HAL
archives-ouvertes.fr

Cost Function Networks to Solve Large Computational Protein Design Problems

David Allouche, Sophie Barbe, Simon de Givry, George Katsirelos, Yahia Lebbah, Samir Loudni, Abdelkader Ouali, Thomas Schiex, David Simoncini, Matthias Zytnicki

▶ To cite this version:

David Allouche, Sophie Barbe, Simon de Givry, George Katsirelos, Yahia Lebbah, et al.. Cost Function Networks to Solve Large Computational Protein Design Problems. Malek Masmoudi, Bassem Jarboui, Patrick Siarry. Operations Research and Simulation in healthcare, Springer, In press. hal-02177634

HAL Id: hal-02177634

https://hal.archives-ouvertes.fr/hal-02177634

Submitted on 9 Jul 2019

Cost Function Networks to Solve Large Computational Protein Design Problems

David Allouche, Sophie Barbe, Simon de Givry, George Katsirelos, Yahia Lebbah, Samir Loudni, Abdelkader Ouali, Thomas Schiex*, David Simoncini, and Matthias Zytnicki

1 Introduction

A protein is a sequence of basic building blocks called *amino acids*. There are 20 natural amino acids. Proteins are involved in nearly all structural, catalytic, sensory, and regulatory functions of living systems [21]. 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.

David Allouche, Simon de Givry, Thomas Schiex*, and Matthias Zytnicki
MIAT, UR-875, INRA, University of Toulouse, F-31320 Castanet Tolosan, France, e-mail: `firstname.lastname@inra.fr`

Sophie Barbe
LISBP, INSA, UMR INRA 792/CNRS 5504, F-31400 Toulouse, France, e-mail: `sophie.barbe@insa-toulouse.fr`

George Katsirelos
UMR MIA-Paris, INRA, AgroParisTech, University of Paris-Saclay, F-75005 Paris, France, e-mail: `gkatsi@gmail.com`

Yahia Lebbah
LITIO, University of Oran 1, 31000 Oran, Algeria, e-mail: `ylebbah@gmail.com`

Samir Loudni
GREYC, UMR 6072-CNRS, University of Caen Normandy, F-14032 Caen, France, e-mail: `samir.loudni@unicaen.fr`

Abdelkader Ouali
Orpailleur team, LORIA, INRIA - Nancy, France, e-mail: `abdelkader.ouali@inria.fr`

David Simoncini
IRIT, UMR 5505-CNRS, University of Toulouse, F-31042 cedex 9, France, e-mail: `david.simoncini@ut-capitole.fr`

* Corresponding author

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 [60]. This holds great interest for medicine, synthetic biology, nanotechnologies and biotechnologies [55, 63, 32]. 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 bio-fuels, chemical feedstocks and therapeutics [38, 8]. 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 [49, 10, 70]. 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 (bio)chemical products [25]. 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* [53].

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 [59]: 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 [67] 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 [39]).

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 [69].

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 ap-

proach relies on dead-end elimination (DEE), a specific form of dominance analysis that was introduced in [19], and later strengthened in [30]. 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 (CFN) and solved as a Weighted Constraint Satisfaction Problem. In this context, DEE is similar to neighborhood substitutability [23, 48, 29].

To evaluate the efficiency of the CFN approach, we model the CPD problem using different combinatorial optimization formalisms. We compare the performance of the 0/1 linear programming solver `cplex`, 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. Each of the 20 amino acids consists 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. In water, most proteins of interest *fold* into a 3D shape that is determined by 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. Anfinsen's postulated [4] that the 3D structure of a protein is entirely defined by its sequence. This structure, defined by the backbone's structure and all side-chain rotations, is called the *conformation* of the protein and determines its chemical reactivity and biological function.
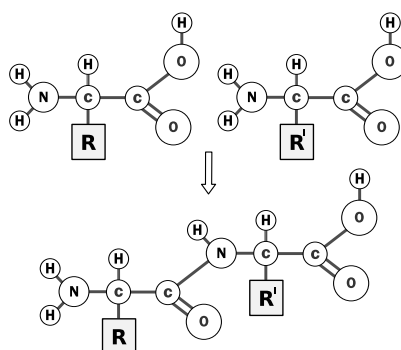

Fig. 1 A representation of how amino acids, carrying specific side chains R and R', can link together through their core to form a chain (modified from wikipedia). One molecule of water is generated in the process.

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 [40, 31]. Therefore, the design problem is usually approached as an inverse folding problem [59], 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 [6]. In structural biology, the stability of a conformation can be directly evaluated through the energy of the conformation, a stable fold being of minimum energy [4].

In CPD, two approximations are common. First, it is assumed that the resulting designed protein retains the overall fold of the chosen scaffold: the protein *backbone* is considered fixed. At specific positions chosen automatically or by the molecular modeler, the amino acid used can be modified, thus changing the *side chain* as shown in Fig. 2. Second, the domain of conformations available to each amino acid side chain is 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* [37], are derived from the most frequent conformations in the experimental repository of known protein structures, the PDB (Protein Data Bank, `www.pdb.org`). Different discretizations have been used in constraint-based approaches to protein structure prediction [7]. More recently, continuous optimization of dihedrals has been addressed in some limited settings [33, 24].

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 [5]. 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. We have used two state-of-the-art energy functions, AMBER9 [9] and Talaris2014 [57], respectively implemented in the CPD dedicated tools `osprey` 2.0 [26] and `Rosetta Molecular Modeling suite` 3 [47].

These energy functions are formulated in such a way that the terms are locally decomposable. Then, the energy of a given protein conformation, defined by a
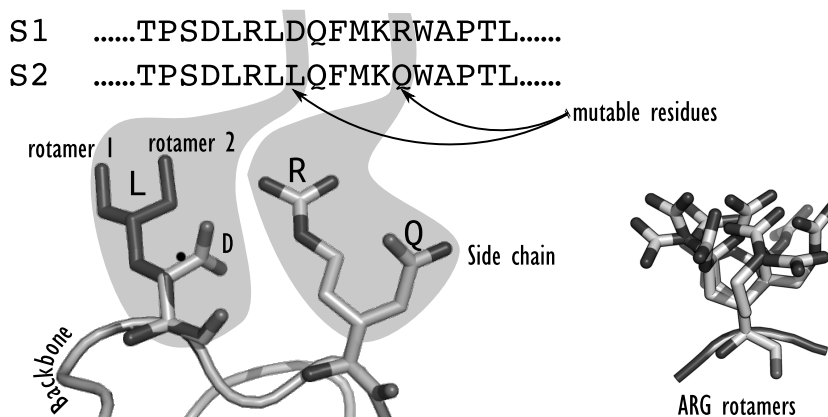
Fig. 2 A local view of the combinatorial sequence exploration considering a common backbone. Changes can be caused by amino acid identity substitutions (*for example D/L* or *R/Q*) or by amino acid side-chain reorientations (rotamers) for a given amino acid. A sparse rotamer library for one amino acid is shown on the right (ARG=Arginine).

choice of one specific amino acid with an associated conformation (rotamer) for each residue, can be written as:

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

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 [19]. 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 discrete rotamer CPD 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 specific 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 [62]. Hence, a variety of meta-heuristics have been applied to it, including Monte Carlo simulated annealing [42], genetic algorithms [64, 12], variable neighborhood search [11], and other algorithms [20]. 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. [71] reported that the accuracy of meta-heuristics also degrades as problem size increases. More importantly, the use of exact search algorithms is important in the usual experimental design cycle, that goes through 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.

Usual exact methods for CPD mainly rely on the dead-end elimination (DEE) theorem [19, 18] and the A^* algorithm [46, 28]. 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 [19]:

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

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 [30] by directly comparing energies of each rotamer in the same conformation:

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

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 [61, 50, 27]. 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 (lower bound) used by A^* is described in [28].

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 ([13]) 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 $w_\emptyset$. All costs being non-negative, the value of this constant function, $w_\emptyset$, 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 famous Constraint Satisfaction Problem or CSP (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 $w_\emptyset = E_\emptyset$, 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 $E_\emptyset$, $E(\cdot)$ 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 energy factors. 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 [43] (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*) [65, 43] 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) [44] and Virtual Arc Consistency (VAC) [15] have also been introduced. See [13] for a review of existing local consistencies.

Table 1 Time and space complexities of enforcing local consistency properties on a binary CFN (X, W), with $n = |X|$ variables, maximum domain size $d = \max_{i \in X} |D_i|$, $e = |W|$ cost functions with maximum forbidden cost k, and minimum non-zero rational cost $\epsilon \in]0, 1]$. Also, we report polynomial classes which are solved by these local consistencies.

Local consistency	Time complexity	Space complexity	Polynomial classes
NC	$O(nd)$	$O(nd)$	-
AC*	$O(n^2 d^2 + ed^3)$	$O(ed)$	-
EDAC	$O(ed^2 \max(nd, k))$	$O(ed)$	*Tree-structures*
VAC$_\epsilon$	$O(ed^2 k / \epsilon)$	$O(ed)$	*Tree-struct., submodular func.*

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 $_\varnothing$ 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 [65, 43, 45, 17, 44, 14, 16, 15, 13]. 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 $E_\varnothing$. Far more complex sequences or sets of EPTs can be required to enforce other local consistencies [13].

Local consistency can be enforced in polynomial time and space as summarized in Table 1. During search, incrementality is preserved along any branch of the search tree, avoiding to repeatedly apply the same EPTs at every search node.

Unfortunately, finding the optimal *sequence* of EPTs with *integer* costs which maximizes $E_\varnothing$ is NP-hard [17]. It is polynomial for a *set* of EPTs with *rational* costs [16] and corresponds to optimization over the *local polytope* of probabilistic graphical models [13]. In practice, during search, applying suboptimal sequences of EPTs by enforcing EDAC or VAC is the most efficient approach for solving CFNs [36].

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 [23]. In the context of CFN, the authors of [48] introduced partial soft neighborhood substitutability with a definition that is equivalent to Equation 3 for pairwise decomposed energies.

DEE can be enforced in time $O(n^2d^3)$ and it is orthogonal to local consistencies, except for VAC where value removals done by DEE cannot break the VAC property [48]. In practice, DEE and AC* (or EDAC) will be enforced until both properties are verified, with time complexity in $O(n^3d^4)$. In order to reduce this time complexity, during search, we do not test every pair of values in every domain D_i, but only one pair (i_r, i_u) such that $E(i_u)$ is minimum and $E(i_r)$ is maximum. Thus, enforcing this restricted DEE[1] can be done in $O(n^2d)$ and it iterates at most $n \times d$ times with AC* [29].

3.3 Exploiting tree decomposition in a hybrid best-first branch-and-bound method

Hybrid Best-First Search (HBFS) [2] explores the search tree in a best-first manner as in A*. However, each selected open search node is expanded by a depth-first search with a limited number of backtracks. When the limit is reached, all the remaining unexplored search nodes are inserted in the open node list. The limit is dynamically tuned in order to reduce the overhead of reconstructing the CFN corresponding to the selected open node.

The HBFS method was further extended in [2] to exploit a tree decomposition, resulting in the BTD-HBFS method.

Definition 2 A tree decomposition of a connected CFN (X, W) is a pair (C_T, T) where $T = (I, A)$ is a tree with nodes set I and edges set A and $C_T = \{C_i \mid i \in I\}$ is a family of subsets of X, called *clusters*, such that: (i) $\cup_{i \in I} C_i = X$, (ii) $\forall w_S \in W$, $\exists C_i \in C_T$ s.t. $S \subseteq C_i$, (iii) $\forall i, j, k \in I$, if j is on the path from i to k in T, then $C_i \cap C_k \subseteq C_j$.

Definition 3 A graph of clusters for a tree decomposition (C_T, T) is an undirected graph $G = (C_T, E)$ that has a vertex for each cluster $C_i \in C_T$, and there is an edge $(C_i, C_j) \in E$ when $C_i \cap C_j \neq \emptyset$.

The width of a tree decomposition corresponds to the size of the largest cluster minus one. As finding an optimal tree decomposition with minimum width, called *treewidth*, is NP–hard, we use fast approximate algorithms like the *min-fill* heuristic.

BTD-HBFS (*Backtracking with Tree Decomposition*, or BTD for short) uses a restricted variable ordering heuristic, which selects variables from a root (largest) cluster first and then continues by assigning variables in the child clusters in a depth-first manner. Each child cluster has one open node list with associated lower and upper bounds per visited assignment of its variables intersecting with its parent cluster. By doing so, it exploits the lower bounds reported by HBFS in individual clusters to improve the anytime behavior and global pruning of BTD. Given a CFN (X, W) with treewidth t, BTD computes the optimum in time $O(knd^{t+1})$ and space $O(knd^{2t})$ [2].

3.4 A parallel variable neighborhood search method guided by tree decomposition

UDGVNS (for Unified Decomposition Guided Variable Neighborhood Search) [58] is a CFN solving method unifying two complete and incomplete search methods: iterative Limited Discrepancy Search (LDS) and Variable Neighborhood Search (VNS) methods. LDS [34] is a heuristic method that explores the depth-first search tree in a non-systematic way by making a limited number of *wrong* decisions w.r.t. its value ordering heuristic. We assume a binary search tree where at each search node either the selected variable is assigned to its chosen preferred value (left branch) or the value is removed from the domain (right branch). Each value removal corresponds to a wrong decision made by the search, it is called a *discrepancy*. The number of discrepancies is limited by a parameter. Iterative LDS increases this parameter (by a multiplication factor of 2) at each iteration until a complete search is done.

VNS/LDS [54, 51] is a metaheuristic that uses a finite set of pre-selected neighborhood structures N_s, $s = 4, 5, ..., n$ to escape from local minima by systematically changing the neighborhood structure if the current one does not improve the current incumbent solution. The initial solution is found by depth-first search on the whole problem. Then, s variables are randomly chosen and the current solution is partially destroyed by un-assigning the selected variables and an exploration of its (large) neighborhood is performed by LDS with a fixed discrepancy. As soon as a better solution is found, then s is reset to its minimal value (4 in our experiments). DGVNS [22] uses another neighborhood structure $N_{s,c}$, where s is the neighborhood size and C_c is the cluster where the variables will be selected from. If $(s > |C_c|)$, we complete the set of candidate variables to be unassigned by adding clusters adjacent to C_c in the graph of clusters provided by a tree decomposition. The neighborhood change in DGVNS is performed in the same way as in VNS/LDS. However, DGVNS

considers successively all the clusters. This ensures a better diversification by covering a large number of different regions. UDGVNS [58] iterates DGVNS with an increasing discrepancy (when $s = n$ and no better solution was found) as in iterative LDS, until a complete search is done, therefore tuning its compromise between optimality proof and anytime behavior.

The parallel version of UDGVNS relies on a master/worker model. The master process controls the communication over the entire processes and holds the centralized information, while the asynchronous worker processes explore the parts of the search space assigned by the master. The cooperation in parallel UDGVNS is achieved by sharing a single global best solution among the worker processes.

4 Integer linear programming for the CPD

The rigid backbone CPD problem has a simple formulation and can be easily written in a variety of combinatorial optimization frameworks. In our previous work [1], we compared the CFN approach to solvers coming from different fields, including probabilistic graphical model, 0/1 linear programming, 0/1 quadratic programming, and 0/1 quadratic optimization. Here, we present only the 01LP model. Other approaches were shown to be far less efficient than CFN or 01LP [1].

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 inequalities and equalities.

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 01LP:

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

$$\text{s.t.} \quad \sum_r d_{ir} = 1 \quad (\forall i) \tag{4}$$

$$\sum_s p_{irjs} = d_{ir} \quad (\forall i, r, j) \tag{5}$$

$$d_{ir} = 0 \quad (\forall i, r) E(i_r) = k \tag{6}$$

$$p_{irjs} = 0 \quad (\forall i, r, j, s) E(i_r, j_s) = k \tag{7}$$

$$d_{ir} \in \{0, 1\} \quad (\forall i, r) \tag{8}$$

$$p_{irjs} \in \{0, 1\} \quad (\forall i, r, j, s) \tag{9}$$

It is known that the continuous LP relaxation of this model is the dual of the LP problem defined by Optimal Soft Arc Consistency (OSAC) [16, 13] 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 VAC. 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.

This model is also the 01LP model IP1 proposed in [41] for side-chain positioning. It has a quadratic number of Boolean variables. Constraints (6) and (7) explicitly 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 (4) and (5) enforce that the p_{irjs} are set to 0 or 1. In the rest of the paper, we relax constraint (9).

5 Computational Protein Design instances

In our initial experiments with CPD in [3], 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 [26] 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 [3] and additionally created 35 extra instances from existing published designs, as described in [68]. We must insist on the fact that the 12 rebuilt instances do not define the same energy landscape or search space as the initial [3]'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 [52].

Preparation of CPD instances

Missing heavy atoms in crystal structures and hydrogen atoms were added with the *tleap* module of the AMBER9 software package [9]. Each molecular system was then minimized in implicit solvent (Generalized Born model [35]) 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 up to $\frac{n.(n-1)}{2}$ binary cost functions that can be computed independently. The resulting instances have between $n = 11$ and $n = 120$ residues, and between $d = 48$ and $d = 198$ rotamers (see Table 2).

Translation to WCSP and 01LP formats

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 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 [68] with no significant difference in computing efforts.

Note that the approximate treewidth produced by min-fill heuristic of WCSP instances is between 4 and 113, with a mean value of 26.68 (after preprocessing by `toulbar2`).

All the Python and C translating scripts used, including translating from WCSP to 01LP format, are available together with the 47 CPD instances in native and WCSP formats at `http://genotoul.toulouse.inra.fr/~tschiex/CPD-AIJ`.

6 Experimental results

For computing the GMEC, all computations were performed on an Intel Xeon E5-2680 at 2.50GHz, 256 GB of RAM (except `osprey` on an AMD Operon 6176 at 2.3 GHz, 128 GB of RAM), and a 9,000-second time-out. We compared three solvers: DEE/A* `osprey` version 2.0 (`cs.duke.edu/donaldlab/osprey.php`), 01LP solver `cplex` version 12.8.0 (with parameters EPAGAP, EPGAP, and EPINT set to zero to avoid premature stop), and CFN solver `toulbar2` version 1.0.1 (`github.com/toulbar2/toulbar2`).

The procedure of `osprey` starts by extensive DEE pre-processing (*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).

We ran `toulbar2` using VAC in preprocessing and HBFS with EDAC, DEE[1], and binary branching during search (options `-d: -A`). The additional exploitation of a min-fill tree decomposition inside branch-and-bound is denoted as `toulbar2` BTD (extra options `-O=-3 -B=1`). The variable neighborhood search guided by tree decomposition is denoted `toulbar2 UDGVNS` (extra options `-O=-3 -vns`).
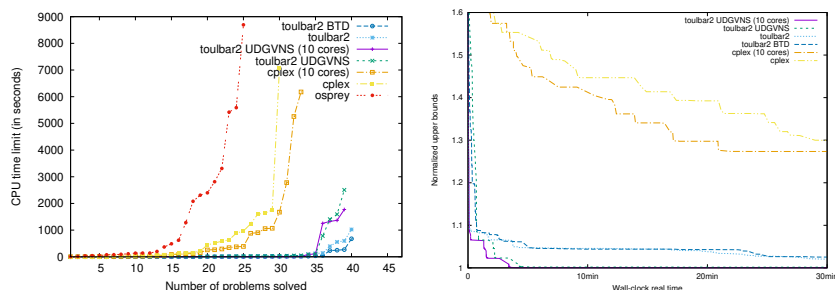
Fig. 3 (Left) A figure showing the number of problems that can be solved by each approach (X-axis) as a function of time allowed for solving each problem (Y-axis). (Right) A figure showing normalized upper bounds (Y-axis) on 47 instances as time passes (X-axis).

Table 2 and Figure 3.left report the number of problems solved within a given time limit. `osprey` solved 25 instances, `cplex` solved 30 (resp. 33 with using 10 cores), `toulbar2` solved 40 (resp. 39 with UDGVNS). 7 instances remain unsolved by any approaches in less than 9,000 seconds. Within the three CFN approaches, `toulbar2 BTD` is the fastest in terms of optimality proofs. Using 10 cores improves `cplex` results by solving 3 more instances and a mean speed-up factor of 2.2 on the same 30 solved instances. A different behavior was found for `toulbar2 UDGVNS` where the mean speed-up factor was only 1.15 in favor of the parallel version. This is because the parallelization of `toulbar2 UDGVNS` is to partially explore the neighborhood in parallel until a single process explores the whole problem sequentially using complete depth-first branch-and-bound and ends the search.

Figure 3.right compares the evolution of upper bounds as time passes for the different methods (except for `osprey` which returns one optimal solution at the end of its A* exploration). Specifically, for each instance, we normalize all energies as follows: the best, potentially suboptimal solution found by any algorithm is 1, the worst solution is 2. This normalization is invariant to translation and scaling. Again, there is a clear separation between CFN and 01LP methods, in favor of CFN approaches. Among the sequential methods, `toulbar2 UDGVNS` has the best anytime profile. Its parallel version `toulbar2 UDGVNS` using 10 cores found the best upper bound for all the 47 instances (see Table 2).

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 (identity 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. An alternative 01LP formulation leads to a much larger number of variables.

On a variety of real instances, we have shown that state-of-the-art optimization algorithms on integer programming or cost function network 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 exploiting a tree decomposition either for optimality proofs or for finding good anytime solutions on very large instances. In practice, the performance of `toulbar2` relies mostly on its fast lower bounds, while adding DEE has a marginal effect [1]. `toulbar2` has now been integrated inside the `osprey` CPD software and is being integrated inside `Rosetta Molecular Modeling suite` [47]. Indeed, another comparison has shown the Rosetta CPD-dedicated Simulated Annealing implementation was outperformed by the exact CFN `toulbar2` BTD approach [66] in terms of GMEC solution quality within a 100-hour time limit. Exact optimization may also be useful in practice as we recently shown using `toulbar2` for the design a real self-assembling protein [56].

We also showed that these CPD problems define challenging benchmarks for AI and OR researchers. Among 47 instances, 7 remains open under 9,000 seconds, having from 28 to 120 mutable residues and at most 198 rotamers per position. Larger problems may exist as most of the known proteins in the PDB database include more than 200 residues[1].

In practice, it must be stressed that just finding the GMEC is not a final answer to real CPD problems. CPD 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` [68, 66].

References

1. Allouche, D., André, I., Barbe, S., Davies, J., de Givry, S., Katsirelos, G., O'Sullivan, B., Prestwich, S., Schiex, T., Traoré, S.: Computational protein design as an optimization problem. Artificial Intelligence **212**, 59–79 (2014)
2. Allouche, D., de Givry, S., Katsirelos, G., Schiex, T., Zytnicki, M.: Anytime Hybrid Best-First Search with Tree Decomposition for Weighted CSP. In: Proc. of CP-15, pp. 12–28. Cork, Ireland (2015)

---

[1] `www.rcsb.org/stats/distribution_residue-count`

Table 2 For each instance: protein (PDB id.), number of mutable residues, maximum domain size (maximum number of rotamers), and CPU-time for solving (with best solution found in parentheses) using `cplex`, `cplex (10 cores)`, `toulbar2`, `toulbar2 BTD`, `toulbar2 UDGVNS`, `toulbar2 UDGVNS (10 cores)`. 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	cplex	cplex-10	toulbar2	BTD	UDGVNS	UDGVNS-10
2TRX	11	48	1.7 (**178440**)	2.1 (**178440**)	**0.1 (178440)**	**0.1 (178440)**	**0.1 (178440)**	**0.1 (178440)**
1PGB	11	49	2.0 (**125306**)	3.8 (**125306**)	**0.1 (125306)**	**0.1 (125306)**	**0.1 (125306)**	**0.1 (125306)**
1PGB	11	198	- (286299)	2781 (**286135**)	**2.5 (286135)**	**2.5 (286135)**	4.1 (**286135**)	4.0 (**286135**)
1HZ5	12	49	5.1 (**150714**)	4.5 (**150714**)	**0.1 (150714)**	**0.1 (150714)**	**0.1 (150714)**	**0.1 (150714)**
1HZ5	12	198	635.1 (**342476**)	264.0 (**342476**)	**1.4 (342476)**	2.1 (**342476**)	2.1 (**342476**)	1.5 (**342476**)
1UBI	13	49	77.7 (**159522**)	34.3 (**159522**)	**0.3 (159522)**	**0.3 (159522)**	**0.3 (159522)**	**0.3 (159522)**
1UBI	13	198	- (383687)	- (382996)	1023 (**380554**)	663.2 (**380554**)	- (**380554**)	- (**380554**)
2DHC	14	198	- (1410934)	- (1411556)	8.2 (**1410850**)	**6.1 (1410850)**	20.4 (**1410850**)	17.2 (**1410850**)
1CM1	17	198	138.4 (**743645**)	116.6 (**743645**)	1.8 (**743645**)	**1.7 (743645)**	3.2 (**743645**)	**1.7 (743645)**
2PCY	18	48	12.8 (**307667**)	14.5 (**307667**)	**0.2 (307667)**	0.3 (**307667**)	**0.2 (307667)**	0.3 (**307667**)
1PIN	28	198	- (1999094)	- (1999094)	- (1994157)	- (1994179)	- (1994135)	- (**1994069**)
1C9O	55	198	- (8024505)	! (8024505)	- (8014215)	- (8014405)	- (8013870)	- (**8013542**)
1FYN	23	186	1235 (**1183427**)	902.3 (**1183427**)	1.7 (**1183427**)	2.6 (**1183427**)	2.3 (**1183427**)	**1.6 (1183427)**
1BK2	24	182	67.2 (**1133737**)	56.3 (**1133737**)	**0.3 (1133737)**	0.6 (**1133737**)	**0.3 (1133737)**	0.4 (**1133737**)
1MJC	28	182	1.8 (**1514364**)	2.4 (**1514364**)	**0.0 (1514364)**	0.1 (**1514364**)	0.1 (**1514364**)	0.1 (**1514364**)
1SHG	28	182	16.3 (**1513151**)	18.9 (**1513151**)	**0.1 (1513151)**	0.3 (**1513151**)	0.3 (**1513151**)	0.2 (**1513151**)
1PIN	28	194	7077 (**1570593**)	881.1 (**1570593**)	2.0 (**1570593**)	**1.3 (1570593)**	3.1 (**1570593**)	2.8 (**1570593**)
1CSK	30	49	8.1 (**1125798**)	9.0 (**1125798**)	**0.1 (1125798)**	0.2 (**1125798**)	**0.1 (1125798)**	**0.1 (1125798)**
1SHF	30	56	5.7 (**1101033**)	7.8 (**1101033**)	**0.1 (1101033)**	0.2 (**1101033**)	0.2 (**1101033**)	**0.1 (1101033)**
1CSP	30	182	964.2 (**2520706**)	120.4 (**2520706**)	**0.4 (2520706)**	**0.4 (2520706)**	0.7 (**2520706**)	0.7 (**2520706**)
1PGB	31	182	- (2442440)	- (2439846)	- (**2433714**)	- (2433843)	- (**2433714**)	- (**2433714**)
1NXB	34	56	10.9 (**2971624**)	11.6 (**2971624**)	**0.1 (2971624)**	0.2 (**2971624**)	0.3 (**2971624**)	0.2 (**2971624**)
1ENH	36	182	- (2637033)	- (2571387)	- (2571065)	- (2570942)	- (**2570849**)	- (**2570849**)
2DRI	37	186	- (2905652)	- (2905276)	135.5 (**2905276**)	**19.2 (2905276)**	762.0 (**2905276**)	1005 (**2905276**)
1FNA	38	48	63.5 (**3750256**)	85.5 (**3750256**)	**0.4 (3750256)**	0.5 (**3750256**)	0.5 (**3750256**)	**0.4 (3750256)**
1CTF	39	56	160.2 (**1881332**)	158.0 (**1881332**)	**0.7 (1881332)**	**0.7 (1881332)**	1.1 (**1881332**)	1.1 (**1881332**)
1TEN	39	66	19.1 (**1959862**)	17.6 (**1959862**)	**0.1 (1959862)**	0.3 (**1959862**)	0.2 (**1959862**)	0.2 (**1959862**)
1UBI	40	182	518.1 (**3068938**)	341.5 (**3068938**)	1.6 (**3068938**)	**1.3 (3068938)**	2.2 (**3068938**)	2.7 (**3068938**)
1CDL	40	186	- (N/A)	- (3594204)	392.6 (**3590514**)	**233.3 (3590514)**	1559 (**3590514**)	1366 (**3590514**)
1CM1	42	186	- (3973849)	6177 (**3895415**)	6.6 (**3895415**)	**6.4 (3895415)**	7.7 (**3895415**)	7.0 (**3895415**)
1C9O	43	182	885.9 (**4959931**)	387.4 (**4959931**)	**1.1 (4959931)**	1.5 (**4959931**)	1.4 (**4959931**)	1.2 (**4959931**)
1BRS	44	194	- (N/A)	- (4008249)	555.3 (**4007610**)	**198.6 (4007610)**	1371 (**4007610**)	1327 (**4007610**)
2PCY	46	56	33.4 (**2935820**)	32.7 (**2935820**)	**0.2 (2935820)**	0.5 (**2935820**)	**0.2 (2935820)**	0.3 (**2935820**)
1POH	46	182	17.6 (**4033880**)	25.9 (**4033880**)	**0.2 (4033880)**	**0.2 (4033880)**	0.3 (**4033880**)	0.3 (**4033880**)
1DKT	46	190	1634 (**4192582**)	1068 (**4192582**)	1.7 (**4192582**)	**1.5 (4192582)**	2.2 (**4192582**)	1.8 (**4192582**)
2CI2	51	183	- (N/A)	- (6391285)	- (6299287)	- (6299277)	- (**6299194**)	- (**6299194**)
1GVP	52	182	- (N/A)	- (5197032)	596.1 (**5196719**)	**217.6 (5196719)**	2469 (**5196719**)	1772 (**5196719**)
1RIS	56	182	- (6222964)	- (6171247)	129.7 (**6171191**)	**21.6 (6171191)**	77.0 (**6171191**)	84.2 (**6171191**)
1LZ1	59	57	581.6 (**7022658**)	292.7 (**7022658**)	**1.5 (7022658)**	**1.5 (7022658)**	1.8 (**7022658**)	2.3 (**7022658**)
2TRX	61	186	443.2 (**7016169**)	374.2 (**7016169**)	**0.8 (7016169)**	1.0 (**7016169**)	1.1 (**7016169**)	1.1 (**7016169**)
2RN2	69	66	210.2 (**8909892**)	257.4 (**8909892**)	**1.1 (8909892)**	1.6 (**8909892**)	1.4 (**8909892**)	2.0 (**8909892**)
3CHY	74	66	- (10461250)	5259 (**10461151**)	88.7 (**10461151**)	**36.9 (10461151)**	76.0 (**10461151**)	88.9 (**10461151**)
1L63	83	182	1602 (**12891031**)	1056 (**12891031**)	**1.7 (12891031)**	2.6 (**12891031**)	2.5 (**12891031**)	2.3 (**12891031**)
1HNG	85	182	1760 (**13532638**)	1672 (**13532638**)	**2.0 (13532638)**	2.6 (**13532638**)	3.2 (**13532638**)	3.6 (**13532638**)
1CSE	97	183	130.3 (**18602292**)	86.0 (**18602292**)	**0.4 (18602292)**	0.8 (**18602292**)	0.8 (**18602292**)	0.7 (**18602292**)
3HHR	115	186	- (N/A)	! (N/A)	- (89193514)	- (89191561)	- (89188633)	- (**89188543**)
1STN	120	190	- (37111502)	! (37111502)	- (37068565)	- (37070863)	- (37055381)	- (**37054896**)
			30 (30)	33 (34)	**40 (41)**	**40 (40)**	39 (43)	39 (**47**)

3. 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. In: Principles and Practice of Constraint Programming, pp. 840–849. Springer (2012)

4. Anfinsen, C.: Principles that govern the folding of protein chains. Science **181**(4096), 223–253 (1973)

5. Boas, F.E., Harbury, P.B.: Potential energy functions for protein design. Current opinion in structural biology **17**(2), 199–204 (2007)

6. Bowie, J.U., Luthy, R., Eisenberg, D.: A method to identify protein sequences that fold into a known three-dimensional structure. Science **253**(5016), 164–170 (1991)

7. Campeotto, F., Dal Palãź, A., Dovier, A., Fioretto, F., Pontelli, E.: A constraint solver for flexible protein models. J. Artif. Int. Res. (JAIR) **48**(1), 953–1000 (2013)

8. Carothers, J.M., Goler, J.A., Keasling, J.D.: Chemical synthesis using synthetic biology. Current opinion in biotechnology **20**(4), 498–503 (2009)

9. 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.: Amber 9. Tech. rep., University of California, San Francisco (2006)

10. Champion, E., André, I., Moulis, C., Boutet, J., Descroix, K., Morel, S., Monsan, P., Mulard, L.A., Remaud-Siméon, M.: Design of α-transglucosidases of controlled specificity for programmed chemoenzymatic synthesis of antigenic oligosaccharides. Journal of the American Chemical Society **131**(21), 7379–7389 (2009)

11. Charpentier, A., Mignon, D., Barbe, S., Cortes, J., Schiex, T., Simonson, T., Allouche, D.: Variable neighborhood search with cost function networks to solve large computational protein design problems. Journal of Chemical Information and Modeling **59**(1), 127–136 (2019)

12. Chowdry, A.B., Reynolds, K.A., Hanes, M.S., Voorhies, M., Pokala, N., Handel, T.M.: An object-oriented library for computational protein design. J. Comput. Chem. **28**(14), 2378–2388 (2007)

13. Cooper, M., de Givry, S., Sanchez, M., Schiex, T., Zytnicki, M., Werner, T.: Soft arc consistency revisited. Artificial Intelligence **174**, 449–478 (2010)

14. Cooper, M.C.: High-order consistency in Valued Constraint Satisfaction. Constraints **10**, 283–305 (2005)

15. Cooper, M.C., de Givry, S., Sánchez, M., Schiex, T., Zytnicki, M.: Virtual arc consistency for weighted CSP. In: Proc. of AAAI'08, vol. 8, pp. 253–258 (2008)

16. Cooper, M.C., de Givry, S., Schiex, T.: Optimal soft arc consistency. In: Proc. of IJCAI'2007, pp. 68–73. Hyderabad, India (2007)

17. Cooper, M.C., Schiex, T.: Arc consistency for soft constraints. Artificial Intelligence **154**(1-2), 199–227 (2004)

18. Dahiyat, B.I., Mayo, S.L.: Protein design automation. Protein science **5**(5), 895–903 (1996)

19. Desmet, J., De Maeyer, M., Hazes, B., Lasters, I.: The dead-end elimination theorem and its use in protein side-chain positioning. Nature **356**(6369), 539–42 (1992)

20. 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 **48**(1), 31–43 (2002)

21. Fersht, A.: Structure and mechanism in protein science: a guide to enzyme catalysis and protein folding. WH. Freemean and Co., New York (1999)

22. Fontaine, M., Loudni, S., Boizumault, P.: Exploiting tree decomposition for guiding neighborhoods exploration for VNS. RAIRO OR **47**(2), 91–123 (2013)

23. Freuder, E.C.: Eliminating interchangeable values in constraint satisfaction problems. In: Proc. of AAAI'91, pp. 227–233. Anaheim, CA (1991)

24. Friesen, A.L., Domingos, P.: Recursive decomposition for nonconvex optimization. In: Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI'15), pp. 253–259 (2015)

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

26. 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.: Osprey: Protein design with ensembles, flexibility, and provable algorithms. Methods Enzymol (2012)

27. Georgiev, I., Lilien, R.H., Donald, B.R.: Improved Pruning algorithms and Divide-and-Conquer strategies for Dead-End Elimination, with application to protein design. Bioinformatics **22**(14), e174–83 (2006)

28. Georgiev, I., Lilien, R.H., Donald, B.R.: 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), 1527–42 (2008)

29. de Givry, S., Prestwich, S., O'Sullivan, B.: Dead-End Elimination for Weighted CSP. In: Proc. of CP-13, pp. 263–272. Uppsala, Sweden (2013)

30. Goldstein, R.F.: Efficient rotamer elimination applied to protein side-chains and related spin glasses. Biophysical journal **66**(5), 1335–40 (1994)

31. Gront, D., Kulp, D.W., Vernon, R.M., Strauss, C.E., Baker, D.: Generalized fragment picking in rosetta: design, protocols and applications. PloS one **6**(8), e23294 (2011)

32. Grunwald, I., Rischka, K., Kast, S.M., Scheibel, T., Bargel, H.: Mimicking biopolymers on a molecular scale: nano(bio)technology based on engineered proteins. Philosophical transactions. Series A, Mathematical, physical, and engineering sciences **367**(1894), 1727–47 (2009)
33. Hallen, M.A., Keedy, D.A., Donald, B.R.: Dead-end elimination with perturbations (deeper): A provable protein design algorithm with continuous sidechain and backbone flexibility. Proteins: Structure, Function, and Bioinformatics **81**(1), 18–39 (2013)
34. Harvey, W.D., Ginsberg, M.L.: Limited discrepancy search. In: Proc. of the 14th IJCAI. Montréal, Canada (1995)
35. 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)
36. Hurley, B., O'Sullivan, B., Allouche, D., Katsirelos, G., Schiex, T., Zytnicki, M., de Givry, S.: Multi-Language Evaluation of Exact Solvers in Graphical Model Discrete Optimization. Constraints **21**(3), 413–434 (2016)
37. Janin, J., Wodak, S., Levitt, M., Maigret, B.: Conformation of amino acid side-chains in proteins. Journal of molecular biology **125**(3), 357–386 (1978)
38. Khalil, A.S., Collins, J.J.: Synthetic biology: applications come of age. Nature Reviews Genetics **11**(5), 367–379 (2010)
39. 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.: Computational redesign of a mononuclear zinc metalloenzyme for organophosphate hydrolysis. Nature chemical biology **8**(3), 294–300 (2012)
40. Khoury, G.A., Smadbeck, J., Kieslich, C.A., Floudas, C.A.: Protein folding and *de novo* protein design for biotechnological applications. Trends in biotechnology **32**(2), 99–109 (2014)
41. Kingsford, C.L., Chazelle, B., Singh, M.: Solving and analyzing side-chain positioning problems using linear and integer programming. Bioinformatics **21**(7), 1028–36 (2005)
42. Kuhlman, B., Baker, D.: Native protein sequences are close to optimal for their structures. Proceedings of the National Academy of Sciences of the United States of America **97**(19), 10383–8 (2000)
43. Larrosa, J.: On arc and node consistency in weighted CSP. In: Proc. AAAI'02, pp. 48–53. Edmondton, (CA) (2002)
44. 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 19th IJCAI, pp. 84–89. Edinburgh, Scotland (2005)
45. Larrosa, J., Schiex, T.: Solving weighted CSP by maintaining arc consistency. Artif. Intell. **159**(1-2), 1–26 (2004)
46. Leach, A.R., Lemon, A.P.: Exploring the conformational space of protein side chains using dead-end elimination and the A* algorithm. Proteins **33**(2), 227–39 (1998)
47. Leaver-Fay, A., Tyka, M., Lewis, S.M., Lange, O.F., Thompson, J., Jacak, R., Kaufman, K., Renfrew, P.D., Smith, C.A., Sheffler, W., Davis, I.W., Cooper, S., Treuille, A., Mandell, D.J., Richter, F., Ban, Y.E.A., Fleishman, S.J., Corn, J.E., Kim, D.E., Lyskov, S., Berrondo, M., Mentzer, S., Popović, Z., Havranek, J.J., Karanicolas, J., Das, R., Meiler, J., Kortemme, T., Gray, J.J., Kuhlman, B., Baker, D., Bradley, P.: Rosetta3: an object-oriented software suite for the simulation and design of macromolecules. Methods Enzymol. **487**, 545–574 (2011)
48. Lecoutre, C., Roussel, O., Dehani, D.: WCSP integration of soft neighborhood substitutability. In: Principles and Practice of Constraint Programming, pp. 406–421. Springer (2012)
49. Lewis, J.C., Bastian, S., Bennett, C.S., Fu, Y., Mitsuda, Y., Chen, M.M., Greenberg, W.A., Wong, C.H., Arnold, F.H.: Chemoenzymatic elaboration of monosaccharides using engineered cytochrome p450bm3 demethylases. Proceedings of the National Academy of Sciences **106**(39), 16550–16555 (2009)
50. Looger, L.L., Hellinga, H.W.: 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**(1), 429–45 (2001)
51. Loudni, S., Boizumault, P.: Solving constraint optimization problems in anytime contexts. In: Proc. of IJCAI, pp. 251–256. Acapulco, Mexico (2003)

52. Lovell, S.C., Word, J.M., Richardson, J.S., Richardson, D.C.: The penultimate rotamer library. Proteins **40**(3), 389–408 (2000)
53. Martin, V.J., Pitera, D.J., Withers, S.T., Newman, J.D., Keasling, J.D.: Engineering a mevalonate pathway in escherichia coli for production of terpenoids. Nature biotechnology **21**(7), 796–802 (2003)
54. Mladenović, N., Hansen, P.: Variable Neighborhood Search. Comput. Oper. Res. **24**(11), 1097–1100 (1997)
55. Nestl, B.M., Nebel, B.A., Hauer, B.: Recent progress in industrial biocatalysis. Current Opinion in Chemical Biology **15**(2), 187–193 (2011)
56. Noguchi, H., Addy, C., Simoncini, D., Wouters, S., Mylemans, B., Van Meervelt, L., Schiex, T., Zhang, K.Y., Tame, J.R., Voet, A.R.: Computational design of symmetrical eight-bladed β-propeller proteins. IUCrJ **6**(1) (2019)
57. O'Meara, M.J., Leaver-Fay, A., Tyka, M., Stein, A., Houlihan, K., DiMaio, F., Bradley, P., Kortemme, T., Baker, D., Snoeyink, J., Kuhlman, B.: A combined covalent-electrostatic model of hydrogen bonding improves structure prediction with rosetta. J. Chem. Theory Comput. **11**(2), 609–622 (2015)
58. Ouali, A., Allouche, D., de Givry, S., Loudni, S., Lebbah, Y., Eckhardt, F., Loukil, L.: Iterative Decomposition Guided Variable Neighborhood Search for Graphical Model Energy Minimization. In: Proc. of UAI-17, pp. 550–559. Sydney, Australia (2017)
59. Pabo, C.: Molecular technology. Designing proteins and peptides. Nature **301**(5897), 200 (1983)
60. Peisajovich, S.G., Tawfik, D.S.: Protein engineers turned evolutionists. Nature methods **4**(12), 991–4 (2007)
61. 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)
62. Pierce, N.A., Winfree, E.: Protein design is NP-hard. Protein engineering **15**(10), 779–82 (2002)
63. Pleiss, J.: Protein design in metabolic engineering and synthetic biology. Current opinion in biotechnology **22**(5), 611–7 (2011)
64. Raha, K., Wollacott, A.M., Italia, M.J., Desjarlais, J.R.: Prediction of amino acid sequence from structure. Protein science **9**(6), 1106–19 (2000)
65. Schiex, T.: Arc consistency for soft constraints. In: Principles and Practice of Constraint Programming - CP 2000, *LNCS*, vol. 1894, pp. 411–424. Singapore (2000)
66. Simoncini, D., Allouche, D., de Givry, S., Delmas, C., Barbe, S., Schiex, T.: Guaranteed discrete energy optimization on large protein design problems. Journal of Chemical Theory and Computation **11(12)**, 5980–5989 (2015)
67. 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)
68. Traoré, S., Allouche, D., André, I., de Givry, S., Katsirelos, G., Schiex, T., Barbe, S.: A new framework for computational protein design through cost function network optimization. Bioinformatics **29**(17), 2129–2136 (2013)
69. Traoré, S., Roberts, K.E., Allouche, D., Donald, B.R., André, I., Schiex, T., Barbe, S.: Fast search algorithms for computational protein design. Journal of computational chemistry **37**(12), 1048–1058 (2016)
70. Verges, A., Cambon, E., Barbe, S., Salamone, S., Le Guen, Y., Moulis, C., Mulard, L.A., Remaud-Siméon, M., André, I.: Computer-aided engineering of a transglycosylase for the glucosylation of an unnatural disaccharide of relevance for bacterial antigen synthesis. ACS Catalysis **5**(2), 1186–1198 (2015)
71. Voigt, C.A., Gordon, D.B., Mayo, S.L.: Trading accuracy for speed: A quantitative comparison of search algorithms in protein sequence design. Journal of molecular biology **299**(3), 789–803 (2000)