of the energy model as a pairwise decomposition. The problem of searching for a model of global minimum energy (GMEC : Global Minimum-Energy Conformation) over the conformational space of rotamers and possibly backbones being NP-hard ([10]), a variety of methods, both meta-heuristics (Monte Carlo simulated annealing, genetic algorithms,…[11]) and provable algorithms (Dead-end elimination, Branch-and-bound algorithms, integer linear programming, dynamic programming,…[11]–[13]) have been proposed over the years. However, there is still a need for more efficient optimization techniques, capable of exploring vaster combinatorial spaces, representing more realistic, flexible protein models.

This paper focuses on exact optimization techniques. Since provable methods know when a global optimum is reached, the search can be stopped with confidence and an exact solution obtained, sometimes in significantly less time than with meta-heuristics. It has also been observed that the accuracy of meta-heuristic approaches tends to degrade in unpredictable ways as the problem size increases [14]. Finally, exact methods are useful for improving biophysical models because they ensure that discrepancies between CPD predictions and experimental results come exclusively from modeling inadequacies and not from the algorithm. Currently, the most usual provable and deterministic methods for CPD rely on the Dead-End Elimination (DEE) theorem and the A^* algorithm [15]. DEE is used as a pre-processing method. It removes rotamers which are energetically dominated by other rotamers and therefore useless to identify a global optimum. This idea has later been extended to prune pairs or higher order combinations of rotamers at different residues in order to improve the pruning power [16]–[18]. However, because CPD is NP-hard, the polynomial time DEE algorithms usually cannot identify a unique sequence-conformation model. It is followed by an A^* search (a Best-First search) which expands a sequence-conformation tree by tentatively assigning rotamers to residues. By relying on a dedicated admissible heuristics, A^* is able to produce an energy-sorted list of solutions. But the worst-case exponential time and memory consumption of A^* means that it can easily choke on problems with many undominated rotamers.

In a recent work, we have shown that the rigid backbone discrete rotamer CPD problem could be formulated and efficiently solved as a Cost Function Network (CFN) [19]–[21]. CFN algorithms are able to handle complex combinatorial spaces which are out of reach of usual DEE/A^* approaches as implemented in the CPD-dedicated software *osprey* [3], [22]. The *toulbar2* CFN solver provides speedups of several orders of magnitude both to provably find the GMEC and to exhaustively enumerate ensembles of near-optimal solutions, offering an attractive alternative to the usual DEE/ A^* approaches.

Herein, we report further methodological advances derived from this CFN-based method [19]–[21] together with their implementation within the *osprey* software. With this integration inside a usual CPD framework, it is possible to benefit from the speedups offered by CFN algorithms on a variety of problems tackled by *osprey*, including those that offer some flexibility modeling [23]. The performances of these methods have been assessed on the design of more stable proteins and cofactor-bound proteins, as well as protein-ligand and protein-protein interfaces. The results were ultimately compared to those obtained using the DEE/A* approach implemented in the *osprey* software.

BACKGROUND

The CPD problem formulation

The rigid backbone and discrete rotamer CPD problem is defined by: *i*) a fixed backbone of a protein structure; *ii*) a set of amino-acid residues to be designed, called designable residues; *iii*) a group of allowed amino-acids for each designable residue and their respective set of discrete low energy side-chain conformations, so-called rotamers and *iv*) pairwise atomic energy functions to discriminate between models. Rotamers usually correspond to cluster centers of well represented amino-acid conformations within 3D protein structures. In the case of protein-ligand systems, the conformational flexibility of peptide ligands is described similarly by discrete rotamer libraries. In the case of non-peptide ligands, the treatment of organic molecule flexibility is often left to the user to pre-calculate an ensemble of low energy conformers for the ligand (playing the role of rotamer library).

A sequence-conformation model is defined by the choice of one specific amino-acid with one associated conformation (rotamer) for each designable residue. Its total energy (E_{total}) is defined by:

$$\mathrm{E_{total}} = \mathrm{E_c} + \sum_{\mathrm{i}} \mathrm{E(i_r)} + \sum_{i} \sum_{\mathrm{j}<i} \mathrm{E(i_r, j_s)} \quad (1)$$

where E_c is a constant energy contribution capturing interactions between fixed parts of the model, $E(i_r)$ depends on rotamer r at position i (and its reference energy) and $E(i_r, j_s)$ is the pairwise interaction energy between rotamer r at position i and rotamer s at position j.

The combinatorial optimization problem is to find a complete rotamer assignment that provably minimizes E_{total}.

Modeling CPD as a Cost Function Network

The problem of finding the set of rotamers that will minimize the total energy (E_{total}) can easily be formulated as a Cost Function Network problem (CFN) [19]–[21].

A CFN P is defined by a set of variables which are each involved in a set of local cost functions [24]. Formally, a CFN P is a triple $P = (X, D, C)$ where $X = \{1,2,..,n\}$ is a set of n variables. Each variable $i \in X$ has a discrete domain $d_i \in \mathrm{D}$ that defines the set of values that it can take. A set of local cost functions C defines a network over X. Each cost function $c_S \in C$ is defined over a subset of variables $S \subseteq X$ (called its scope), has a domain $\prod_{i \in S} d_i$ and takes integer values in $\{0,1,2,..,k\}$. The cost k represents a maximum tolerable cost, and can be infinite or set to a finite upper bound. In binary CFN, cost functions involve at most two variables. Values or pairs of values which are forbidden by a cost function are simply mapped to k. The global cost of a complete assignment A is defined as the sum of all cost functions on this assignment (or k if this sum is larger than k). The Weighted Constraint Satisfaction Problem defined by P consists in finding an assignment of all variables that minimizes this global cost. Notice that it is usually assumed that C contains one constant cost function, with an empty scope, denoted as $c_\emptyset$. Since all cost functions in a CFN are non-negative, this constant cost function $c_\emptyset \in C$ defines a lower bound on the optimization problem.

The CPD optimization problem, in its pairwise-decomposed form, can be easily formulated as a binary CFN. Every designable amino-acid residue i is represented by a variable i and the set of rotamers available to the residue defines its domain d_i. Then, each energy term in E_{total} is represented as a cost function [20], [21], [25]. The constant term E_c is captured as the constant cost function with empty scope ($c_\emptyset$) and terms $E(i_r)$ and $E(i_r, j_s)$ are represented by unary and binary cost functions involving the variables of the corresponding residues. Energy terms can be mapped to positive integers through shifting and scaling according to desired precision [20], [21], [25]. Such operations preserve the set of optimal solutions and an optimal solution of the CFN is an assignment that defines a GMEC for the CPD problem.

In contrast with CPD, where dominance analysis through the DEE theorem dominates, the fundamental idea in CFNs relies on so-called Equivalence Preserving Transformations (EPTs). An EPT is a local transformation of the CFN which can shift cost (or energy) between cost functions of intersecting scopes without changing the global energy distribution. These EPTs are iteratively applied in so-called local consistency enforcing algorithms that iterate EPTs until the CFN satisfies the local consistency property. Many of these local consistency properties and associated polynomial time enforcing algorithms have been defined. Depending on the locality of

the property, which may apply to one variable, one cost function or more, they are called Node, Arc or higher order consistencies. As an example, the node consistency of a variable i with associated cost function c_i requires that d_i contains at least one value v such that $c_i(v) = 0$ and no value w such that that $c_\emptyset + c_i(w) = k$ (the forbidden cost). Equivalently, this means that there is at least one value that does not increase cost locally and no value that would lead to intolerable costs. If a variable does not satisfy these properties, then by deleting values and shifting costs to $c_\emptyset$, the variable can be made node consistent. The amount of pruning therefore increases with smaller values of the upper bound k. Arc consistencies are defined similarly but are significantly more involved (see eg. [26]–[28]). Globally, the essential effect of enforcing local consistencies is that values may be pruned and the constant cost function $c_\emptyset$, a global lower bound on the optimum, may be increased. This is obtained without changing the global energy distribution.

Compared to local consistency enforcing, DEE, which has also been studied in CFN under the name of substitutability ([21], [29], [30]) does not preserve the global energy distribution as it may remove some (but not all) optimal solutions and sub-optimal solutions.

Since local consistency algorithms are polynomial time algorithms, they cannot solve all CFNs and an exhaustive Depth First Branch and Bound (DFBB) tree search is used to provably solve the problem. Contrarily to the DEE/A^* approach however, dominance and local consistency analysis are not performed only at the beginning of the search, as preprocessing, but also incrementally maintained at each node of the tree search. By increasingly simplifying the problem and strengthening the lower bound $c_\emptyset$, they give information that can be used to prune and heuristically guide the search in a tree defined by branching.

Our recent work [19]–[21] highlighted the power of these DFBB method to efficiently solve various CPD problems and hence, spurred new developments to tackle more complex and challenging CPD problems.

Branching schemes

When preprocessing is unable to solve the problem, tree search algorithms such as A^* or DFBB try to simplify the problem by making assumptions on variables. Each additional assumption generates a new node, son of the previous node. In the DFBB search, successive assumptions are made until either all variables become assigned to a single value: a new solution is found and the upper bound k can be updated to its cost (we are only interested in better solutions), or the current lower bound $c_\emptyset \geq k$. In this last case, we know that with the current assumptions, we cannot reach a cost less than $c_\emptyset$ and therefore less than k. We need to backtrack, that is we reconsider our last assumption and *branch* on a new assumption.

The most obvious type of branching uses a chosen variable i and considers all its current values as possible assumptions. This is called n-ary branching. An alternative branching scheme consists in selecting a variable i and a value a and uses as assumptions the fact that the variable i either takes the value a, or not (and the value can be removed from the domain). By exploiting results in proof theory, this scheme has been shown to be more powerful than the previous one (with a given local consistency being maintained at each node, it may explore an exponentially smaller number of nodes than the previous one, and the converse is impossible [31]. N-ary branching can however be polynomially better on some problems. An even more general branching method is dichotomic branching where the domain of a chosen variable i is split in two chosen sets.

Variable and value ordering heuristics

If branching schemes define the shape of the tree explored, DFBB also needs to choose the next assumption to make. Variable and value ordering heuristics are used to choose respectively the next variable to branch one and the next value to consider for this chosen variable.

Variable ordering heuristics may have a tremendous effect on the efficiency of the search algorithm. They are all based on the 'fail-first' principle [32] : 'To succeed, try first where you are most likely to fail'. Several measures have been used to try to evaluate the likelihood of failing by fixing a variable. One simple measure is the current size of the domain (*dom*) [32]. Under this measure, the variable which has the smallest domain should be assigned next. To take into account the number of cost functions that involve a variable (the so-called degree of the variable), more sophisticated heuristics select the variable that has the minimum ratio of the domain size over the current degree (*dom/ddeg* [33]), over the degree weighted by the number of failures observed in the past for each cost function (*dom/wdeg* [25], [34]) or by the sum of the median cost functions (*dom/cmed* [20], [21]). Additionally, the last conflict heuristics [35] simply tries to select the last variable that led to inconsistency during search (if any).

Once a variable is chosen, binary and n-ary scheme need to choose the next value to consider. The effect of value ordering heuristics is often less dramatic than for variables. However, a good value ordering heuristics may help to quickly find a good solution. Because we are then interested only in strictly better solutions, we may set k to the cost of this new solution and improve pruning. The most usual value ordering heuristics in CFNs is to choose first a value a that has a unary cost $c_i(a) = 0$. Such a value always exists thanks to Node Consistency enforcing [36].

Limited Discrepancy Search

If good value ordering heuristics may allow to quickly find a good solution, they may fail. This may be because of the very first assumption made that will not be reconsidered before a complete subtree is explored. Limited discrepancy search (LDS) [37] tries to overcome this situation by exploring only paths that have a bounded number of variable assignments that differs from those defined by the value heuristics (called discrepancies). LDS can often find good solutions much faster than DFBB search. The best solution found can be used to set the initial upper bound k. Obviously, the power of LDS strongly depends on the quality of the heuristic used for variable and value ordering.

CONTRIBUTIONS OF THE CURRENT WORK

The work reported herein provides novel CFN-based methodological advances targeted at highly complex CPD problems. It also makes most of the CFN framework together with our new developments more accessible to the CPD community by implementing them in the open source CPD framework *osprey* 2.0 [22]. Note that we focus in this section on these methodological developments, but all the technology described in the Background section has also been implemented in *osprey* within this work.

Best-First search has several advantages over DFBB search. It is known that it always develops less nodes than DFBB if the same lower bound (or admissible heuristics in A^* terminology) is used. It is also able to directly produce a sorted list of solutions of increasing energies. In order to

assess the efficiency of local consistency enforcing and associated lower bound in this context, two Best-First approaches were developed. The first one is a plain A^* algorithm using the CFN lower bound as its admissible heuristics. It will be denoted as "Best-First" to avoid confusion with the traditional DEE/$A*$ algorithm. The second one is another A^* algorithm using as heuristics a strengthened CFN lower bound, obtained by a limited DFBB exploration of the current node (using the best lower bound found at the leaves of this depth limited search as the new lower bound). This method is referred to as Best-First-DL. The performances of both search methods were compared to those of the traditional A^*search, commonly used in CPD, on various protein design problems.

To better take into account the specificities of CPD, new variable and value ordering heuristics were designed and integrated within the CFN-based search. Their performances within DFBB search were assessed on real CPD instances.
Since the initial value of the upper bound k may have a strong influence on pruning and therefore efficiency, we also developed a new amino-acid based upper bound heuristic to tighten the initial upper bound. Four heuristics to update this upper bound were compared with each other and a best option identified. Finally, specific properties of the CPD problems were used to build new binary branching strategies based on amino-acid types within the DFBB approach.

METHODS

New variable and value ordering heuristics

In our previous work [21], we developed a new variable ordering heuristics (*dom/cmed*) based on the median costs in the cost table of all the cost functions involving each variable.

Two new value heuristics, defined by dedicated value metrics, are proposed in this work: the value with a minimum metric should be selected first. For each value $a \in d_i$ a first metric combines the unary cost of the value with the median of all the cost that may incur with connected variables:

$$hval_c(a) = C_i(a) + \sum_{j \neq i} \underset{b \ni D_j}{\text{median}} \left[C_{ij}(a, b) \right] \quad (2)$$

A more sophisticated variant also includes the unary costs of each neighbor variables, leading to:

$$hval_v(a) = C_i(a) + \sum_{j \neq i} \underset{b \ni D_j}{\text{median}} \left[C_{ij}(a, b) + C_j(b) \right] \quad (3)$$

The corresponding value ordering heuristics are denoted *cmed* and *vmed*, respectively. Of note, in all our branching schemes, the unary support supplants the value ordering. Hence, except during LDS, the value ordering heuristics do not affect the sheer binary branching scheme. Nevertheless, when either a simple or amino acid-based dichotomic branching mechanism is activated, the heuristic is applied. These heuristics are used as a basis for the definition of our new side chain positioning-based upper bounding.

As for variable ordering heuristics, we consider metrics defined by the ratio of the domain size over the median unary costs (*dom/umed*) and a second variable ordering heuristic using minimum over all values a of the ratio of the domain size by the metric $hval_v(a)$ denoted as *dom/vmed*.

Side chain positioning-based upper bounding

To quickly identify a good upper bound, we reduce the CPD problem to a side chain positioning (SCP) problem by restricting the domain of every mutable position to rotamers corresponding to an unique amino-acid. Clearly, the cost of any solution of this restricted SCP problem defines an upper bound of the original CPD problem. To this end, we use a generic heuristic $h_{AA}(i)$ to select the amino-acid type (AA) at position i. Four such heuristics were considered. The first one selects the amino-acid of the rotamer selected by the current value ordering heuristic (*Value−heuristic−AA*). The second one selects the wild-type amino-acid if it is still available after local consistency enforcing and otherwise uses the first option (*Wt−or−value−heuristic−AA*). The third one (*ArgMin−AA−min*) selects the amino-acid that satisfies:

$$h_{AA}(i) = \underset{aa \in AAs(D_i)}{argMin} \min_{a \in D_i |\, type(a)=aa} (hval_v(a)) \quad (4)$$

Finally, the last option (*ArgMin−AA−median*) selects the amino-acid that satisfies:

$$h_{AA}(i) = \underset{aa \in AAs(D_i)}{argmin} \operatorname{median}_{a \in D_i |\, type(a)=aa} (hval_v(a)) \quad (5)$$

After solving the side chain placement problem and updating the upper bound k, local consistency enforcing is maintained on the initial CPD problem. All this is only done if at least one residue has more than one amino-acid identity in its domain following local consistency enforcing.

New CFN amino-acid based branching schemes

Our new amino acid selection heuristics $h_{AA}(i)$ can be used to define novel dichotomic branching schemes. For a chosen variable i, an amino-acid is selected according to the $h_{AA}(i)$ heuristic and a dichotomic branching is defined by a first set of values defined by the rotamers of the selected amino-acid type, the second set containing remaining values. This dichotomic branching incrementally reduces the CPD problem to a side-chain placement issue: each time a branching is performed, the domain of a variable is restricted to a single amino-acid in the first branch. It is applied as a dichotomic branching scheme embedded in a binary branching. It can either be used in full (referred to as Binary-AA-Branch in the following) or only for domains of large sizes (referred to as limited-Binary-AA-Branch).

Best-First Search with a CFN lower bound

The completeness of the A^* search method used in the DEE/A^* approach [38] relies on the use of a so-called “admissible heuristic function”. This function must provide an optimistic estimate of the total energy of all conformations below the current node, i.e. must be a lower bound on the optimum of the current problem. Thus, the lower bound defined by $c_\emptyset$ and local consistency enforcing can be directly plugged in the A^*algorithm and replace the existing heuristics.

Best-First search methods maintain a priority queue of open nodes for expansion. CFN processing by local consistency is performed on each node, pruning the domains of every associated sub-problem and offering the lower bound $c_\emptyset$ as a priority. When a node is considered for the expansion, n-ary branching is used, local consistency is applied on each generated independent sub-problem and those that have a lower bound less than the current upper bound k are inserted in the queue.

One of the major weakness of Best-First methods being worst-case exponential space because of this queue, a bounded depth refinement of the lower bound of every node considered for insertion in the expansion queue was implemented (referred to as Best-First-DL). If the obtained lower bound exceeds k there is no need to insert it in the queue. The depth limit used is an adjustable parameter.

Our CFN-based CPD framework

The overall strategy of the Cost Function-based CPD framework developed in this work is described in Figure 1. Local consistency (possibly with integrated DEE dominance analysis [21], [30]) is always maintained during the whole process. A first upper bound is sought by LDS and optionally, improved by solving the SCP problem defined by one selected heuristic $h_{AA}(i)$. Finally, the remaining search space is explored to extract the GMEC and enumerate suboptimal solutions when desired. Two main alternative search strategies can be performed: a Best-First Search or a DFBB. Subsequently, n-ary or binary branching may apply. Dichotomic amino-acid branching schemes and (simple) dichotomic split are both an option for binary branching. In all the search strategies, a dynamic variable and value ordering can be performed. Finally, when Best-First is combined with DFBB, we get Best-First-DL.

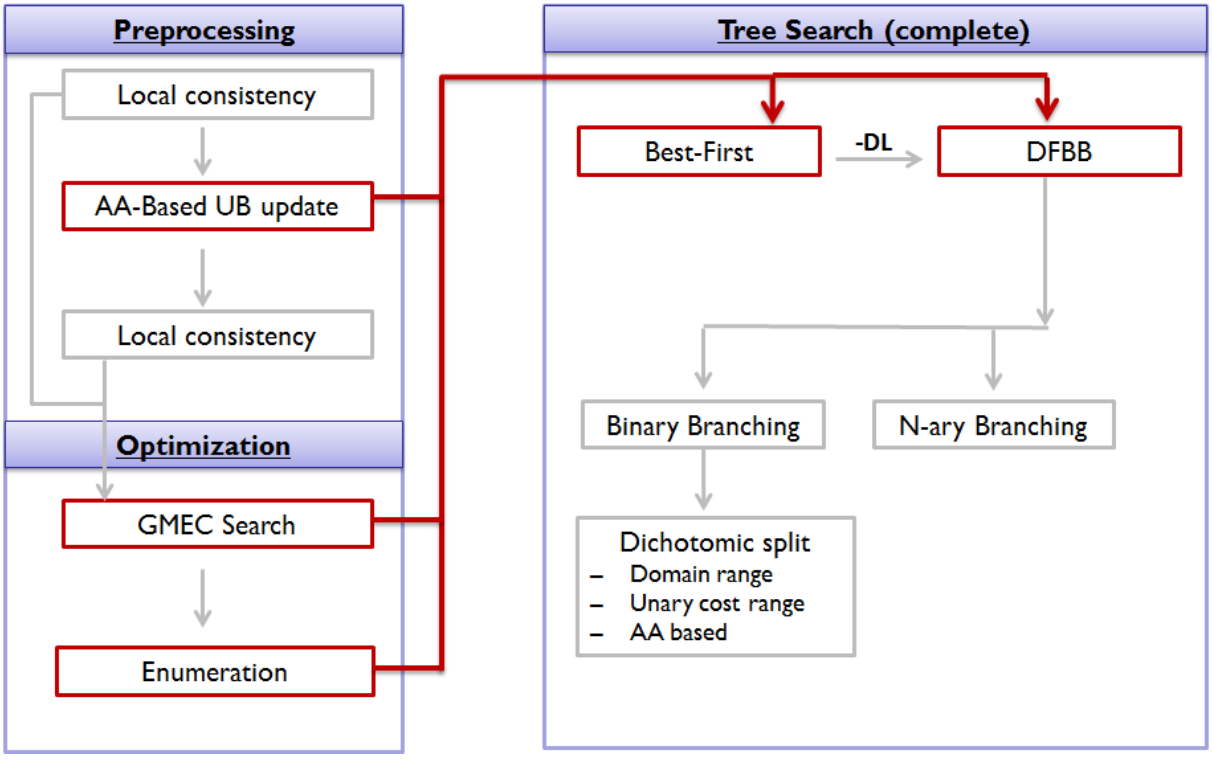


Figure 1 CFN-based CPD Framework

The complete approach has been implemented in *Osprey* (version 2.0, [22]) which allows for discrete and continuous modeling of the protein conformation at the side-chain and backbone level [39]. All those models ultimately reduce to a CPD pairwise energy matrix which can be optimized by CFN-based algorithms. Thus, the present work increases the efficiency of this CPD package without losing provability, giving access to high dimensional protein design problems.

RESULTS AND DISCUSSION

The benchmark set of 30 CPD instances used in our previous studies [20], [21] was used to assess the performances brought by our contributions. This benchmark set includes a variety of protein structures, alone or in complex with a protein or a ligand, derived from high resolution structures deposited in the Protein Data Bank (PDB). In these selected systems, diverse sizes of sequence-conformation spaces are present, varying by the number of mutable residues, the number of alternative amino-acid types at each position and the number of rotamers for each amino-acid.
All computations were performed on one core of an AMD Opteron^{TM} Processor 6176@2.3 GHz. We used 128GB of RAM and a 9,000 sec timeout. The integrated CFN with incremental DEE (DEE^1 option) pruning option previously described was implemented here and activated in all CFN-based experiments [21].

First, the four heuristics designed for the SCP-based upper bound tuning were assessed against each other in order to identify the best option enabling to efficiently reduce the rotamer space before the search tree. Secondly, within the DFBB search method, we assessed the performance of the variables (with last conflict [35]) and values ordering heuristics, as well as those of the new amino-acid based dichotomic branching schemes for solving the GMEC identification problem. Finally, the performances of the new Best-First and Best-First-DL methods were compared to those of the A^* algorithm, for solving the GMEC identification problem as well as for enumerating an ensemble of sub-optimal models.

Side-chain positioning based upper bounding

When tree search is used, any value pruned at the root node also prunes an exponential number of conformations at leave nodes. Thus, any effort to increase pruning at the root node is likely to improve speed. This can be simply obtained by strengthening the initial upper bound k.

We evaluated the effect of improved initial upper bounds provided by solving to optimality a simpler side chain positioning sub-problem [13] and using the optimum found as an upper bound of the initial CPD problem. The median cost variable ordering heuristic (*dom/cmed*) was used in conjunction with the increasing unary cost value ordering heuristic. DFBB with a binary branching scheme is used.

We tested our four amino-acid selection heuristics (*Value−heuristic−AA*, *Wt−or−value−heuristic−AA*, *ArgMin−AA−min* and *ArgMin−AA−median*) and for reference we also performed a run without these options (Figure 2).

Without any such initial upper bound, the GMEC could never be identified at the root node. With the initial SCP based upper bounding, respectively 20, 17, 18 and 17 cases (out of 23 solved cases) were immediately solved at the root node using respectively, the heuristics *Value−heuristic−AA, Wt−or−value−heuristic−AA, ArgMin−AA−min and ArgMin−AA−median*. As shown in Figure 2 with all four heuristics, up to *94%* of values were pruned at the root node, instead of merely *41%* without upper bounding.

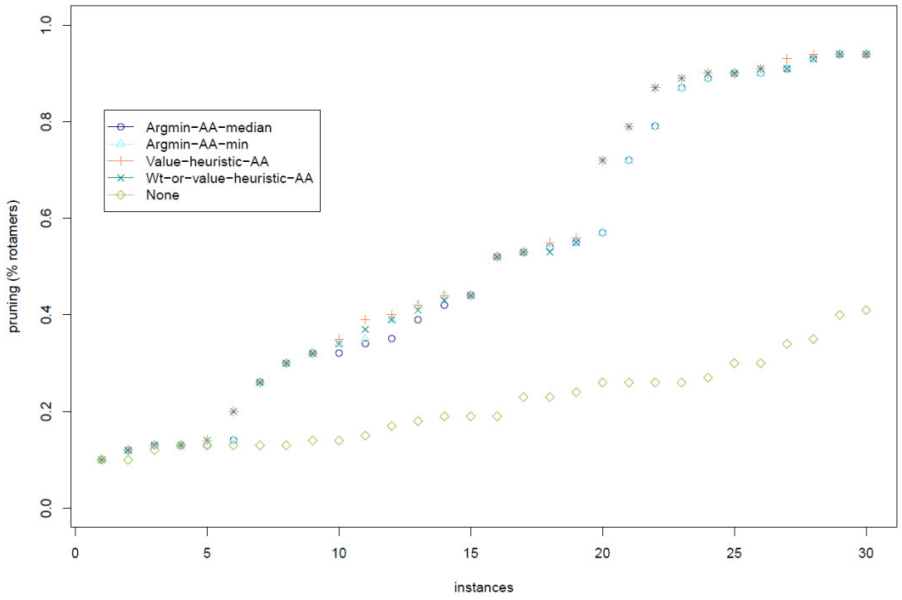


Figure 2 Improvement of the pruning power at preprocessing step

New variable and value ordering heuristics

The performance of the two new variables ordering heuristics, *dom/umed* and *dom/vmed* as well as *dom/cmed* [20], [21] were assessed against the widely used variable ordering heuristic, *dom/wdeg* [34] and the simple *dom* heuristic[32] using the *DFBB* search method for identifying the GMEC (Table 1). A DFBB was performed and the option *Value−heuristic−AA was* used for amino-acid-based upper bound tuning heuristic at root (assessed thereafter). The increasing unary cost value ordering was performed.

The new heuristic *dom/vmed* has the same time complexity as *dom/cmed.* The most expensive part of the heuristic is the contribution of binary terms. When considering variable i, *dom/cmed* involves the corresponding unary cost function C_i and all the binary cost functions C_{ij}. However, *dom/vmed* involves also these terms but it additionally considers C_j. The second new variable ordering heuristic *dom/umed* is a simpler variant of *dom/cmed*. Indeed, only unary terms are considered in *dom/umed*.

All the heuristics managed to solve the same 23 design cases with performances varying according to the instance considered. Among the 23 cases solved by DFBB with the *dom/wdeg* variable ordering heuristic (considered as reference), the GMEC was found faster (resp. slower) for 8(13), 12(11), 12(11) and 11(12) cases using respectively, *dom*, *dom/cmed*, *dom/umed* and *dom/vmed*. Thence, the new cost based variable ordering heuristics offer comparable performances to the widely used *dom/wdeg* heuristic (which is considered very efficient for increasing the speed of the search [25], [34]). The averaged performance gain in term of runtime for *dom*, *dom/cmed*, *dom/umed* and *dom/vmed* is respectively, 48.0; 223.7; 147.7 and 142.8 sec compared to *dom/wdeg*.

Next, we also evaluated the efficiency of designed value ordering heuristics, *vmed* and *cmed*, in combination with the variable ordering heuristic *dom/cmed* (Table 1, columns *dom/cmed + vmed* and *dom/cmed + cmed*). The same 23 instances are all solved again. Adding *vmed* or *cmed* enabled to solve respectively 13(10) and 12(11) cases with lower (resp. greater) runtime

compared to the *dom/wdeg* reference. The averaged performance gains in term of runtime are respectively, 229.6 and 216.2. Therefore, this value ordering heuristics do not seem to provide additional performance gain compared to the usage of the increasing unary cost ordering (i.e. *dom/cmed* with default value ordering which led to a runtime gain of 223.7 compared to *dom/wdeg*, as mentioned above).

Table 1: Ordering heuristics - CPU-time to solve the GMEC using DFBB.

PDB accession code	Sequence Conformation Space Size	*dom/wdeg*	*dom*	*dom/cmed*	*dom/umed*	*dom/vmed*	*dom/cmed + vmed*	*dom/cmed + cmed*
1MJC	4.36E+26	2.9	**1.9**	**1.8**	**1.8**	**2.0**	**1.9**	**1.9**
1CSP	5.02E+30	125.6	136.7	**40.7**	**111.5**	**32.9**	**38.9**	**43.0**
1BK2	1.18E+32	21.8	22.3	22.6	24.2	30.3	**20.2**	25.1
1SHG	2.13E+32	16.4	16.4	**13.7**	28.4	**12.0**	**13.8**	**13.6**
1CSK	4.09E+32	3.4	3.8	3.8	4.0	4.5	3.6	3.9
1SHF	1.05E+34	3.5	**3.4**	3.6	3.6	4.1	3.8	4.2
1FYN	5.04E+36	267.8	286.3	275.0	308.5	404.1	291.5	282.0
1PIN	5.32E+39	468.3	468.6	**347.7**	505.6	530.1	**361.0**	**374.1**
1NXB	2.61E+41	3.9	5.3	4.9	5.3	5.3	4.9	5.4
1TEN	6.17E+43	7.6	7.6	8.6	**6.4**	8.4	8.4	9.4
1POH	8.02E+43	17.2	**17.0**	17.3	**14.9**	**15.8**	**15.4**	**16.7**
2DRI	1.16E+47	-	-	-	-	-	-	-
1FNA	3.02E+47	109.7	177.1	121.8	**65.0**	123.8	111.6	**106.7**
1UBI	2.43E+49	5,275.0	**4,141.3**	**1,070.2**	**2,749.7**	**1,248.9**	**962.2**	**951.3**
1C9O	3.77E+49	436.3	**422.6**	**223.2**	**338.5**	**205.2**	**186.9**	**190.5**
1CTF	3.95E+51	531.1	**506.0**	**382.1**	**352.6**	**495.5**	**459.5**	**477.6**
2PCY	2.34E+52	10.7	12.6	**9.4**	**9.2**	11.7	13.2	11.3
1DKT	3.94E+58	382.8	391.2	**202.3**	**216.1**	**261.3**	**188.5**	**187.2**
2TRX	9.02E+59	124.7	125.6	138.2	132.0	**120.2**	134.0	126.2
1CM1	3.73E+63	-	-	-	-	-	-	-
1BRS	1.67E+64	-	-	-	-	-	-	-
1CDL	5.68E+65	-	-	-	-	-	-	-
1LZ1	1.04E+72	1,273.0	1,321.2	**382.8**	**698.1**	**744.5**	**432.2**	**498.7**
1GVP	1.51E+78	-	-	-	-	-	-	-
1RIS	1.23E+80	-	-	-	-	-	-	-
2RN2	3.68E+80	316.4	**283.0**	**247.0**	**275.7**	412.4	**222.8**	414.2
1CSE	8.35E+82	19.1	20.8	23.0	20.1	21.2	25.1	24.4
1HNG	3.70E+88	946.4	**876.9**	1,712.2	1,063.8	2,417.7	1,629.5	1,668.6
3CHY	2.36E+92	-	-	-	-	-	-	-
1L63	2.17E+94	543.9	556.6	**510.0**	575.1	**511.9**	**496.7**	**499.4**
# Nb of times faster (resp. slower) than *dom/wdeg*			8(13)	12(11)	12(11)	11(12)	13(10)	12(11)

A '-' indicates an exceeded memory size (128G) or computation time (9000 sec).

Binary amino-acid Branching

New binary branching heuristics which better take into consideration the CPD characteristics are proposed herein. Indeed, since values in CPD correspond to amino-acid rotamers, it is possible that restricting the set of allowed amino acids to the identity of a selected amino-acid (aa_i) gives lower cost compared to the opposite problem that excludes rotamers of amino acid type aa_i. In that case, a binary branching at amino-acid level may speed up the search. Subsequent problems become simpler not only because of the domain size reduction but also because the initial CPD problem incrementally becomes a side chain placement problem which has been shown simpler to solve[13].

We used the reference ordering heuristic, *dom/wdeg* (as in Table 1) and the $Value{-}heuristic{-}AA$ for amino-acid-based upper bound tuning heuristic at root. Of note, for Binary-AA-Branch and Limited-Binary-AA-Branch, this option is incrementally performed during the search (and not at root). Simple Binary branching plays the role of the reference experiment (since that option is

generally used to solve CFN), and we assessed against it, the N-ary branching, the Binary branching with dichotomic split, the Limited-Binary-AA-Branch and the Binary-AA-Branch (Table 2).

Table 2: Assessment of branching options for CPU-time for solving the GMEC.

PDB accession code	Sequence Conformation Space Size	Binary branching	N-Arybranching	Binary branching with dichotomic split	Limited-Binary-AA-Branch	Binary-AA-Branch
1MJC	4.36E+26	1.8	2.1	2.5	2.1	3.3
1CSP	5.02E+30	40.7	**26.3**	51.1	**31.6**	**32.1**
1BK2	1.18E+32	22.6	**22.0**	**22.1**	**22.5**	**22.4**
1SHG	2.13E+32	13.7	**8.0**	**10.8**	**9.6**	**8.5**
1CSK	4.09E+32	3.8	4.2	**3.6**	**3.6**	**3.5**
1SHF	1.05E+34	3.6	3.7	3.9	**3.5**	**3.4**
1FYN	5.04E+36	275.0	353.2	446.9	**158.8**	**153.1**
1PIN	5.32E+39	347.7	1,651.0	-	**305.8**	**311.1**
1NXB	2.61E+41	4.9	5.2	**4.4**	**4.5**	**4.3**
1TEN	6.17E+43	8.6	**8.0**	**6.8**	**8.4**	**7.7**
1POH	8.02E+43	17.3	**16.1**	**16.2**	18.4	17.4
2DRI	1.16E+47	-	-	-	-	-
1FNA	3.02E+47	121.8	**113.2**	**67.3**	**74.2**	**83.1**
1UBI	2.43E+49	1,070.2	**472.3**	**528.5**	**912.0**	**972.5**
1C9O	3.77E+49	223.2	466.5	**173.0**	**186.7**	**178.5**
1CTF	3.95E+51	382.1	**311.4**	**334.1**	**211.5**	**210.7**
2PCY	2.34E+52	9.4	10.3	9.9	9.8	10.1
1DKT	3.94E+58	202.3	**192.1**	217.3	**194.3**	**193.3**
2TRX	9.02E+59	138.2	**129.3**	291.8	**130.5**	**135.8**
1CM1	3.73E+63	-	-	-	-	-
1BRS	1.67E+64	-	-	-	-	-
1CDL	5.68E+65	-	-	-	-	-
1LZ1	1.04E+72	382.8	427.8	**292.3**	**364.4**	391.0
1GVP	1.51E+78	-	-	-	-	-
1RIS	1.23E+80	-	-	-	-	-
2RN2	3.68E+80	247.0	**241.7**	**165.6**	**222.2**	**202.7**
1CSE	8.35E+82	23.0	23.9	24.7	**21.0**	**21.5**
1HNG	3.70E+88	1,712.2	**704.2**	**706.9**	**1,026.5**	**1,017.9**
3CHY	2.36E+92	-	-	-	-	-
1L63	2.17E+94	510.0	**501.5**	**509.6**	541.3	547.2
# Nb of cases solved strictly faster (resp. slower) than when using Binary branching			13(10)	14(9)	19(4)	18(5)

A '-' indicates an exceeded memory size (128G) or computation time (9000 sec).

N-ary Branching, Binary branching with dichotomic split, limited-Binary-AA-Branch and Binary-AA-Branch enabled to find the GMEC with a faster (resp. slower) runtime for respectively 13(10), 14(9), 19(4) and 18(5) cases over the 23 cases solved by both approaches. Hence, the amino-acid based dichotomic branching schemes improved the binary branching as designed for and were shown to perform the best. Nevertheless, unexpectedly, N-ary Branching gives a slightly better performance compared to a plain binary branching.

Best-First Search with a CFN lower bound

The GMEC solving performances of the new Best-First and Best-First-DL algorithms using the CFN lower bound were compared to those of the usual A^* algorithm used in CPD after a DEE pre-processing. The same reference ordering and amino acid selection heuristics () of Table 1 (*Value−heuristic−AA*) are used. For Best-First-DL, a depth limit of 3 was used.

A^* solved 17 cases while Best-First and Best-First-DL managed to solve 21 cases within the 9000 sec timeout (Table 3). On all the instances that were solved by A^* and by the CFN Best-First methods (15 cases), the CFN based methods were found to be faster by up to 2 orders of magnitude . With dead-lines of 900 sec, the A^* approach solved 11 cases , while the Best-First

and Best-First-DL approaches solved 16 and 15 instances respectively. This clearly shows that the use of CFN lower bound instead of the traditional heuristic [38] used in A^* for CPD provides speedups, giving access to more complex designs. The variant using a bounded depth refinement of the lower bound does not perform better than a plain Best-First search.

In our previous work, we have shown that the DFBB approach implemented in the *toulbar2* solver outperformed the DEE/*A** approach of *osprey* by several orders of magnitude [20], [21], [25]. We therefore compared the performance of the DFBB now implemented in *osprey* against those of the Best-First and Best-First-DL (Table 3) . Out of the 30 design cases, the DFBB solved 23 instances within the 9000 sec and DFBB only failed on 7 instances, which were also unsolved by the other methods. Among the 23 cases solved by DFBB, 21 were solved in less than 900 sec. The DFBB search method remains therefore overall more efficient than a Best-First search method for handling large protein design problems.

Table 3: CPU-time for solving the GMEC using *A**, Best-First,Best-First–DL and DFBB.

PDB accession code	Sequence Conformation Space Size	*A*#*	Best-First	Best-First-DL	DFBB
1MJC	4.36E+26	4.6	1.9	1.7	1.8
1CSP	5.02E+30	200.0	33.7	28.6	40.7
1BK2	1.18E+32	93.2	23.6	25.6	22.6
1SHG	2.13E+32	138.0	9.7	12.1	13.7
1CSK	4.09E+32	41.7	3.9	4.5	3.8
1SHF	1.05E+34	44.3	3.7	3.7	3.6
1FYN	5.04E+36	622.0	364.1	377.1	275.0
1PIN	5.32E+39	-	3,618.0	2,641.6	347.7
1NXB	2.61E+41	11.1	5.5	5.7	4.9
1TEN	6.17E+43	113.0	23.6	20.6	8.6
1POH	8.02E+43	77.9	19.9	16.6	17.3
2DRI	1.16E+47	-	-	-	-
1FNA	3.02E+47	3,310	856.9	925.0	121.8
1UBI	2.43E+49	-	1,876.8	1,720.6	1,070.2
1C9O	3.77E+49	2,310	455.4	258.2	223.2
1CTF	3.95E+51	-	2,381.1	-	382.1
2PCY	2.34E+52	2,080	12.4	10.2	9.4
1DKT	3.94E+58	5,420	206.7	190.2	202.3
2TRX	9.02E+59	487.0	-	4,916.9	138.2
1CM1	3.73E+63	-	-	-	-
1BRS	1.67E+64	-	-	-	-
1CDL	5.68E+65	-	-	-	-
1LZ1	1.04E+72	-	1,523.2	3,369.3	382.8
1GVP	1.51E+78	-	-	-	-
1RIS	1.23E+80	-	-	-	-
2RN2	3.68E+80	-	2,159.6	6,464.4	247.0
1CSE	8.35E+82	367.0	24.9	25.4	23.0
1HNG	3.70E+88	5,590	-	-	1,712.2
3CHY	2.36E+92	-	-	-	-
1L63	2.17E+94	-	521.1	479.7	510.0
# Nb of cases solved in 900 sec		11	16	15	21
# Nb of cases solved in 9000 sec		17	21	21	23

A '-' indicates an exceeded memory size (128G) or computation time (9000 sec).

Results issued from [21]

While Best-first search is known to explore less node than Depth First search when the same heuristic is used (variables and values), DFBB has the opportunity to improve its upper bound k and therefore its heuristic while searching. This probably explains the performance gain of depth first equipped with the CFN lower bound. The polynomial space use of DFBB is also likely to help in the context of CPU processors using multiple levels of increasingly slow caches.

Enumeration

Sequence-conformations within $2\,kcal.mol^{-1}$ of the GMEC were enumerated for 1SHF, the only case for which the enumeration was completed by DEE/$A*$ with a run time of $1.20e+05$ [20]. We performed both Binary branching and Binary-AA-Branch with DFBB. Best-First and Best-First-DL were also assessed. In the latter case, a simple Binary branching was performed. In all these experiments, the variable ordering *dom/cmed* and the increasing unary cost value ordering were used.

Respectively, Binary branching, Binary-AA-Branch, Best-First and Best-First-DL took 25.07, 23.21, 20.92 and 21.21 sec to enumerate all the conformations. Thus, from these settings, the CFN-based approaches gave performances approaching 10^4 speedup compared to $A*$. Of note, the CFN-based Best-First approaches benefit from the initial LDS at root. This might explain its comparative performance compared to runs based on DFBB. However, in contrast to the GMEC search, Best-First-DL did not perform better than a simple Best-First. This is somehow unsurprising to us because the gain in the upper bound update given by Best-First-DL during GMEC search is not useful during the enumeration phase.

CONCLUSION

This work reports the use of Cost Function Network optimization techniques to solve Computational Protein Design problems. Compared to prior work[19]–[21], it introduces novel search heuristics in order to speedup search methods, which were here implemented in a CPD-dedicated software called *osprey*. The presented search algorithms have shown to be more efficient than optimization methods based on the DEE/A^* framework for both GMEC search and suboptimal solutions enumeration (which is performed in order to account for inaccuracies and approximations made in the CPD modeling). The implementation of these new methods in *osprey* enables novel opportunities for their use in combination with other functionalities already existing in this software, in particular for molecular flexibility modeling.

Indeed, any macromolecular flexibility method that can be represented as an optimization problem with a single matrix and an upper bound can be performed with the presented CFN-based methods. This includes the recently introduced provable DEEPer modeling[23], [40] that handles designs with discrete and continuous flexibility of side chains and backbone. Moreover, the speedup reached in the enumeration of conformations made in the current study, due to CFN modeling, may accelerate the ensemble-based CPD simulations, such as the K^* algorithm implemented in *osprey* [1].

ACKNOWLEDGEMENTS

This work has been funded by the "AgenceNationale de la Recherche", references ANR 10-BLA-0214 and ANR-12-MONU-0015-03. We thank the Computing Center of Region Midi-Pyrénées (CALMIP, Toulouse, France) and the GenoToul Bioinformatics Platform of INRA-Toulouse for providing computing resources and support. S. Traoré was supported by a grant from the INRA and the Region Midi-Pyrénées.

REFERENCES

[1] R. H. Lilien, B. W. Stevens, A. C. Anderson, B. R. Donald, *J. Comput. Biol.*, **2005**, DOI:10.1089/cmb.2005.12.740.
[2] B. Stevens, R. Lilien, I. Georgiev, *Biochemistry*, **2006**.
[3] C. Chen, I. Georgiev, A. C. Anderson, B. R. Donald, **2009**, *106*.

[4] K. M. Frey, I. Georgiev, B. R. Donald, A. C. Anderson, *Proc. Natl. Acad. Sci. U. S. A.*, **2010**, DOI:10.1073/pnas.1002162107.

[5] K. E. Roberts, P. R. Cushing, P. Boisguerin, D. R. Madden, B. R. Donald, **2011**, 361–376.

[6] N. P. King, J. B. Bale, W. Sheffler, D. E. McNamara, S. Gonen, T. Gonen, T. O. Yeates, D. Baker, *Nature*, **2014**, DOI:10.1038/nature13404.

[7] I. Grunwald, K. Rischka, S. M. Kast, T. Scheibel, H. Bargel, *Philos. Trans. A. Math. Phys. Eng. Sci.*, **2009**, DOI:10.1098/rsta.2009.0012.

[8] B. M. Nestl, B. A. Nebel, B. Hauer, *Curr. Opin. Chem. Biol.*, **2011**, DOI:10.1016/j.cbpa.2010.11.019.

[9] J. Pleiss, *Curr. Opin. Biotechnol.*, **2011**, DOI:10.1016/j.copbio.2011.03.004.

[10] N. A. Pierce, E. Winfree, *Protein Eng.*, **2002**, *15*, 779–782.

[11] L. Wernisch, S. Hery, S. J. Wodak, *J. Mol. Biol.*, **2000**, DOI:10.1006/jmbi.2000.3984.

[12] A. Leaver-Fay, B. Kuhlman, J. Snoeyink, *Pac. Symp. Biocomput.*, **2005**, 16–27.

[13] C. L. Kingsford, B. Chazelle, M. Singh, *Bioinformatics*, **2005**, DOI:10.1093/bioinformatics/bti144.

[14] C. a Voigt, D. B. Gordon, S. L. Mayo, *J. Mol. Biol.*, **2000**, DOI:10.1006/jmbi.2000.3758.

[15] J. Desmet, M. De Maeyer, B. Hazes, I. Lasters, *Nature*, **1992**, *356*, 539–542.

[16] R. F. Goldstein, *Biophys. J.*, **1994**, DOI:10.1016/S0006-3495(94)80923-3.

[17] N. A. Pierce, J. A. Spriet, J. Desmet, S. L. Mayo, P. E. T. Al, *J. Comput. Chem.*, **2000**, *21*, 999–1009.

[18] D. B. Gordon, G. K. Hom, S. L. Mayo, N. a Pierce, *J. Comput. Chem.*, **2003**, DOI:10.1002/jcc.10121.

[19] D. Allouche, S. Traoré, I. André, S. de Givry, G. Katsirelos, S. Barbe, T. Schiex, in Proc.\ of CP-12; 2012Quebec City, Canada, 2012.

[20] S. Traoré, D. Allouche, I. André, S. de Givry, G. Katsirelos, T. Schiex, S. Barbe, *Bioinformatics*, **2013**, DOI:10.1093/bioinformatics/btt374.

[21] D. Allouche, I. André, S. Barbe, J. Davies, S. de Givry, G. Katsirelos, B. O'Sullivan, S. Prestwich, T. Schiex, S. Traoré, *Artif. Intell.*, **2014**, DOI:10.1016/j.artint.2014.03.005.

[22] P. Gainza, K. E. Roberts, I. Georgiev, R. H. Lilien, D. A. Keedy, C.-Y. Chen, F. Reza, A. C. Anderson, D. C. Richardson, J. S. Richardson, others, *Methods Enzym.*, **2012**.

[23] P. Gainza, K. E. Roberts, B. R. Donald, *PLoS Comput. Biol.*, **2012**, DOI:10.1371/journal.pcbi.1002335.

[24] T. Schiex, H. Fargier, G. Verfaillie, *Int. Jt. Conf. Artif. Intell.*, **1995**, *14*, 631–639.

[25] D. Allouche, S. Traoré, I. André, S. de Givry, G. Katsirelos, S. Barbe, T. Schiex, in Proc.\ of CP-12; 2012Quebec City, Canada, 2012.

[26] M. Cooper, T. Schiex, *Artif. Intell.*, **2004**, DOI:10.1016/j.artint.2003.09.002.

[27] J. Larrosa, T. Schiex, *Artif. Intell.*, **2004**, DOI:10.1016/j.artint.2004.05.004.

[28] M. Cooper, S. de Givry, T. Schiex, in 8th International CP-06 Workshop on Preferences and Soft Constraints; 2006Nantes, France, 2006.

[29] C. Lecoutre, O. Roussel, D. E. Dehani, in Principles and Practice of Constraint Programming; 20122012.

[30] S. de Givry, S. Prestwich, B. O'Sullivan, in Principles and Practice of Constraint Programming--CP 2013; 2013, Springer, Ed.; 2013.

[31] D. Mitchell, ... *Pract. Constraint Program. 2003*, **2003**.

[32] R. M. Haralick, G. L. Elliot, *Artif. Intell.*, **1980**, *14*, 263–313.

[33] C. Bessière, J.-C. Régin, in Proc. of the Second International Conference on Principles and Practice of Constraint Programming; 1996Cambridge (MA), 1996.

[34] C. Lecoutre, L. Sais, S. Tabary, V. Vidal, in ECAI 2006: 17th European Conference on Artificial Intelligence, August 29-September 1, 2006, Riva del Garda, Italy: including Prestigious Applications of Intelligent Systems (PAIS 2006): proceedings; 20062006.

[35] C. Lecoutre, L. Sa"is, S. Tabary, V. Vidal, *Artif. Intell.*, **2009**, *173*, 1592,1614.

[36] J. Larrosa, T. Schiex, *Int. Jt. Conf. Artif. Intell.*, **2003**, *18*, 239–244.

[37] W. D. Harvey, M. L. Ginsberg, in Proc.\ of the 14^{th} IJCAI; 1995Montr{é}al, Canada, 1995.

[38] A. R. Leach, A. P. Lemon, *Proteins*, **1998**, *33*, 227–239.

[39] M. A. Hallen, D. A. Keedy, B. R. Donald, *Proteins*, **2013**, DOI:10.1002/prot.24150.

[40] M. a Hallen, D. A. Keedy, B. R. Donald, *Proteins Struct. Funct. Bioinforma.*, **2013**, DOI:10.1002/prot.24150.

3.5 REFERENCES

[1] D. Allouche, S. Traoré, I. André, S. de Givry, G. Katsirelos, S. Barbe, and T. Schiex, "Computational Protein Design as a Cost Function Network Optimization Problem," in *Proc.\ of CP-12*, 2012.

[2] D. Allouche, I. André, S. Barbe, J. Davies, S. de Givry, G. Katsirelos, B. O'Sullivan, S. Prestwich, T. Schiex, and S. Traoré, "Computational protein design as an optimization problem," *Artif. Intell.*, vol. 212, pp. 59–79, Mar. 2014.

[3] B. I. Dahiyat, C. a Sarisky, and S. L. Mayo, "De novo protein design: towards fully automated sequence selection.," *J. Mol. Biol.*, vol. 273, no. 4, pp. 789–796, 1997.

[4] B. Stevens, R. Lilien, and I. Georgiev, "Redesigning the PheA domain of gramicidin synthetase leads to a new understanding of the enzyme's mechanism and selectivity," *Biochemistry*, 2006.

[5] N. Koga, R. Tatsumi-Koga, G. Liu, R. Xiao, T. B. Acton, G. T. Montelione, and D. Baker, "Principles for designing ideal protein structures.," *Nature*, vol. 491, no. 7423, pp. 222–7, Nov. 2012.

[6] S. Traoré, D. Allouche, I. André, S. de Givry, G. Katsirelos, T. Schiex, and S. Barbe, "A new framework for computational protein design through cost function network optimization.," *Bioinformatics*, vol. 29, no. 17, pp. 2129–36, 2013.

4 Concluding Remarks & Outlook

Contents

4.1 Outline of the main results and their implications 153
4.2 Addressing the many energetically equivalent conformation issues 154
4.3 Improving the CPD input model(s) 155
4.4 Toward Multiple Objective Optimization & experimental applications 155
4.5 References 158

In this chapter, we present the main conclusions drawn from the work carried out during this thesis and we outline the current limitations of the CPD approaches along with directions for future work.

4.1 Outline of the main results and their implications

Along this thesis, we mainly focused on the development and the assessment of algorithmic strategies in order to efficiently solve Computational Protein Design problems. From a computational point of view, state-of-the-art Combinatorial Optimization techniques have been found highly effective in handling the underlying optimization problem of CPD. This led us to propose novel frameworks dedicated to CPD and based on Cost Function Networks (CFN).

To the best of our knowledge, this was the first time these methods were used in the CPD field. The obtained results demonstrate the efficiency of CFN techniques to accomplish the optimization phase on a variety of instances (varying in their nature and their combinatorial size). Compared to complete deterministic methods commonly used in CPD, such as the DEE/A^*, CFN-based approaches solved CPD problems faster by several orders of magnitude and enabled the identification of both optimal solutions (GMEC) and ensembles of sub-optimal solutions [1], [2]. Moreover, these CFN-based methods turned out to be more efficient than several other solvers issued from recent research in artificial intelligence [3]. These encouraging results led us to integrate these methods into a CPD-dedicated software, *osprey*, developed by Donald's group. The incorporation of these methods into this software aimed at facilitating the combined utilization of these methods with the flexibility models and methods developed for CPD by Donald's group to better account for protein inherent dynamics. In particular, side chain and backbone flexibility models resulting in a single matrix and an upper bound are straightforward to solve by the CFN methodologies. Flexible backbone and Backrub motions only need to perform a pre-computation of the backrub angle values of variable positions before calculating the matrix. This just results in a change of the minimizer used during matrix computation and energy evaluation and thus, does not affect the optimization step. Our current integrated CFN-based framework within the *osprey* package already benefits from these features. In the near future, other types of perturbations already present in the last version of *osprey (*i.e. DEEPer [4]) could also be utilized in combination with CFN optimization to handle larger conformational changes.

In CPD experiments, the matrix computation often includes minimization. Since the individual terms of the matrix are calculated and minimized separately, they provide a lower bound of the corresponding terms calculated in the context of the complete conformation which is not available during the matrix calculation phase. Along the computation (and minimization) of each element of the matrix, the side chain of only one or two positions is present, the other positions are missing at this stage. Let's refer to the introduced pairwise approximation error $\epsilon\ kcal.mol^{-1}$. If n is the number of designable positions, this corresponds to a total error of $\epsilon \frac{n(n+1)}{2}$ for each complete conformation. For example, if $n = 10$ and $\epsilon = 0.5$ (which is an optimistic hypothesis, this value can easily reach more than $1\ kcal.mol^{-1}$ per designable position[5]), the total pairwise approximation error is $27.5\ kcal.mol^{-1}$. To compensate for this error, optimization can be addressed in two different ways. First, the associated optimization problem can be considered as a

continuous optimization problem [5]. In this case, all solutions within a window of at least $27.5\ kcal.mol^{-1}$ must be exhaustively enumerated. These solutions are then all individually minimized to find the minimized optimum or a suboptimal set. This solution is used by exact deterministic approaches to preserve the minimized optimum. A large amount of solutions can however be discarded by this approach if their minimized energies fall outside the error interval (plus the enumeration window) afterwards. Although very accurate (i.e. it provides the true minimized GMEC), this approach aiming at correcting the pairwise approximation error can be very expensive in terms of solutions to enumerate and minimize (vainly for most of them). An alternative is to address the problem as a discrete optimization problem and attempt to correct the pairwise approximation errors. Correction factors can be applied to the minimized matrix so as to reduce the error [6] as well as some *ad-hoc* modeling scheme using side chain carbon pseudo atoms to mimic missing side chains during matrix computation [7].

Other types of approximations are made in order to estimate some properties. The binding constant is estimated by means of populations of rotameric conformations for both the bound and unbound states. In this context, ensemble-based methods such as the hybrid approach DEE/A^*/K^* include: *i)* a sequence prediction step followed by *ii)* a step consisting in the ranking of sequences using the corresponding K^* score (which approximate the binding constant). Indeed, the DEE/A^* algorithm is used in both steps. In the first step, it is used to enumerate suboptimal ensembles of sequence-conformations and in the second step, it is applied to enumerate the conformation ensemble for a fixed sequence in order to assess their associated K^*score. Hence, the effectiveness of these steps is highly dependent on the speed of the algorithm to enumerate the suboptimal ensembles. Thus, the performance gain achieved with the CFN framework compared to the DEE/A^* algorithm should be transposable also into this context.

4.2 **Addressing the many energetically equivalent conformation issues**

The primary objective of CPD is to produce a rank ordered list of mutants that satisfies a given design target with the aim of guiding the experimental construction of a set of sequences. Often, a certain interval within which all solutions are enumerated is defined to set the upper bound used during the CPD optimization with exact deterministic methods [5], [8], [9]. Such enumeration, and also metaheuristics, can lead to a very large number of conformations while not giving any guaranty regarding the diversity of the generated sequences. This diversity depends on the energy distribution of the sequences near the GMEC (for exact deterministic methods and near local minima for metaheuristics). Thus, the optimization may spend a large part of the runtime to enumerate a variety of conformations for a single sequence. Top energy sequences will actually provide a large number of acceptable conformations [2] and take most of the runtime while, in contrast, some other sequences are ignored during the CPD enumeration if their score is slightly outside the defined interval while they could become acceptable solutions upon refinement. Nevertheless, generating one single conformation per sequence would be sufficient in a first step of filtering, and subsequent refinements could handle conformational variability and entropic effects (and Boltzmann ensembles [10], [11]). If only one conformation per sequence is enumerated in the first place, we can focus on the enumeration of a larger number of sequences for further refinement and more accurate sequence scoring and ranking afterwards. Some approaches have been proposed to address the sequence redundancy problem of near-optimal solutions by intro-

ducing a constraint to forbid already found sequences during the optimization [12]. However, this information has to be propagated during the search in order to skip multiple conformations of a single sequence.

An alternative to this strategy that we are currently exploring in our laboratory is the *a priori* enforcement of the sequence diversity by partitioning the sequence space using amino acid similarities. Suppose a partitioning of the amino acids into two groups (the method applies of course for any grouping): *apolar* on one side and the remaining amino acids on the other side. The allowed amino acids at a mutable position can be split into two groups respecting the above partitioning rules. If we have n mutable residues, that corresponds to 2^n independent similarity-based partitions. Of course, flexible positions do not affect the number of partitions. With a small number of mutable residues like in core or active site design, the sub-problems can easily be solved in parallel on a small number of clusters *cpus*. For $n = 10$, we get 1024 partitions, with 32 *cpus* and 10 sec for solving a GMEC (using *toulbar2* for example to solve them [1], [2]). Solving the 1024 independent GMECs in parallel will take around 320 sec. The GMEC of each partition will represent the best solutions of the partitions. A partition here is a group of similar sequences (depending on the amino acid type grouping). The amino acid grouping could be finer but a large number of groups may lead to a huge number of partitions: k^n, where k is the number of amino acid groups per position. Interestingly, this optimization reduces to the classic CPD optimization when the number of groups is one per position (leading to only one sequence partition). Independent suboptimal enumeration can also be carried out for each sequence similarity-based partition in order to overcome approximation inaccuracies. In order to limit the number of partitions, the partitioning positions can be limited to a subset of mutable residues. Of note, the approach does not completely suppress the sequence redundancy but can strongly reduce it.

4.3 Improving the CPD input model(s)

Often, CPD starts from a 3D model derived from crystallographic coordinates. However, crystallographic structures are static and obtained under non-physiological conditions, what can affect the quality of CPD predictions. Therefore, to compensate for this drawback, molecular dynamics simulations can be carried out at room temperature and pressure on the protein initially derived from crystallographic structure in order to generate a relaxed 3D model which is more realistic under biologically active conditions. This equilibrated model extracted from molecular dynamics can then be used as starting point for CPD studies instead of the crystallographic structure. Additionally, several equilibrated models can also be used in order to perform multiple backbone design [13]. Another complementary option is to use the side-chain conformational variability observed in these ensembles of models to enrich rotamer libraries on the fly.

4.4 Toward multiple objective optimization & experimental applications

The computational design of molecular recognition, whether in the case of a protein-ligand, protein-protein or protein-nucleic acid interactions, aims at introducing mutations in the protein to optimize interactions at the interface (or create *de novo* a new interface) while preserving the foldability of the protein (and the stability of the complex). Hence, the nature of the associated

optimization problem is inherently a multiple criteria optimization problem, also known as Multiple Objective Optimization (MOO) problem. Simply put, MOO involves the development of an aggregate objective function (also called utility function) which combines a set of different objectives. The resulting problem is then solved by a single objective solver where the objective function is the aggregate function. Efficient solvers are available to handle single objective optimization problems. However, the effective conception of an aggregate function for multiple criteria problems is an unsolved issue in the CPD field (and others) [14]–[16]. The Weighted Sum approach is commonly used because of its simplicity. The Weighted Sum utility function reduces the multiple objective problems to a single objective optimization problem by associating a scalar weight to each objective.

The elementary task of interest is thus to minimize the aggregate function. In MOO, the initial optimization problem is to find a solution that simultaneously minimizes all the objectives, regardless of the weight. Such a solution is often called the *utopia point*, because it generally does not exist in the design space [17]. This requirement is thus softened by rather seeking the so-called Pareto Set or Pareto Front. The Pareto Set is composed of solutions that are not dominated by another solution with respect to one objective without an increase in at least one other objective. All the elements of the Pareto Set are mathematically equivalent.

A well-studied issue in the MOO field arises from the fact that the involved objective functions can have very different magnitudes. This is well known to affect the accuracy of MOO and referred to as magnitude problem or dimensionality problem [17]. Thus, many forms of mathematical transformation functions have been introduced (in the MOO field) in order to address this pitfall and make the problem dimensionless while preserving the meaning of optimal solutions and elements of the Pareto Set. In an ongoing work, we are addressing this issue by designing and assessing CPD dedicated transformation functions that suppress the dimensionality between the affinity and stability objective functions.

Another fundamental element that strongly controls the effectiveness of the MOO is the formulation of the aggregate function itself. Although the Weighted Sum approach is widespread in the CPD field, its performance is highly dependent on the correct choice and variation of weights which is not straightforward for an arbitrary MOO problem [17]. In addition, it is known to be unsuited to non-convex MOO problems. An attractive approach which does not suffer from these drawbacks is the so-called Physical Programming [17]–[19]. This approach does not use any weighting and its superiority over Weighted Sum has been extensively shown in the MOO field along with its capacity to completely extract the Pareto Set, given boundary constraints on the objectives. The aggregate function of Physical Programming is also very simple and similar to the Weighted Sum formulation. It associates a so-called Class Function to each objective which serves as individual utility function with additional soft constraint expressed as bounding constraints for each objective. The Class Functions are obtained by the application of transformation functions to the objectives. From a CPD point of view, the additional constraints can be simply modeled by using upper bounds. We are currently designing and assessing such an approach to solve multiple criteria CPD problems.

Finally, to conclude this manuscript, it is worth noting that experimental validation is a necessary step either for real applications or to get insightful feedbacks to improve CPD methodologies. In order to carefully consider those aspects, we have made algorithmic choices that more effectively decouple the different steps involved in CPD, what renders feedbacks easier to interpret and exploit. Because provable algorithms make no error with respect to the input model, feedbacks from experiments will greatly help to improve the input model (including both design space representation and selection of objective functions). Hence, the computational developments presented in this manuscript might be useful in the near future for predicting sequences leading to proteins and enzymes of interest for practical applications.

4.5 REFERENCES

[1] D. Allouche, S. Traoré, I. André, S. de Givry, G. Katsirelos, S. Barbe, and T. Schiex, "Computational Protein Design as a Cost Function Network Optimization Problem," in *Proc.\ of CP-12*, 2012.

[2] S. Traoré, D. Allouche, I. André, S. de Givry, G. Katsirelos, T. Schiex, and S. Barbe, "A new framework for computational protein design through cost function network optimization.," *Bioinformatics*, vol. 29, no. 17, pp. 2129–36, 2013.

[3] D. Allouche, I. André, S. Barbe, J. Davies, S. de Givry, G. Katsirelos, B. O'Sullivan, S. Prestwich, T. Schiex, and S. Traoré, "Computational protein design as an optimization problem," *Artif. Intell.*, vol. 212, pp. 59–79, Mar. 2014.

[4] M. A. Hallen, D. A. Keedy, and B. R. Donald, "Dead-end elimination with perturbations (DEEPer): a provable protein design algorithm with continuous sidechain and backbone flexibility.," *Proteins*, vol. 81, no. 1, pp. 18–39, Jan. 2013.

[5] P. Gainza, K. E. Roberts, and B. R. Donald, "Protein design using continuous rotamers," *PLoS Comput. Biol.*, vol. 8, no. 1, p. e1002335, Jan. 2012.

[6] T. Gaillard and T. Simonson, "Pairwise decomposition of an MMGBSA energy function for computational protein design.," *J. Comput. Chem.*, May 2014.

[7] N. Pokala and T. T. M. Handel, "Energy functions for protein design: adjustment with protein-protein complex affinities, models for the unfolded state, and negative design of solubility and specificity.," *J. Mol. Biol.*, vol. 347, no. 1, pp. 203–277, Mar. 2005.

[8] I. Georgiev, R. H. Lilien, and B. R. Donald, "Improved Pruning algorithms and Divide-and-Conquer strategies for Dead-End Elimination, with application to protein design.," *Bioinformatics*, vol. 22, no. 14, pp. e174–e183, Jul. 2006.

[9] E. Kloppmann, G. M. Ullmann, and T. Becker, "An extended dead-end elimination algorithm to determine gap-free lists of low energy states.," *J. Comput. Chem.*, vol. 28, no. 14, pp. 2325–2335, 2007.

[10] I. Georgiev, R. H. Lilien, and B. R. Donald, "The Minimized Dead-End Elimination Criterion and Its Application to Protein Redesign in a Hybrid Scoring and Search Algorithm for Computing Partition Functions over Molecular Ensembles," *Bioinformatics*, vol. 22, no. 14, pp. 530–545, Jul. 2006.

[11] R. H. Lilien, B. W. Stevens, A. C. Anderson, and B. R. Donald, "A novel ensemble-based scoring and search algorithm for protein redesign and its application to modify the substrate specificity of the gramicidin synthetase a phenylalanine adenylation enzyme.," *J. Comput. Biol.*, vol. 12, no. 6, pp. 740–61, Jan. 2005.

[12] M. Fromer and C. Yanover, "Accurate prediction for atomic-level protein design and its application in diversifying the near-optimal sequence space.," *Proteins*, vol. 75, no. 3, pp. 682–705, May 2009.

[13] H. Fung, C. Floudas, M. Taylor, L. Zhang, and D. Morikis, "Toward full-sequence de novo protein design with flexible templates for human beta-defensin-2," *Biophys. J.*, vol. 94, no. 2, pp. 584–99, Jan. 2008.

[14] M. Suárez, P. Tortosa, J. Carrera, and A. Jaramillo, "Pareto Optimization in Computational Protein Design with Multiple Objectives," vol. 12, 2008.

[15] M. Suarez, P. Tortosa, M. M. Garcia-Mira, D. Rodríguez-Larrea, R. Godoy-Ruiz, B. Ibarra-Molero, J. M. Sanchez-Ruiz, and A. Jaramillo, "Using multi-objective computational design to extend protein promiscuity.," *Biophys. Chem.*, vol. 147, no. 1–2, pp. 13–9, Mar. 2010.

[16] S. Polydorides, N. Amara, C. Aubard, P. Plateau, T. Simonson, and G. Archontis, "Computational protein design with a generalized Born solvent model: application to Asparaginyl-tRNA synthetase.," *Proteins*, vol. 79, no. 12, pp. 3448–68, Dec. 2011.

[17] R. T. Marler and J. S. Arora, "Survey of multi-objective optimization methods for engineering," *Structural and Multidisciplinary Optimization*, vol. 26. pp. 369–395, 2004.

[18] A. Messac, C. P. Sukam, and E. Melachrinoudis, "Mathematical and pragmatic perspectives of physical programming," *AIAA Journal*, vol. 39. pp. 885–893, 2001.

[19] A. Messac and A. Ismail-Yahaya, "Multiobjective robust design using physical programming," *Struct. Multidiscip. Optim.*, vol. 23, pp. 357–371, 2002.

5 Résumé Long en Français

Contents

5.1 Introduction 161
5.2 Stratégies de Design Computationnel de Protéines 162
5.3 De nouvelles approches basées sur les réseaux de fonctions de coût 166
5.4 Conclusion et perspectives 169
5.5 Références 171

5.1 Introduction

Le design computationnel de protéines, en anglais « Computational Protein Design » (CPD), vise à remodeler des protéines existantes afin de les doter de propriétés physico-chimiques et fonctionnelles améliorées ou nouvelles tout en préservant le repliement de la protéine de départ. Durant ces dernières décennies, le CPD s'est avéré être un champ de recherche très prometteur au vu des différents succès de conception de protéines aux propriétés désirées qu'il a permis d'accomplir [1]–[10]. Il s'agit d'une approche complémentaire des méthodes expérimentales permettant de guider et d'accélérer l'ingénierie des protéines et ainsi, de réduire les couts humains et financiers. En guise d'illustration, pour une protéine de taille moyenne, de 300 résidus d'acides aminés, avec à chaque position, l'introduction possible de chacun des 20 acides aminés naturels, l'espace théorique de séquences aurait une taille de 20^{300}. Une inspection exhaustive de cet espace pour identifier les combinaisons de mutations les plus bénéfiques pour la propriété recherchée est ainsi hors de portée des approches expérimentales. Seulement une infime partie de cet espace pourrait être effectivement exploré expérimentalement. Le CPD intervient alors pour rationaliser cette sélection.

Dans ce contexte, nous nous sommes intéressés au développement de nouvelles méthodes de CPD pour faciliter la conception de nouvelles enzymes. Le CPD s'appuie sur trois composantes majeures. La première concerne la modélisation du système moléculaire à traiter, incluant notamment le choix de la protéine de départ et des degrés de liberté associés. La deuxième repose sur l'utilisation de fonctions mathématiques adéquates (fonctions objectives) capables d'évaluer virtuellement les propriétés cibles. La troisième composante fait appel à des méthodes d'optimisation combinatoires efficaces pour explorer et filtrer ces larges espaces de séquences-conformations. Nous résumerons succinctement ces étapes dans la Section 5.2. Le cycle de CPD est souvent suivi par une étape ultérieure d'analyse d'un sous-ensemble des résultats les plus prometteurs en faisant appel à des méthodes de modélisation plus fines mais aussi plus couteuses en temps de calcul.

Malgré les avancées méthodologiques réalisées ces dernières années dans le domaine du CPD, de nombreux défis restent à relever pour améliorer la fiabilité de prédiction de ces approches et étendre leurs domaines d'applications. En particulier, l'un des obstacles majeurs rencontrés par le CPD est la taille exorbitante de l'espace combinatoire à explorer. Dans le cadre de cette thèse, nous avons recherché d'autres solutions calculatoires permettant un filtrage plus efficace de ces vastes espaces de recherche. C'est pourquoi, nous nous sommes intéressés aux dernières avancées méthodologiques réalisées dans le domaine de l'intelligence artificielle. Ainsi, en collaboration avec l'équipe de Thomas Schiex du MIAT-INRA Toulouse, nous avons adapté et évalué différents algorithmes d'optimisation combinatoire pour traiter les espaces combinatoires complexes du CPD. Ces nouvelles approches, basées sur l'optimisation de réseaux de fonctions de coûts (CFN) se sont avérées bien plus efficaces pour traiter des problèmes difficiles de design de protéines par rapport aux méthodes habituellement utilisées en CPD [11]–[13]. Sur la base de ces nouvelles approches d'optimisation combinatoire, nous avons proposé un nouveau Framework dédié au CPD [12]. De plus, nous avons également implémenté ces méthodes dans un outil de référence de CPD, *osprey* [14] pour qu'à terme ces nouvelles approches puissent être couplées

aux techniques récentes de traitement de la flexibilité moléculaire, développées par le groupe de B. Donald à l'Université de Duke-USA (où j'ai effectué un séjour de 3 mois durant ma thèse).

5.2 **Stratégies de Design Computationnel de Protéines**

La modélisation du problème de CPD se décompose en trois étapes : *i)* sélection d'une structure tridimensionnelle (3D) protéique fournissant le châssis moléculaire le mieux adapté aux objectifs de design considérés ; *ii)* définition de l'espace de recherche impliquant le choix des résidus variables et les états (identités et/ou conformations) qu'ils peuvent adopter ; *iii)* définition de fonctions objectives adaptées au problème du design et basées sur des potentiels énergétiques capables de discriminer rapidement et précisément les solutions optimales.

5.2.1 Représentation de l'espace de design

Une des conditions de base du CPD est la préservation du repliement de la protéine de départ. Pour cela, la modification de la conformation du squelette, quand elle est permise, suit un principe minimaliste. De faibles variations peuvent être autorisées en prenant notamment en compte des conformations observées expérimentalement. La flexibilité des chaînes latérales est quant à elle modélisée par des ensembles discrets de conformations majoritairement observées dans la Protein Data Bank (PDB), appelées rotamères. Outre sa variabilité conformationnelle, l'identité du résidu d'acide aminé mutable peut être changée par un des 20 acides aminés naturels ou bien un sous-ensemble. A noter que les acides aminés de type proline, glycine et cystéine sont souvent exclus pour préserver les éléments de structures secondaires.

5.2.2 Fonctions d'énergie et fonctions objectives

Par un souci de simplification, les termes d'énergie utilisés en CPD sont approximés pour être additifs par paires de rotamères, ce qui permet un pré-calcul et stockage des énergies d'interaction. Ainsi, l'échantillonnage plusieurs fois du même rotamère ou paires de rotamères ne nécessite pas de recalculer les contributions individuelles. Cette représentation accélère les méthodes heuristiques d'optimisation et est une condition requise pour les approches d'optimisation exactes déterministes (brièvement introduites ci-après). Durant la phase d'optimisation, la matrice sera donc lue pour trouver la combinaison optimale d'affectation des rotamères aux positions variables. Ainsi, l'énergie totale, dénommée E_{total}, d'une conformation de la protéine est donnée par la formule suivante:

$$E_{\text{total}} = E_c + \sum_i E(i_r) + \sum_{i,j} E(i_r, j_s) \qquad \textbf{(5-1)}$$

E_c est l'énergie d'interaction entre les parties fixes du modèle (elle n'est pas utilisée dans l'optimisation), $E\ (i_r)$ est un terme d'énergie dépendant du rotamère r à la position i et $E\ (i_r, j_s)$ est l'énergie d'interaction de paire entre le rotamère r à la position i et le rotamère s à position j.

Les interactions entre les résidus de la protéine, ou avec leurs partenaires (ligands, cofacteurs ou autres protéines ou des acides nucléiques, par exemple) sont estimées à l'aide de deux types de fonctions d'énergie empiriques. Il peut s'agir de fonctions basées sur les principes de la physique qui sont en général précises mais souvent lourdes en terme de temps de calcul ou de fonctions basées sur la connaissance (statistique) qui sont généralement plus rapides. Ainsi, les fonctions utilisées en CPD peuvent inclure celles issues : *i)* des champs de force de Mécanique Moléculaire (MM), *ii)* de l'analyse statistique de données expérimentales ou bien *iii)* une combinaison des deux.

Ainsi, les termes énergétiques individuels incluent généralement les interactions atomiques de « packing » tels que les angles dièdres et le Lennard-Jones. Les contributions de liaison et d'angle de liaison prises en compte en MM ne sont habituellement pas utilisées en CPD car leur géométrie idéale est souvent utilisée. Les interactions électrostatiques et les liaisons hydrogènes sont également considérées. Par ailleurs, le solvant est généralement représenté par un modèle implicite car sa modélisation explicite serait trop couteuse. D'autres termes *ad hoc* peuvent également être utilisés, par exemple pour estimer l'entropie des chaînes latérales ou la propension des éléments de structure secondaire.

Outre la définition précise de la fonction d'énergie au niveau atomique, la phase d'optimisation nécessite, pour un design donné, la définition d'une ou plusieurs fonctions objectives permettant d'établir la correspondance entre l'espace de séquence-conformation et le paysage de la fonction de fitness. Plus précisément, les propriétés physico-chimiques (tels que la stabilité thermodynamique, le repliement et la solubilité) et les propriétés fonctionnelles (telles que la liaison à une petite molécule ou une autre macromolécule) doivent être modélisées comme des fonctions objectives à optimiser pour obtenir une protéine stable et fonctionnelle. À noter que, lors de l'optimisation, plusieurs objectifs peuvent être agrégés en une seule fonction (autrement appelée optimisation à objectifs multiples). Le design négatif peut également être utilisé pour optimiser simultanément une fonction de fitness qui tient compte de deux ou plusieurs états concurrents où l'un des états doit être discriminé. En guise d'exemple pour le design de stabilité, une définition de l'état déplié est requise. La fonction objective sera donc une différence entre l'énergie totale et l'énergie de l'état déplié.

5.2.3 Optimisation combinatoire

5.2.3.1 Les approches habituelles en CPD

Sur la base de ces fonctions objectives, des algorithmes d'optimisation combinatoire sont utilisés pour rechercher la conformation du minimum global d'énergie, autrement appelée en anglais «Global Minimum Energy Conformation» (GMEC), ou un ensemble de solutions de basses énergies. Du point de vue algorithmique, le problème de CPD est NP-difficile [15]. De plus, le problème de placement de chaînes latérales s'est avéré NP-complet à approximer [16]. Pour cette raison, et au vu de la dimension élevée de l'espace de séquence-conformation, des méthodes heuristiques ont été largement développées pour résoudre les problèmes d'optimisation du CPD. La première famille de méthodes de cette classe est constituée souvent de variantes du Monte Carlo (MC) [17] ou de l'Algorithme Génétique (GA) [18], [19]. Il s'agit d'approches stochastiques,

basées sur des choix locaux aléatoires au cours de cycles d'optimisation. Ainsi, deux cycles heuristiques indépendants peuvent donner des résultats différents, correspondants à des minima locaux distincts. Par conséquent, bien que ces approches ne garantissent pas l'identification du GMEC en un seul cycle heuristique, leur consommation raisonnable en CPU permet de réaliser des centaines de milliers de cycles heuristiques afin de converger vers le GMEC. Elles sont donc appropriées pour traiter des problèmes complexes de CPD. Cependant, le nombre de cycles, notamment pour le problème d'énumération d'ensembles sous-optimaux, semble être un point critique pour les approches heuristiques [17]. Par conséquent, une analyse systématique de la convergence de ces méthodes doit être effectuée pour produire des résultats significatifs. Ainsi, malgré leurs avantages en termes de temps de calcul, ces méthodes ne garantissent pas l'identification de l'optimum de la fonction objective. Ainsi, lorsque les résultats expérimentaux et les expériences *in silico* diffèrent, il est difficile de savoir si l'écart est lié à un échantillonnage insuffisant par ces méthodes heuristiques ou à des imprécisions dans la modélisation du problème. De plus, ces approches étant incapables de reconnaître le GMEC même si celui-ci a été énuméré, de nombreux cycles supplémentaires peuvent être effectués inutilement, ce qui tend à dégrader la vitesse de ces méthodes heuristiques pour des problèmes de grandes dimensions [20].

Pour pallier à ces limitations et faciliter les cycles itératifs entre optimisation *in silico* et essais expérimentaux, les méthodes déterministes présentent un intérêt majeur. L'algorithme de ce type le plus couramment utilisé repose sur le théorème de «Dead-End Elimination » (DEE) [21] en combinaison avec le A^* afin d'extraire la (les) solution(s) de l'espace restant après élagage par le DEE [17], [22]. Des approches de programmation linéaire en nombres entiers (ILP) et de programmation dynamique [23] ont été également appliquées au CPD [24] avec des performances encourageantes. Des simulations de CPD ont été également réalisées à l'aide de méthodes basées sur la théorie du champ moyen [25]–[28]. Cependant, bien que déterministes puisque deux simulations conduisent aux mêmes résultats, ces méthodes basées sur le champ moyen ne sont néanmoins pas complètes dans la mesure où l'identification du GMEC n'est pas garantie.

L'approche de DEE élimine un rotamère dans une position tout en s'assurant qu'il ne peut exister au sein de la solution optimale. Des critères similaires existent également pour éliminer des paires de rotamères ou des combinaisons de rotamères d'ordres supérieurs. Ces critères d'élimination sont appliqués jusqu'à convergence ou jusqu'à un nombre prédéfini d'étapes. Toutefois, étant donné que le DEE ne converge pas toujours vers une solution unique lorsqu'il est confronté à des problèmes de CPD complexes, un algorithme d'énumération tel que A^* est ensuite appliqué pour extraire le GMEC de l'espace restant. Deux exécutions de ce type de combinaison d'algorithmes donnent le même résultat, et fournissent la meilleure solution mathématique, le GMEC, lorsqu'elles convergent. De plus, des solutions sous-optimales peuvent être énumérées dans une fenêtre d'énergie spécifiée par l'utilisateur. Cependant, la convergence n'est pas assurée et ces méthodes peuvent être extrêmement consommatrices de CPU. Ainsi, elles peuvent échouer pour traiter des problèmes de CPD complexes.

Dès lors, de nouvelles approches déterministes exactes et complètes sont nécessaires afin de surpasser les limitations des méthodes actuelles et faire face à des designs de haute complexité combinatoire.

5.2.3.2 Une nouvelle approche pour le CPD : les réseaux de fonction de cout

Un réseau de contraintes est un modèle mathématique où un ensemble de contraintes est défini sur un ensemble de variables discrètes. Chaque contrainte limite les valeurs autorisées pour une ou un sous-ensemble de variables. Le Problème de Satisfaction de Contraintes (CSP) est de trouver simultanément une valeur pour chacune des variables de manière à satisfaire à toutes les contraintes (également appelée une solution). Le CSP est NP-complet [29]. Un réseau de fonction de coût (CFN pour «Cost Function Network») élargit le cadre des réseaux de contraintes en remplaçant les contraintes avec des fonctions de coût [30], [31]. Dans un CFN, nous avons un ensemble de variables avec chacun un domaine fini associé et un ensemble de fonctions de coût locales (c'est-à-dire impliquant uniquement un sous-ensemble de toutes les variables). Le problème de satisfaction de contrainte pondéré (WCSP pour « Weighted CSP ») est de trouver une valeur pour toutes les variables qui minimise la somme de toutes les fonctions de coût. Les CFNs ont été utilisés comme un outil de modélisation pour représenter et résoudre des problèmes d'optimisation combinatoire dans de nombreux domaines, incluant la bioinformatique et l'affectation des ressources [32]–[34].

Formellement, un CFN P est un triplet $P = (X, D, C)$ avec $X = \{1,2,3,\ldots,n\}$ un ensemble de n variables. Chaque variable $i \in X$ possède un domaine discret $D_i \in$ D. Dans le triplet, C est un ensemble de fonctions de coût locales. Chaque fonction de coût $c_S \in C$ est définie sur un sous-ensemble de variables $S \subseteq X$ (appelé sa portée), a comme domaine $\prod_{i \in S} D_i$ et prend ses valeurs dans $\mathbb{N} \cup \{+\infty\}$. Les valeurs, paires et affectations jointes d'ordre supérieures sont représentées par des coûts infinis appelés contraintes dures et toutes les fonctions de coût doivent être non négatives. Souvent, dans la pratique, nous connaissons une « bonne » solution de coût k, rendant toutes les solutions de coût supérieure à k inintéressantes. Toutes ces solutions de coût au-delà de k peuvent ainsi être considérées comme infini. Le coût d'une affectation A est la somme des coûts de toutes les fonctions de coût locales. Il est généralement supposé que C contienne une fonction de coût constante, avec une portée vide, notée $c_\emptyset$. Un CFN P définit une distribution des coûts joints sur toutes les variables X définies par le coût des affectations. Étant donné que toutes les fonctions de coût dans un CFN sont non négatives, la fonction de coût constante $c_\emptyset \in C$définit un minorant sur cette distribution.

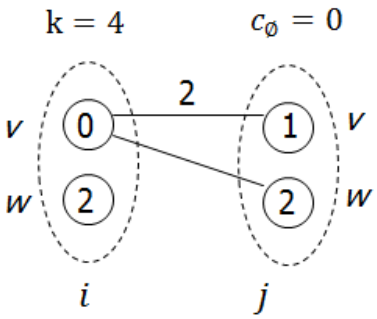


Fig 5-1 Un exemple de CFN

La **Fig 5-1** est ce qu'on appelle une représentation en microstructure de CFN sous forme de graphe sur un exemple très simple. Les deux variables impliqués $\{i, j\}$ sont représentées par des cercles en pointillés. Les valeurs de domaine sont représentées par de petits cercles (sommets, $D_x = D_y = \{v, w\}$). Les étiquettes des sommets sont les coûts unaires correspondants. Les arêtes représentent des termes binaires, l'étiquette correspondante est son coût. Pour plus de clarté, les arêtes qui ont un coût nul ne sont pas représentées et l'étiquette n'est pas écrite quand le coût = 1. Le majorant initial du problème est 4.

Le problème de CPD se modélise assez naturellement sous la forme d'un réseau de fonctions de coût. En effet, chaque résidu variable du CPD est représenté par une variable i du CFN, son ensemble de rotamères permis étant modélisé par D_i. Le terme constant dans E_{total} peut être capturé par $c_\emptyset$, les termes $E(i_r)$ sont modélisés par les fonctions locales d'arité unaire et les termes $E(i_r, j_s)$ sont capturés par les fonctions locales d'arité binaire. On peut ainsi modéliser le problème d'optimisation du CPD comme un WCSP binaire. Pour faire cette correspondance entre WCSP et CPD, il est nécessaire d'appliquer une transformation des énergies (des nombres réels), en nombres naturels. Pour ce faire, le minimum de la matrice d'énergie est retranché à tous les termes de la matrice pour avoir des réels positifs, suivi de leur multiplication par une puissance de 10 pour avoir un nombre voulu de chiffres après la virgule. L'optimum du WCSP ainsi formulé est le GMEC du problème de CPD.

Le problème de WCSP est généralement résolu par des algorithmes de recherche en profondeur d'abord avec séparation-évaluation (« Depth-First Branch and Bound »). Ils intègrent à chaque nœud, des méthodes incrémentales de cohérence d'arc qui maintiennent en temps polynomial un minorant fort sur le problème d'optimisation [35], [36]. La maintenance de ces propriétés de cohérence d'arc, en plus d'incrémenter le minorant, permet d'élaguer l'arbre de recherche en supprimant des rotamères incompatibles avec au moins une contrainte. Ce majorant est initialement infini et est mis à jour à chaque fois qu'une conformation complète est identifiée. La recherche est terminée quand le minorant devient supérieur ou égal au majorant

5.3 **De nouvelles approches basées sur les réseaux de fonctions de coût (CFN)**

Les travaux de ma thèse furent principalement consacrés à l'évaluation de nouvelles méthodes d'optimisation exactes déterministes pour traiter des tailles de combinatoire aussi larges que celles explorées par les méthodes heuristiques. En collaboration avec l'équipe MIAT-INRA, spécialisée dans l'optimisation combinatoire, nous avons adapté pour la première fois des approches basées sur les réseaux de fonctions de coût (CFN) au problème de CPD et évalué leurs performances sur plusieurs cas de design de protéines.

Dans un premier temps, ces approches basées sur le CFN ont été évaluées sur un jeu de 12 cas distincts de design de protéines. Elles se sont avérées bien plus efficaces en terme de temps de calcul pour identifier le GMEC que les approches habituellement utilisées en CPD et basées sur le DEE/A^*. De par l'accélération du calcul (par plusieurs ordres de grandeur), elles ont permis de trouver la solution optimale pour de nombreux cas non résolus par les autres méthodes. Ces travaux sont décrits dans un article publié dans les proceedings du « 18th International Conference on Principles and Practice of Constraint Programming » (Québec, Canada, October, 8-12 2012) [11].

Sur la base de ces résultats encourageants, nous avons alors évalué les performances d'autres méthodes d'optimisation combinatoire (0/1 Linear Programming 0/1 Quadratic Programming, 0/1 Quadratic optimization, Weighted Partial Max SAT and Graphical Model Optimization) sur un jeu plus large de cas de CPD (40 au lieu de 12). Ces travaux ont montré que les méthodes CFN

sont bien plus efficaces que toutes les autres approches testées (**Fig 5-2**). Ces résultats sont présentés dans un article publié dans *Artificial Intelligence* [13].

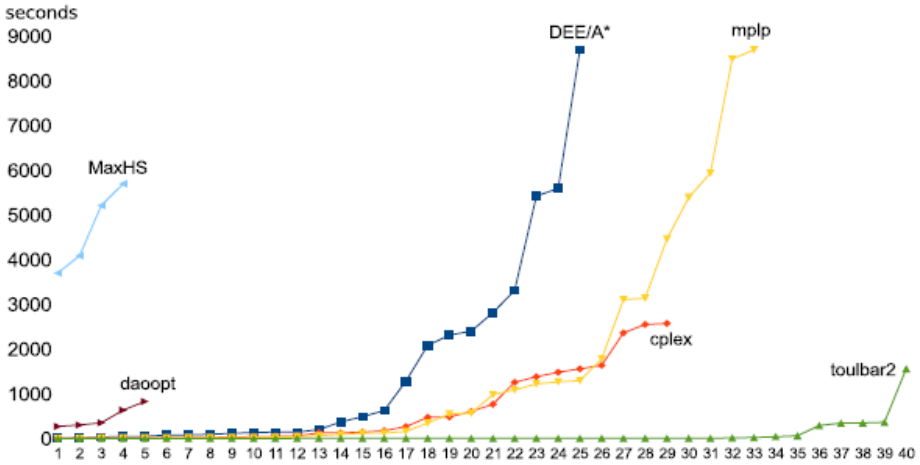


Fig 5-2 Evalution de différentes méthodes d'optimisation combinatoire (solveurs *toulbar2 (CFN), osprey (DEE/A*, cplex (0/1 Linear Programming), MaxHS (Weighted Partial Mas SAT), mplp et daoopt (Graphical Model Optimization))* pour résoudre différents problèmes de design de protéines. Axe X : Nombre de problèmes résolus ; Axe Y : Temps CPU alloué pour résoudre chaque problème [13].

Au vu des performances de ces approches basées sur le CFN pour résoudre le problème d'identification du GMEC, nous avons alors appliqué ces méthodes pour traiter un autre problème du CPD qui est l'énumération d'un ensemble de solutions sous-optimales dans un intervalle défini au-delà de l'optimum. Un intervalle de 2 $kcal.mol^{-1}$ a été utilisé dans notre études. Nous avons imposé une limite de temps de 100h et une limite de mémoire de 128G. Sur 35 cas de designs (**Tableau 5-1**), les méthodes basées sur le CFN ont réussi à énumérer les solutions sous-optimales pour 30 cas (en 7 heures de calcul pour le plus long) alors que le DEE/A^* n'a pu les énumérer que pour un seul cas. L'approche DEE/A^* a ainsi échoué pour 34 cas soit à cause des limites de temps (30 cas) ou de mémoire (4 cas). Bien que la seule instance qu'elle ait résolue ((1SHF) corresponde à l'un des plus petits espaces combinatoires étudiées ($\sim 10^{34}$), ~ 37 heures de calcul ont quand même été nécessaires pour trouver l'ensemble des modèles de basse énergie. La même instance a été résolue par le CFN en moins d'une seconde. Plus encore, le temps de calcul le plus long pour le CFN fut observé pour 1L63 qui a requis ~7h (pour un espace combinatoire de $\sim 10^{94}$) pour énumérer un grand nombre de solutions sous-optimales ($8\ x\ 10^8$).

Au cours de cette étude, nous avons également introduit de nouveaux critères de choix de l'espace de mutations, basés sur la mesure de l'enfouissement des résidus. Alors que de nombreux travaux de CPD mesurent l'aire de la surface exposée au solvant pour évaluer la localisation des résidus mutables dans la structure 3D et ainsi définir les acides aminés autorisés à ces positions [37]–[39], notre approche repose quant à elle sur une mesure de l'enfouissement des résidus plus précise : le rayon de solvatation des résidus [40]. Dans le cadre de ces travaux, nous avons alors proposé un nouveau Framework permettant de réaliser un calcul de CPD complet, allant de la modélisation du problème jusqu'à l'optimisation combinatoire basée sur le CFN, en combinant l'utilisation de deux outils : un logiciel dédié au CPD, *osprey* (développé à l'université de Duke, USA, équipe de B. Donald) et le solveur CFN, *toulbar2* (développé au MIAT-INRA,

équipe de T. Schiex). L'ensemble de ces travaux a donné lieu à un article publié dans *Bioinformatics* [12].

Tableau 5-1 Temps CPU pour l'identification du GMEC en utilisant DEE/A* (*osprey*), ILP (*cplex*) and CFN (*toulbar2*) [12]**.**

PDB	*Taille espace de Séquence-Conformation*	*Temps (s)*		
		DEE/A*	ILP	CFN
1MJC	4.36e+26	4.57.	3.94	0.08
1CSP	5.02e+30	200.00	360.00	0.84
1BK2	1.18e+32	93.20	303.00	0.65
1SHG	2.13e+32	138.00	42.30	0.25
1CSK	4.09e+32	41.70	24.90	0.15
1SHF	1.05e+34	44.30	11.10	0.17
1FYN	5.04e+36	622.00	2.26e+03	3.79
1PIN	5.32e+39	9.54e+03	3.00e+03	3.99
1NXB	2.61e+41	11.10	21.20	0.24
1TEN	6.17e+43	113.00	81.70	0.33
1POH	8.02e+43	77.90	31.80	0.45
2DRI	1.16e+47	T_{A*}	2.92e+5	24.5
1FNA	3.02e+47	3.31e+03	419.00	0.73
1UBI	2.43e+49	T_{A*}	704.00	2.14
1C9O	3.77e+49	2.31e+03	1.40e+03	2.20
1CTF	3.95e+51	T_{A*}	580.40	1.23
2PCY	2.34e+52	2.08e+03	76.70	0.51
1DKT	3.94e+58	5.42e+03	1.85e+03	3.95
2TRX	9.02e+59	487.00	1.34e+03	1.7
1PGB	5.10e+61	T_{DEE}	T	T
1CM1	3.73e+63	T_{A*}	2.65e+04	19.2
1BRS	1.67e+64	T_{A*}	2.39e+05	426.0
1ENH	6.65e+64	T_{DEE}	T	T
1CDL	5.68e+65	T_{A*}	T	191.1
1LZ1	1.04e+72	T_{A*}	1.25e+03	2.11
2CI2	7.26e+75	T_{DEE}	T	T
1GVP	1.51e+78	T_{A*}	T	593.6
1RIS	1.23e+80	T_{A*}	T	501.00
2RN2	3.68e+80	M_{A*}	1.14e+03	2.77
1CSE	8.35e+82	367.00	205.93	1.36
1HNG	3.70e+88	5.59e+03	4.15e+03	6.97
3CHY	2.36e+92	T_{A*}	2.91e+04	171.00
1L63	2.17e+94	M_{A*}	2.82e+03	6.41
3HHR	2.98e+171	T_{DEE}	M	T
1STN	2.00e+249	T_{DEE}	M	M
# Nb de cas résolus en 10 min		11	13	30
# Nb de cas résolus en 100 h		18	27	30

Un 'M' indique que la limite de mémoire a été dépassée (128G) et un 'T' indique que la limite de temps de calcul est dépassée (100h). Pour l'approche DEE / A *, le A * et le DEE notés en indice de M ou T indiquent l'étape au cours de laquelle a eu lieu le dépassement de la limite de temps de calcul ou de mémoire.

Enfin, nous avons aussi implémenté les approches basées sur le CFN directement dans le logiciel *osprey* afin de faciliter leur accès à la communauté scientifique. Ces travaux ont été réalisés pendant le séjour de trois mois que j'ai réalisé au sein du groupe de B. Donald (Université de Duke, Caroline du Nord, États-Unis). Une autre motivation pour intégrer les méthodes CFN dans ce logiciel était de bénéficier des développements déjà réalisés par le groupe de B. Donald, notamment pour le traitement de la flexibilité moléculaire. De plus, nous avons proposé de nouvelles avancées méthodologiques dérivées des approches de CFN ainsi que de nouvelles heuristiques d'ordonnancement de variables et de valeurs pour améliorer encore davantage les performances de la phase d'optimisation combinatoire. Ces travaux sont décrits dans ce manuscrit et un article sera très prochainement soumis à *Journal of Computational Chemistry.*

5.4 **Conclusion et perspectives**

Dans le cadre de cette thèse, nos travaux ont été centrés sur le problème de l'optimisation combinatoire en CPD. Différentes méthodes dérivées de la recherche en Intelligence Artificielle ont été adaptées et évaluées sur des problèmes variés de design de protéines. Ainsi, un nouveau Framework complètement dédié au CPD et basé sur des méthodes d'optimisation combinatoire jamais mises en œuvre auparavant dans le domaine du CPD a pu être proposé.

L'adaptation au CPD des méthodes basées sur les réseaux de fonctions de coût a conduit à une accélération de l'optimisation de la combinatoire de plusieurs ordres de grandeurs par rapport aux approches existantes dédiées au CPD. En effet, leur efficacité a pu être éprouvée à la fois sur le problème d'obtention du GMEC et sur le problème d'énumération d'ensembles de solutions sous-optimales. Elles se sont avérées efficaces pour la résolution exacte du problème d'optimisation du CPD (recherche du GMEC), ainsi que pour l'énumération d'ensembles de modèles proches de l'optimum. Alors que dans de nombreux cas de design, l'approche DEE/A^* a échoué dans l'identification du GMEC ou dans l'énumération des modèles sous-optimaux, notre Framework basé sur le CFN s'est avéré capable quant à lui de résoudre ces problèmes avec des temps de calcul extrêmement courts sur monoprocesseur, allant de moins d'une seconde à quelques minutes pour l'identification du GMEC à quelques heures pour l'énumération d'un ensemble de modèles sous-optimaux (dans un intervalle de $2\ kcal.mol^{-1}$ du GMEC). Ainsi, nos travaux ont conduit à proposer un Framework performant pour le CPD basé sur des méthodes originales issues d'intelligence artificielle.

En ce qui concerne la suite de ce travail, plusieurs pistes d'amélioration de notre méthodologie de CPD sont envisagées. Tout d'abord une tendance en CPD est d'améliorer la prise en compte de la flexibilité moléculaire, notamment en introduisant des rotamères continus et/ou des minimisations des rotamères rigides durant le calcul de matrices [41]. Certaines approches couplent cela avec une optimisation « continue » de manière à couvrir l'optimum de l'espace continu, ce qui a pour conséquence d'améliorer les prédictions [41]. Cependant, ces approches restent très couteuses en temps CPU et sont donc limitées à de petites tailles de combinatoires. Une alternative plus réaliste serait de continuer à faire de l'optimisation discrète mais avec des rotamères continus et des minimisations tout en essayant d'améliorer conjointement la concordance entre les valeurs d'énergie totale issues des conformations complètes et celles issues de la discrétisation de la matrice.

Par ailleurs, un problème majeur dans l'énumération d'ensemble de solutions sous-optimales est la redondance de conformations pour la même séquence et la présence de solutions très proches énergétiquement sans que cela n'augmente la diversité de séquences. Pour pallier à ce problème, nous avons entamé une réflexion qui vise à structurer l'espace des mutations de manière à forcer la diversité de séquences au détriment de la redondance de conformations. En effet, l'objectif premier n'est pas une étude de l'entropie conformationnelle mais bien l'identification de diverses séquences sous-optimales. La redondance est donc utile mais seulement dans un second temps avec un échantillonnage plus poussé que ce qui est faisable pendant l'identification des séquences.

Quand il s'agit de réaliser le design d'interfaces protéine-protéine, protéine-ligand ou bien enzyme-substrat, une tendance est de considérer non pas une seule conformation par mutant mais un ensemble thermodynamique de conformations, appelé ensemble de Boltzmann [42]. Ces approches effectuent un classement des mutants sur la base d'une évaluation de l'affinité de liaison entre les deux molécules en faisant appel à des méthodes classiques de calcul d'énergie libre de liaison ou bien à de nouvelles méthodes, appelées K^* [42]. Ces approches ont été appliquées avec succès pour des designs d'interactions protéine-protéine et enzyme-substrat [38], [42]–[45]. Cependant, l'efficacité de ces méthodes peut être limitée par le filtrage préalable de l'espace de mutations qui peut élaguer des mutants alors qu'ils auraient pu être classés en tête de liste par le K^* ou les calculs d'énergie libre. Le développement de méthode d'optimisation à objectifs multiples trouve tout son intérêt dans ce contexte.

Enfin, un autre point d'amélioration possible du CPD réside dans la préparation du modèle tridimensionnel de départ du design. Le CPD utilise habituellement des modèles 3D épurés à partir de coordonnées cristallographiques. Toutefois, ces modèles statiques peuvent être biaisés par les conditions expérimentales (pH, température, pression, solvant, contacts cristallins, ..). Pour contourner ces problèmes, il serait possible d'utiliser comme modèle de départ une ou plusieurs conformations à l'équilibre thermodynamique issus de simulations de dynamique moléculaire dans les conditions voulues (i.e., par exemple à pression, température et pH physiologique). D'autre part, les conformations des chaines latérales observées à l'équilibre pourraient être ajoutées à la volée à la librairie de rotamères en tant que rotamères natifs durant le calcul de matrice (comme c'est déjà souvent le cas en CPD mais sur le modèle structural de départ uniquement).

Pour finir, il est important de noter que l'expérimentation biologique est une étape essentielle du CPD pour acquérir des informations qui permettront d'améliorer les méthodologies développées. Pour faciliter l'intégration des données issues de l'expérimentation, nous avons effectué des choix algorithmiques permettant de découpler l'étape d'optimisation du reste de la procédure de CPD. De cette façon, tout progrès sur l'optimisation reste bien sûr un acquis, et les feedbacks des expériences biologiques devraient permettre de déceler les imperfections dans la formulation du problème et des fonctions objectives utilisées pour mesurer les propriétés recherchées.

5.5 RÉFÉRENCES

[1] P. B. Harbury, J. J. Plecs, B. Tidor, T. Alber, and P. S. Kim, “High-resolution protein design with backbone freedom.,” *Science*, vol. 282, no. 5393, pp. 1462–1467, Nov. 1998.

[2] B. Kuhlman, G. Dantas, G. C. Ireton, G. Varani, B. L. Stoddard, and D. Baker, “Design of a novel globular protein fold with atomic-level accuracy.,” *Science*, vol. 302, no. 5649, pp. 1364–8, Nov. 2003.

[3] L. A. Clark, P. A. Boriack-Sjodin, J. Eldredge, C. Fitch, B. Friedman, K. J. M. Hanf, M. Jarpe, S. F. Liparoto, Y. Li, A. Lugovskoy, S. Miller, M. Rushe, W. Sherman, K. Simon, and H. Van Vlijmen, “Affinity enhancement of an in vivo matured therapeutic antibody using structure-based computational design.,” *Protein Sci.*, vol. 15, no. 5, pp. 949–60, May 2006.

[4] S. J. Fleishman, T. A. Whitehead, D. C. Ekiert, C. Dreyfus, J. E. Corn, E.-M. Strauch, I. A. Wilson, and D. Baker, “Computational design of proteins targeting the conserved stem region of influenza hemagglutinin.,” *Science*, vol. 332, no. 6031, pp. 816–21, May 2011.

[5] H. W. Hellinga and F. M. Richards, “Construction of new ligand binding sites in proteins of known structure. I. Computer-aided modeling of sites with pre-defined geometry.,” *J. Mol. Biol.*, vol. 222, no. 3, pp. 763–85, Dec. 1991.

[6] A. L. Pinto, H. W. Hellinga, and J. P. Caradonna, “Construction of a catalytically active iron superoxide dismutase by rational protein design.,” *Proc. Natl. Acad. Sci. U. S. A.*, vol. 94, no. 11, pp. 5562–7, May 1997.

[7] L. L. Looger, M. a Dwyer, J. J. Smith, and H. W. Hellinga, “Computational design of receptor and sensor proteins with novel functions.,” *Nature*, vol. 423, no. 6936, pp. 185–190, 2003.

[8] D. Röthlisberger, O. Khersonsky, A. M. Wollacott, L. Jiang, J. DeChancie, J. Betker, J. L. Gallaher, E. a Althoff, A. Zanghellini, O. Dym, S. Albeck, K. N. Houk, D. S. Tawfik, and D. Baker, “Kemp elimination catalysts by computational enzyme design.,” *Nature*, vol. 453, no. 7192, pp. 190–5, May 2008.

[9] L. Jiang, E. a Althoff, F. R. Clemente, L. Doyle, D. Röthlisberger, A. Zanghellini, J. L. Gallaher, J. L. Betker, F. Tanaka, C. F. B. Iii, D. Hilvert, K. N. Houk, B. L. Stoddard, D. Baker, and C. F. Barbas, “De novo computational design of retro-aldol enzymes.,” *Science*, vol. 319, no. 5868, pp. 203–277, Mar. 2008.

[10] J. B. Siegel, A. Zanghellini, H. M. Lovick, G. Kiss, A. R. Lambert, J. L. St Clair, J. L. Gallaher, D. Hilvert, M. H. Gelb, B. L. Stoddard, K. N. Houk, F. E. Michael, and D. Baker, “Computational design of an enzyme catalyst for a stereoselective bimolecular Diels-Alder reaction.,” *Science*, vol. 329, no. 5989, pp. 309–13, Jul. 2010.

[11] D. Allouche, S. Traoré, I. André, S. de Givry, G. Katsirelos, S. Barbe, and T. Schiex, “Computational Protein Design as a Cost Function Network Optimization Problem,” in *Proc.\ of CP-12*, 2012.

[12] S. Traoré, D. Allouche, I. André, S. de Givry, G. Katsirelos, T. Schiex, and S. Barbe, “A new framework for computational protein design through cost function network optimization.,” *Bioinformatics*, vol. 29, no. 17, pp. 2129–36, 2013.

[13] D. Allouche, I. André, S. Barbe, J. Davies, S. de Givry, G. Katsirelos, B. O’Sullivan, S. Prestwich, T. Schiex, and S. Traoré, “Computational protein design as an optimization problem,” *Artif. Intell.*, vol. 212, pp. 59–79, Mar. 2014.

[14] P. Gainza, K. E. Roberts, I. Georgiev, R. H. Lilien, D. A. Keedy, C.-Y. Chen, F. Reza, A. C. Anderson, D. C. Richardson, J. S. Richardson, and others, “OSPREY: Protein design with ensembles, flexibility, and provable algorithms,” *Methods Enzym.*, 2012.

[15] N. A. Pierce and E. Winfree, “Protein design is NP-hard.,” *Protein Eng.*, vol. 15, no. 10, pp. 779–782, Oct. 2002.

[16] B. Chazelle, C. Kingsford, and M. Singh, “A Semidefinite Programming Approach to Side Chain Positioning with New Rounding Strategies,” *INFORMS J. Comput.*, vol. 16, no. 4, pp. 380–392, Nov. 2004.

[17] L. Wernisch, S. Hery, and S. J. Wodak, “Automatic protein design with all atom force-fields by exact and heuristic optimization.,” *J. Mol. Biol.*, vol. 301, no. 3, pp. 713–36, Aug. 2000.

[18] D. T. Jones, “De novo protein design using pairwise potentials and a genetic algorithm.,” *Protein Sci.*, vol. 3, no. 4, pp. 567–74, Apr. 1994.

[19] J. Desjarlais and T. Handel, “De novo design of the hydrophobic cores of proteins,” *Protein Sci.*, vol. 4, no. 10, pp. 2006–18, Oct. 1995.

[20] C. a Voigt, D. B. Gordon, and S. L. Mayo, “Trading accuracy for speed: A quantitative comparison of search algorithms in protein sequence design.,” *J. Mol. Biol.*, vol. 299, no. 3, pp. 789–803, Jun. 2000.

[21] J. Desmet, M. De Maeyer, B. Hazes, and I. Lasters, “The dead-end elimination theorem and its use in protein side-chain positioning.,” *Nature*, vol. 356, no. 6369, pp. 539–542, 1992.

[22] A. R. Leach and A. P. Lemon, “Exploring the conformational space of protein side chains using dead-end elimination and the A* algorithm.,” *Proteins*, vol. 33, no. 2, pp. 227–239, 1998.

[23] A. Leaver-Fay, B. Kuhlman, and J. Snoeyink, “An adaptive dynamic programming algorithm for the side chain placement problem.,” *Pac. Symp. Biocomput.*, pp. 16–27, 2005.

[24] C. L. Kingsford, B. Chazelle, and M. Singh, “Solving and analyzing side-chain positioning problems using linear and integer programming.,” *Bioinformatics*, vol. 21, no. 7, pp. 1028–1036, Apr. 2005.

[25] A. Roitberg and R. Elber, "Modeling side chains in peptides and proteins: Application of the locally enhanced sampling and the simulated annealing methods to find minimum energy conformations," *J. Chem. Phys.*, vol. 95, no. 12, p. 9277, 1991.

[26] H. Kono and J. Doi, "Energy minimization method using automata network for sequence and side-chain conformation prediction from given backbone geometry.," *Proteins*, vol. 19, no. 3, pp. 244–55, Jul. 1994.

[27] M. Vásquez, "Modeling side-chain conformation," *Curr. Opin. Struct. Biol.*, vol. 6, no. 2, pp. 217–221, Apr. 1996.

[28] M. Delarue and P. Koehl, "The inverse protein folding problem: self consistent mean field optimisation of a structure specific mutation matrix.," *Pac. Symp. Biocomput.*, pp. 109–21, Jan. 1997.

[29] J. Larrosa and T. Schiex, "Solving weighted CSP by maintaining arc consistency," *Artif. Intell.*, vol. 159, no. 1–2, pp. 1–26, 2004.

[30] T. Schiex, H. Fargier, and G. Verfaillie, "Valued constraint satisfaction problems: Hard and easy problems," *Int. Jt. Conf. Artif. Intell.*, vol. 14, pp. 631–639, 1995.

[31] S. Bistarelli, U. Montanari, and F. Rossi, "Semiring based constraint solving and Optimization," *J. ACM*, vol. 44, no. 2, pp. 201–236, 1997.

[32] B. Cabon, S. De Givry, L. Lobjois, T. Schiex, and J. P. Warners, "Radio link frequency assignment," *Constraints*, vol. 4, no. 1, pp. 79–89, 1999.

[33] S. de Givry, Z. Vitezica, I. Palhiere, and T. Schiex, "MendelSoft: Mendelian error detection in complex pedigree using weighted constraint satisfaction techniques," in *8th World Congress on Genetics Applied to Livestock Production*, 2006, p. 2p.

[34] M. Zytnicki, C. Gaspin, and T. Schiex, "DARN! A weighted constraint solver for RNA motif localization," *Constraints*, vol. 13, no. 1, pp. 91–109, 2008.

[35] T. Schiex, "Arc consistency for soft constraints," *Princ. Pract. Constraint Program. 2000*, pp. 411–425, 2000.

[36] J. Larrosa and T. Schiex, "Solving weighted CSP by maintaining arc consistency," *Artif. Intell.*, vol. 159, no. 1, pp. 1–26, 2004.

[37] B. I. Dahiyat, C. a Sarisky, and S. L. Mayo, "De novo protein design: towards fully automated sequence selection.," *J. Mol. Biol.*, vol. 273, no. 4, pp. 789–796, 1997.

[38] B. Stevens, R. Lilien, and I. Georgiev, "Redesigning the PheA domain of gramicidin synthetase leads to a new understanding of the enzyme's mechanism and selectivity," *Biochemistry*, 2006.

[39] N. Koga, R. Tatsumi-Koga, G. Liu, R. Xiao, T. B. Acton, G. T. Montelione, and D. Baker, "Principles for designing ideal protein structures.," *Nature*, vol. 491, no. 7423, pp. 222–7, Nov. 2012.

[40] G. Archontis and T. Simonson, "A residue-pairwise generalized born scheme suitable for protein design calculations.," *J. Phys. Chem. B*, vol. 109, no. 47, pp. 22667–73, Dec. 2005.

[41] P. Gainza, K. E. Roberts, and B. R. Donald, "Protein design using continuous rotamers," *PLoS Comput. Biol.*, vol. 8, no. 1, p. e1002335, Jan. 2012.

[42] R. H. Lilien, B. W. Stevens, A. C. Anderson, and B. R. Donald, "A novel ensemble-based scoring and search algorithm for protein redesign and its application to modify the substrate specificity of the gramicidin synthetase a phenylalanine adenylation enzyme.," *J. Comput. Biol.*, vol. 12, no. 6, pp. 740–61, Jan. 2005.

[43] C. Chen, I. Georgiev, A. C. Anderson, and B. R. Donald, "Computational structure-based redesign of enzyme activity," vol. 106, no. 10, 2009.

[44] K. M. Frey, I. Georgiev, B. R. Donald, and A. C. Anderson, "Predicting resistance mutations using protein design algorithms.," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 107, no. 31, pp. 13707–12, Aug. 2010.

[45] K. E. Roberts, P. R. Cushing, P. Boisguerin, D. R. Madden, and B. R. Donald, "Design of protein-protein interactions with a novel ensemble-based scoring algorithm," pp. 361–376, Mar. 2011.

REMERCIEMENTS

Je tiens tout d'abord à exprimer ma gratitude envers Isabelle ANDRE et Sophie BARBE pour avoir encadré ma thèse. Je les remercie aussi pour leur patience et leur soutien durant ces trois années. Leur esprit de collaboration scientifique m'a donné la chance d'accéder à de nouvelles connaissances aussi bien avec elles au sein du LISBP-INSA qu'à travers les nombreux partenaires scientifiques avec lesquelles j'ai eu l'opportunité de travailler. Je remercie beaucoup Magali REMAUD-SIMEON pour son accueil sympathique et son soutien. Mes remerciements vont aussi à l'endroit de l'INRA et la région Midi-Pyrénées pour avoir financé cette thèse.

Merci à mes rapporteurs Catherine ETCHEBEST et Raphaël GUEROIS ainsi que l'ensemble des membres du jury pour avoir évalué mon travail ainsi que pour leurs regards critiques et les questions fondamentales dont nous avons eu à discuter.

Une majeure partie de mes travaux de thèse a été effectuée en collaboration avec l'équipe de Thomas SCHIEX (MIAT-INRA Toulouse). Ils ont réussi, grâce à leur dynamisme, à susciter en moi un engouement sans cesse croissant pour leur domaine de recherche très stimulant. Je leur en suis très reconnaissant. Spécialement les échanges avec David, Simon et Thomas m'ont beaucoup permis d'avancer dans la compréhension des réseaux de fonctions de coût. Cela a finalement permis de faciliter mon travail durant mon séjour dans le laboratoire de Bruce DONALD (Université de Duke, Etats-Unis) que je remercie ainsi que l'ensemble de ses étudiants. J'ai bénéficié de leur part un accueil très chaleureux et leur enthousiasme m'a permis de passer un séjour très bénéfique.

J'adresse mes chaleureux remerciements à Christopher TOPHAM pour sa disponibilité et l'éclairage qu'il a pu m'apporter durant ces trois années. Je remercie l'ensemble des membres du projet ProtiCAD (LAAS-CNRS, BIOS-Polytechnique & KINEO) auquel j'ai eu l'opportunité de participer. Cela m'a permis d'avoir un aperçu de la richesse de ce que peuvent apporter les approches de robotique en CPD. Notamment les interactions que j'ai eues avec Juan CORTES, Marc VAISSET et Nicola SIMEON m'ont été très enrichissantes scientifiquement.

Merci à Thomas Simonson et l'ensemble des personnes qui ont participé à mon initiation au CPD durant mes stages de Master (Thomas GAILLARD, David MIGNON, Alexey ALEKSANDROV, Najette AMARA, Priyadarshi SATPATI, ...). C'est ce qui m'a finalement donné envie de poursuivre en thèse dans ce domaine.

Je remercie vivement Agreenium et le département CEPIA de l'INRA d'une part pour avoir pris en charge mon séjour aux Etats-Unis, et d'autre part de façon plus générale pour le dispositif de formations doctorales complémentaires qui a été mis à ma disposition. J'adresse également mes cordiaux remerciements à Monique AXELOS pour son soutien et ses précieux conseils dans le cadre du dispositif de suivi mis en place par l'Ecole Internationale de Recherche Agreenium.

Je remercie énormément le Centre de Calcul de la Région Midi-Pyrénées (CALMIP) et la plateforme Bioinformatique de l'INRA-Toulouse (GenoToul) pour les ressources de calcul qu'ils ont mises à ma disposition ainsi que leur assistance durant ces trois années.

Mes chaleureux remerciements aux amis que je me suis faits dans le laboratoire ainsi que tous mes sympathiques collègues du LISBP. Grâce à la bonne ambiance qu'ils ont réussie à créer, j'ai pu passer d'agréables moments dans et en dehors du laboratoire. Je pense avec nostalgie au rituel du midi, les pauses café, les soirées, les sorties, etc.

Enfin, je rends hommage à ma famille pour les valeurs qu'elle m'a transmises ainsi que pour son soutien inconditionnel en tout temps et en tout lieu. Merci infiniment à tous mes amis de par le monde.

Merci à tous!