JCTC Journal of Chemical Theory and Computation

Article

Guaranteed Discrete Energy Optimization on Large Protein Design Problems

David Simoncini, david allouche, Simon de Givry, Céline Delmas, Sophie Barbe, and Thomas Schiex

J. Chem. Theory Comput., **Just Accepted Manuscript** • DOI: 10.1021/acs.jctc.5b00594 • Publication Date (Web): 11 Nov 2015

Downloaded from http://pubs.acs.org on November 23, 2015

ACS Publications

Journal of Chemical Theory and Computation is published by the American Chemical Society. 1155 Sixteenth Street N.W., Washington, DC 20036

Guaranteed Discrete Energy Optimization on Large Protein Design Problems

David Simoncini,[†] David Allouche,[†] Simon de Givry,[†] Celine Delmas,[†] Sophie Barbe,[‡] and Thomas Schiex[*,†]

[†]*INRA MIAT, UR 875, Castanet-Tolosan, 31326 Cedex, France*

[‡]*INRA-INSA 792,CNRS-INSA 5504, LISBP, Toulouse, France*

E-mail: tschiex@toulouse.inra.fr

Abstract

In Computational Protein Design (CPD), assuming a rigid backbone and amino-acid rotamer library, the problem of finding a sequence with an optimal conformation is NP-hard.

In this paper, using Dunbrack's rotamer library and Talaris2014 decomposable energy function, we use an exact deterministic method combining branch and bound, arc consistency and tree-decomposition to provenly identify the global minimum energy sequence-conformation on full-redesign problems, defining search spaces of size up to 10^{234}. This is achieved on a single core of a standard computing server, requiring a maximum of 66GB RAM. A variant of the algorithm is able to exhaustively enumerate all sequence-conformations within an energy threshold of the optimum.

These proven optimal solutions are then used to evaluate the frequencies and amplitudes, in energy and sequence, at which an existing CPD-dedicated simulated annealing implementation may miss the optimum on these full redesign problems. The probability of finding an optimum drops close to 0 very quickly. In the worst case, despite $1,000$ repeats, the annealing algorithm remained more than 1 Rosetta unit away of the optimum, leading to design sequences that could differ from the optimal sequence by more than 30% of their amino-acids.

Introduction

The main objective of Computational Protein Design (CPD) is to find amino-acid sequences that adopt a desired tertiary structure, be it for functional or structural purposes.[1] This requires an energy function that accurately reflects protein stability and a reliable search method to identify a sequence with a conformation of optimal stability (Global Minimum Energy Conformation or GMEC).

These two ingredients for success are not independent. A perfect funnel-like energy function would directly lead to optimal designs while a perfect search method would enable us to identify flaws in the energy function and improve it.[2]

The most usual description of the CPD GMEC problem relies on a pairwise decomposable energy function, a discretized description of the amino-acid conformational space based on a library of frequent side-chain conformations (or rotamers)[3] and a rigid backbone. Under such assumptions, the problem of searching for a sequence with a minimum energy conformation is known to be NP-hard.[4] On this basis, most CPD tools prefer to rely on stochastic methods, with no finite time convergence guarantees. As an example, simulated annealing[5] is used in the *Rosetta Modeling suite*[6] and *EGAD* uses a Genetic Algorithm.[7]

A conceptually more attractive solution would be to rely on a deterministic algorithm, providing a direct finite-time access to the ultimate meaning of the energy function, avoiding stochastic sampling noise. The Dead End Elimination theorem (DEE[8]) combined with an A* search, as available in the OSPREY design suite,[9] is the deterministic method of choice in CPD. However, on medium or large size designs, the finite but worst-case exponential time and space needed by DEE/A* is so huge that it is often unusable in practice and stochastic algorithms remain the only alternative.

Therefore, a majority of CPD research relies on stochastic methods and implicitly or explicitly assumes that they lead to optimal or very close to optimal solutions with high frequency.[10] This capacity to efficiently identify sequences with optimal conformations is especially challenging in *de novo* design, where the full sequence of a protein is designed. After two decades of research, there only exists a handful of experimentally verified *de novo* designs, half of which have been obtained with the *Rosetta Modeling suite*.[6] The designs of the first globular protein with a novel fold[11] and of an enzyme catalyst for a stereo-selective bimolecular Diels-Alder reaction[12] are two striking examples of these rare successes.

Beyond *de novo* design, accessing sequences with an optimal conformation becomes of utmost importance when the aim is not to design a new protein but to exploit the flaws of the designs to improve the energy function. For example, this has been done in the composition-constrained full redesign of the 56-residue β1 domain of streptococcal protein G.[2]

We show in this paper that it is possible to obtain a sequence with a guaranteed optimal conformation in reasonable time, on standard hardware, on computationally extremely challenging problems of a size that is far beyond what has been solved with the usual deterministic DEE/A^* CPD approach[13–15] or other approaches (such as Mixed integer or Quadratic Programming).[15–19] We achieve this assuming a rigid backbone, a recent dense backbone-dependent rotamer library[3] and a recent design-oriented decomposable energy function.[20]

As an example of the computational power in our hands, we have estimated the reliability of the fixed-backbone simulated annealing protocol available in the *Rosetta Modeling suite* on roughly one hundred full redesigns problems. We observe that on most designs the stochastic algorithm misses the optimum sequence-conformation with very high frequency. To further exercise our computational capacities, we attempted to exhaustively enumerate all sequence-conformations with energy within a fixed threshold of the GMEC, providing direct access to a local density of states.

With the drastic reduction of computational efforts they offer, our new computational methods (available as an open source tool) may have a wide impact in CPD, especially for energy function refinement.[2] The pairwise decomposable energy optimization problem also appears as a sub-problem in a variety of more general situations, including design with continuous rotamers or flexible (ensemble) backbone[21] or in multi-state design.[22] The efficiency of our approach will therefore contribute to these generalized models. Its ability to exhaustively and rapidly enumerate conformations could also be useful for conformational affinity computations.[23]

Methods

Modeling CPD as a Cost Function Network

In our settings (fixed-backbone, rotamer library and pairwise decomposable energies), the energy of a given sequence-conformation can be written as:

$$E(c) = E_t + \sum_i E(i_r) + \sum_{i<j} E(i_r, j_s)$$

E_t is the template energy. $E(i_r)$ captures internal side-chain energies and rotamer-backbone interactions for rotamer r at position i. $E(i_r, j_s)$ captures pairwise interactions between rotamers r and s at positions i and j respectively. These terms can be pre-computed and stored in a set of energy matrices. A CFN is a discrete optimization structure introduced in Artificial Intelligence, in Constraint Programming, also known as a Weighted Constraint Satisfaction problem.[24] The problem of finding the GMEC can be easily reduced to solving this NP-hard problem which we describe now.

Cost function networks (CFNs) are deterministic Graphical Models that are derived from Constraint Satisfaction Problems.

Definition 1. *A CFN (X, W, k) is defined by:*

- *a set X of discrete variables $x_i \in X$ indexed by $I = \{1, \ldots, n\}$, each variable x_i takes* *its values in a finite domain D_i of maximum cardinality d.*

- *a set of cost functions $w_S \in W$ each involving a subset $\{x_i \in X \mid i \in S\}$ of all variables and taking* non negative *values in* $[0, k]$.
- *The value k is a finite or infinite cost representing an upper bound on costs: a cost of k or above is considered as forbidden.*

The set $S \subset I$ of a cost function w_S is called the scope of the cost function. We denote by D^S the Cartesian product of the domains of all variables indexed in S: $D^S = \prod_{i \in S} D_i$. Given a tuple $t \in D^S$, and $S' \subset S$ we denote by $t[S']$ the projection of t on $D^{S'}$. We define two operations over cost functions:

1. The sum of two cost functions w_S and $w_{S'}$ is the cost function $(w_S + w_{S'})$ with scope $S \cup S'$ such that $(w_S + w_{S'})(t) = w_S(t[S]) + w_{S'}(t[S'])$.
2. The marginal of a cost function w_S over a subset $T \subset S$ is the cost function $(w_S)_{|T}$ such that $(w_S)_{|T}(t) = \min_{t' \in D^S, t'[T]=t} w_S(t')$.

The cost of an assignment t of all variables is defined as the sum $\sum_{w_S \in W} w_S(t[S])$ of all cost functions. If it is strictly less than k, the assignment is said to be a solution. The weighted constraint satisfaction problem (WCSP) is to identify a solution of guaranteed minimum cost over all $t \in D^X$. Because of the non negativity of all cost functions, the cost function $w_\varnothing \in W$, a constant cost function with no parameters, defines a lower bound on this minimum cost.

Modeling the rigid backbone, discrete rotamer, decomposable energy GMEC problem as a CFN is immediate:[15,17] each position or residue i in the protein sequence defines a variable x_i which takes its values in the set of all possible rotamers for this position. All terms in the energy matrices can be shifted by a sufficient constant to make them all non negative. This preserves the optimal solution. In the resulting matrices, the template energy term E_t is represented as the constant cost function $w_\varnothing$, one body energy terms on position i are represented as cost functions w_i of one variable x_i, and two bodies energy terms between positions i and j are represented as cost functions w_{ij} of the two variables x_i and x_j. Because this defines one-to-one mappings, we confound in the rest of the paper variables with amino-acid positions (or residues), values with rotamers and costs with energies.

Guaranteed GMEC identification with CFN methods

Deterministic CPD methods are usually based on two components:

- Dead End Elimination[8,25] is a polynomial time pre-processing algorithm that prunes rotamers (or combinations of rotamers) that are locally dominated by other rotamers (or combinations of rotamers) in terms of energy. This process may prune sub-optimal solutions, even close to the optimum, but preserves the GMEC.

- Since finding the GMEC is NP-hard, DEE cannot suffice in all cases. It is therefore followed by the exponential time and space best-first search algorithm A^*, using a dedicated lower bound.[26] This lower bound is used to prune branches that have not been pruned by DEE, during search.

Our algorithm combines instead Arc Consistency filtering, a family of CFN pruning and lower bounding techniques[27] with Depth-First Branch and Bound[28] enhanced with variable elimination and graph-based problem decomposition techniques.[29–31]

Local consistency

In CFNs, local consistencies, such as Arc Consistencies, define a family of polynomial time problem reformulation techniques, called filtering algorithms, that take as input an original CFN and that transform it in another *equivalent* CFN. This final CFN has the same set of variables, possibly different cost functions and it must satisfy a specific set of local properties associated to the given local consistency enforced. The filtered CFN is equivalent to the original CFN in the sense that it defines exactly the same set of solutions, each with the same exact cost as in the original CFN. Thus, solving the original CFN or the filtered CFN is absolutely equivalent. This may have the advantage that the filtered CFN has a larger constant cost $w_\varnothing$, providing a stronger lower bound on the optimum.

To perform these transformations, filtering relies on so-called Equivalent Preserving Transformations[32,33] that shift costs (or energies) between cost functions w_S and $w_{S'}$, $S' \subset S$, in such a way that the sum of local costs is unchanged and no negative costs appears.

The general form of these elementary operations is captured by the procedure $\texttt{Shift}(S, S', t', \alpha)$ where $S' \subset S \subset I$ and $t' \in D^{S'}$, that shifts an amount of cost/energy α between $w_{S'}(t')$ and w_S. Preconditions prevent the application of operations that would create negative costs. By simple algebra, it is easy to see that the sum of costs of any solution (optimal or not) is unchanged by any such transformation.

Algorithm 1: Algorithm for applying an Equivalence Preserving Transformation

```
Precondition: $-w_{S'}(t') \leq \alpha \leq \min_{t \in D^S, t[S']=t'} w_S(t)$;
Procedure $\texttt{Shift}(S, S', t', \alpha)$
    $w_{S'}(t') \leftarrow w_{S'}(t') + \alpha$;
    foreach ($t \in D^S$ such that $t[S'] = t'$) and $w_S(t) < k$ do
        $w_S(t) \leftarrow w_S(t) - \alpha$;
```

We show how such operations can be used to reformulate a CFN and increase $w_\varnothing$ on a very simple example. A problem with two variables and two values in each domain is represented in Figure 1(b). We assume that $k = \infty$. This CFN satisfies the preconditions for applying either $\texttt{Shift}(\{1,2\}, \{2\}, b, 1)$ ((a), left) or $\texttt{Shift}(\{1,2\}, \{1\}, b, 1)$ ((c), right). This last problem allows then to apply $\texttt{Shift}(\{1\}, \varnothing, (), 1)$ leading to problem 1(d). The four problems in Figure 1 are all equivalent. However, the problem (d) is the most explicit since $w_\varnothing$ explicitly shows that 1 is a lower bound on the cost of any solution of this problem.

Beyond these transformations, local consistencies will also prune any rotamer/value for which it can directly be proved that they cannot participate in a solution. This is the case if $w_i(a) + w_\varnothing \geq k$ since all cost functions are non negative. Then, the value $a \in D_i$ can be deleted. This preserves equivalence over all solutions (optimal or not). The larger $w_\varnothing$

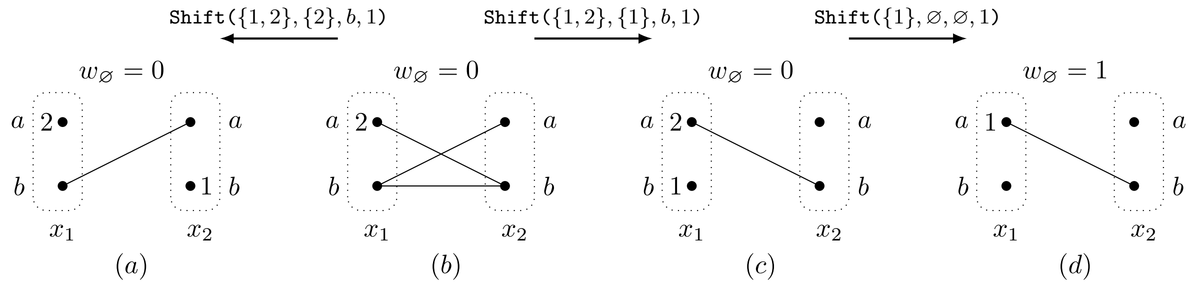


Figure 1: Four equivalent CFNs represented as weighted n-partite graphs. Each vertex corresponds to a value $a \in D_i, \forall i \in I$. All the values of a same variable domain are gathered in a dotted rectangle. If a value a is such that $w_i(a) \neq 0$, the corresponding vertex is weighted by $w_i(a)$. Similarly, an edge between the vertices of $a \in D_i$ and $b \in D_j$ is used to represent the fact that $w_{ij}(a, b) = 1$. An arrow between two CFNs shows how the source CFN can be transformed in the target CFN while preserving equivalence and non negative costs.

and the lower k, the stronger the pruning. In the example above, the cost of the solution $(x_1 = a, x_2 = a)$ is 2. If we are now interested in finding solutions of cost strictly better than 2, we can simply set k to 2. In this case, the value a satisfies the pruning condition and can be removed from D_1.

On a large problem, depending on the sequence of Shift operations which are applied, the lower bound $w_\varnothing$ may increase more or less. A succession of results on different strategies of application of these transformations, each corresponding to specific local properties that need to be satisfied in the target problem, have been proposed in the last decade. In this paper, we use two of these, the first one is Virtual Arc Consistency,[27] that builds a powerful sequence of transformations with a strong final lower bound $w_\varnothing$ and the less expensive but weaker Existential Directed Arc Consistency.[34] The crucial property that guarantees that our method provides exact optima is that these two strategies solely rely on Equivalence Preserving Transformations. They are therefore guaranteed to produce an equivalent problem, with a possibly strengthened lower bound $w_\varnothing$ and thus increased but safe pruning.

Depth First branch and Bound

Because arc consistencies algorithms are polynomial time, they cannot solve all problems and must be completed by tree search. We use a polynomial space Depth-First Branch and Bound algorithm.[28] This algorithm explores a tree whose root is the full CFN describing the original CPD problem. The sons of each node are typically defined by choosing a variable/position which has more than one possible value/rotamer and either set the variable to one value (left branch) or remove this value from the domain of the variable (right branch). The leaves of this tree are complete assignments whose cost can be easily computed. If the tree is exhaustively explored, an optimal solution can be identified as the best leaf encountered when visiting the tree. Depth First Search visits the tree by always developing a deepest unexplored node first. It requires only polynomial memory to do so.[35]

To avoid exploring the whole tree, Branch and Bound prunes branches using two bounds. An upper bound ub is defined by the cost/energy of the best known solution. To find a solution of cost less than ub or prove optimality, we set k to ub. Then, by filtering the problem at any node of the tree, we obtain a local lower bound $w_\varnothing$ on the cost of any leaf/solution below the current node. If $w_\varnothing \geq k$, arc consistency will prune all values. Since arc consistency preserves equivalence, this shows that the current branch can not lead to a solution of cost better than $k = ub$. It can therefore be safely pruned.

The pruning capacities of this algorithm follow from the quality of the lower and upper bounds it exploits. Because of its significant computational cost and associated strong lower bound, we only use Virtual Arc Consistency (VAC[27]) as a pre-processing, to filter the problem at the root node. The resulting filtered equivalent problem is used instead of the original one. During the rest of the tree search, we use the cheaper Existential Directed Arc Consistency (EDAC[34]) to update the lower bound $w_\varnothing$ following left and right branching decisions. To start search with a non naive upper bound, we use the cost of the solution of one Rosetta fixbb protocol to initialize k. Furthermore, during search, each time a new improved solution is found, its cost is used to update the value of k. This incrementally excludes many sub-optimal assignments from the set of solutions, leading to increased value pruning by local consistency. The resulting algorithm is denoted as fixbb-VAC-DFBB(EDAC,dyn-k) as it uses fixbb to initialize k, VAC for pre-processing and then both maintains EDAC and updates k dynamically during search.

Arc consistency only prunes values/rotamers that are proved to never appear in a solution (conformation of cost/energy lower than k). Therefore, by adjusting k, we can precisely control which rotamers are protected from deletion, in a much simpler way than DEE. It is therefore easy to transform our Branch and Bound optimization algorithm above, targeted at finding the GMEC, into an algorithm capable of exhaustive sub-optimal sequence-conformation enumeration: the value of k is initially set to the desired upper bound and never updated during search. Then, the leaves encountered during search will be the exhaustive list of solutions with cost/energy less than k. The resulting algorithm is denoted as fixed-VAC-DFBB(EDAC,stat-k) as is uses a fixed initial value of k, VAC for pre-processing and then maintains EDAC while keeping k fixed during search.

Variable elimination

Every CPD instance defines a so-called interaction graph. In this graph, each position is a vertex and an edge exists between two vertices i and j if a non constant interaction term $E(i_r, j_s)$ exists between the two residues. In CFN theory, this graph is also defined and called the primal graph of the CFN.[36]

Non serial dynamic programming,[37] also called variable or bucket elimination,[38] is a possible approach to efficiently solve problems whose interaction graph is made of different loosely coupled sub-problems. This easily happens in protein design if the energy computations are done using a cutoff function,[39] as is the case with Talaris2014 in Rosetta.

As an example, consider a CFN $P = (X, W, k)$ in which a variable x_1 is only connected to variables x_2 and x_3 by cost functions $w_{12}, w_{13} \in W$. This variable has a degree of only 2 in the interaction graph. It is possible to eliminate variable x_1 from the problem, similarly to what is done with Gaussian elimination on linear equalities. We consider the cost functions involving x_1: these are w_1, w_{12} and w_{13}. We can remove the variable x_1 from X and the cost functions w_1, w_{12} and w_{13} that involve it and replace them by $(w_1 + w_{12} + w_{13})_{|23}$, a new cost function with scope $\{2,3\}$. This defines a new CFN where variable x_1 has been eliminated and denoted as $P - x_1$. By distributivity of min and $+$, one can show that the CFN $P - x_1$ has the same optimal cost as the original CFN P.[37] Furthermore, from an optimal solution of $P - x_1$ it is possible to build an optimal solution of P by simply choosing the value of x_1 that minimizes the sum of the eliminated cost functions $(w_1 + w_{12} + w_{13})$. The complexity of this process is dominated by the computation of the new cost function. For a variable x_i of degree g, this is in $O(d^{g+1})$ and is therefore inexpensive for small degrees.

In our algorithm, we eliminate all variables of degree ≤ 2 as a pre-processing step. We also eliminate all the variables of degree 2 that may appear during tree search.[40] Indeed, when a variable is assigned during depth-first branch and bound, it disappears from the scope of all the cost functions that involved it and the degrees of neighbor variable decreases.

Tree decomposition

Usually, such variable elimination algorithms are not restricted in the degree of the variables they eliminate. All variables are eliminated successively. The sequence of elimination steps is organized cleverly to try to minimize the maximum elimination cost. This is more formally captured by a graph parameter known as treewidth (or dimension[37]).

A tree decomposition of a graph is defined by a tree (C, T). The set of nodes of the tree is $C = \{C_1, \ldots, C_m\}$ where C_e is a set of variables ($C_e \subseteq X$) called a cluster. T is a set of edges connecting clusters and forming a tree (a connected acyclic graph). The set of clusters C must cover all the variables ($\bigcup_{C_e \in C} C_e = X$) and all the cost functions ($\forall \{i,j\} \in E, \exists C_e \in C \; s.t. \; i,j \in C_e$). Furthermore, if a variable i appears in two clusters C_e and C_g, i must also appear in all the clusters C_f on the unique path from C_e to C_g in (C, T). If the cardinality of the largest cluster in a tree decomposition is $\omega + 1$ then the *width* of the decomposition is ω. The *treewidth* of a graph is the minimum width among all its decompositions.[41] An example of a primal graph and an associated tree-decomposition is shown in Figure 2.

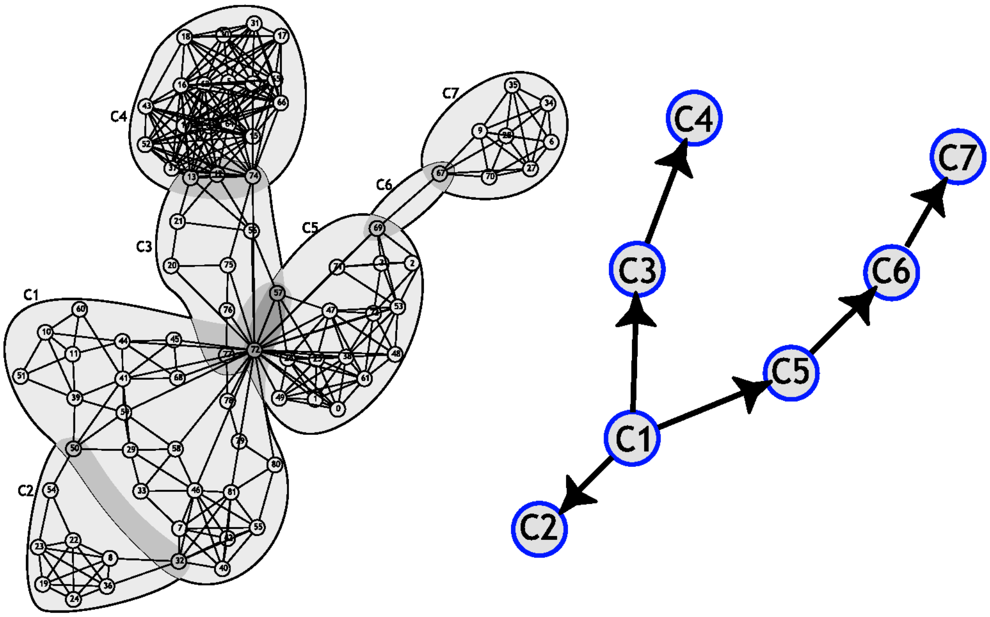


Figure 2: A tree-decomposition of a primal graph. Each cluster is represented as a gray region covering a set of vertices and edges. The organization in a tree is shown on the right. This decomposition has width 26 since the largest cluster C_1 contains 27 variables/residues. The whole problem has 100 variables. The cluster C_1 intersects with cluster C_2 over 2 variables only (variable indexed 32 and 50).

Tree decomposition based variable elimination (dynamic programming) eliminates variables from leaf clusters first. It requires only $n \cdot d^{\omega}$ steps to identify a proven optimal solution. It has been exploited for exactly solving the WCSP,[38] side-chain packing[42,43] and CPD instances,[44] for decompositions of small width. It becomes however unusable, even if ω is not too large, when d increases. This is the case for protein design problems where mutations with several amino-acid types are authorized: there may be several hundreds of rotamers per mutable residue.

Our algorithm does not use brute force variable elimination. It instead simultaneously exploits a tree decomposition and the pruning power of arc consistency filtering during tree search.[30] We root the tree decomposition in an arbitrary cluster (we use the largest cluster, eg. C_1 in Figure 2) and use our fixbb-VAC-DFBB(EDAC,dyn-k) algorithm with a variable assignment order that never assigns a variable from a cluster before all the variables of parent clusters have been assigned. When the variables in the intersection of a parent cluster and a son cluster are assigned, the 2 sub-problems become independent. The son cluster can be solved recursively using the same method and the result memorized in association with the corresponding assignment of the variables in the intersection of the clusters. If later, this same separator assignment is visited by Branch and Bound, the previous memorized result can be reused instead of resolving the sub-problem from scratch leading to worst-case complexity in $O(d^{\omega})$.[29,30] Because of rotamer and branch pruning by arc consistency, only a very small fraction of the work that would have been done using variable elimination is required. The resulting algorithm is denoted as fixbb-VAC-TD-DFBB(EDAC,dyn-k) and the associated flowchart in Figure 3.

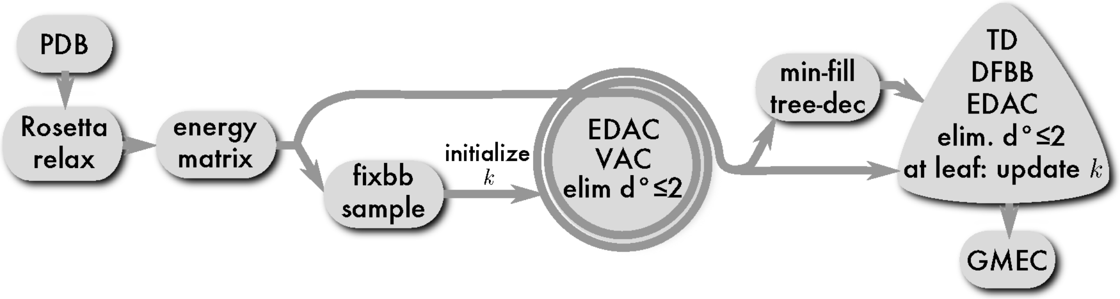


Figure 3: Flowchart of our method: the backbone structure is loaded in Rosetta, relaxed and used to generate the energy matrices and run one fixbb protocol to get an initial upper bound k. The fixbb-VAC-TD-DFBB(EDAC,dyn-k) algorithm then applies EDAC, VAC and variable elimination of variables with degree less than 2 until a fixed point is reached (preserving optimum) as a pre-processing step. The resulting matrix defines an interaction graph and a tree-decomposition is computed using the min-fill heuristics.[45] The matrix is also used in a exhaustive Depth-First Branch and Bound search algorithm that maintains EDAC, eliminates variables of degree less than 2 at each node, updating the upper bound k each time a new solution is found and exploiting the tree-decomposition to avoid redundant calculations. This identifies the GMEC and proves its optimality.

This algorithm has been implemented in our open source solver *toulbar2* (version 0.9.8, sources available at https://mulcyber.toulouse.inra.fr/projects/toulbar2). The weighted-degree[46] and last-conflict heuristics,[47] restricted to explore successive clusters of the tree decomposition are used to choose the next variable to explore during tree search. The EDAC value[34] of the selected variable is explored first.

Compared to previous experiments using CFN methods,[15] this new algorithm differs in the fact that:

1. it uses the non trivial initial upper-bound provided by one run of fixbb protocol to initialize k,

2. it uses the recent Virtual Arc Consistency (VAC) for pre-processing,

3. it exploits a tree decomposition during search.

4. it abandons Dead End Elimination (adding DEE slightly decreases the overall number of solved problems, see SI).

For reproducibility, beyond the sources of our solver, we give access to the Python script used to extract energy matrices from Rosetta, compute the GMEC and enumerate sub-optimal conformation as well as the relaxed backbones of the benchmark set in the public BitBucket repository https://bitbucket.org/satsumaimo/ptcfopd. This script can be directly applied on any PDB backbone structure.

Benchmark preparation

Our benchmark set has been extracted from the PDB (Protein Data Bank[48]) and filtered with the following criteria: monomeric proteins with an X-ray resolved structure below 2 Å, with no missing or non standard residues and no ligand. Chain length was limited to 100 amino-acids. A total of 107 proteins were extracted as of the 1^{st} of September 2014, retrieving only representative structures at 30% sequence identity. The chain lengths scale from 50 to 100 residues, defining a collection of problems of gradually increasing complexity. Each protein was then relaxed 10 times with the default Rosetta relax protocol,[6] using Talaris2014 energy function[20] and the backbone of lowest energy used for benchmarking (See SI for a detailed list of the proteins).

The rotamer library used is the backbone dependent extended version of Dunbrack library 2010 integrated in Rosetta. The Talaris2014 energy function was used to compute the energy matrices.[20] The largest matrix took less then 10 minutes to compute (see SI). The energy matrices were extracted through PyRosetta[49] and compared to the energy matrices used within Rosetta fixbb protocol in order to guarantee that all optimizations use exactly the same scoring function. All residues were considered as mutable with any of the twenty natural amino-acids, except for cysteines involved in disulfide bridges.

Production of Rosetta samples

1,000 sequences were generated for each target with the Rosetta molecular modeling suite. We used the *fixbb* protocol with default parameters and Talaris2014 energy function[20] (see SI).

Results and discussion

Our experiments are targeted at evaluating the computational limits of the CFN methods on Computational Protein Design. Our benchmark represents a collection of protein design problems of gradually increasing size and hardness, with a maximum design space of 10^{234} sequence-conformations.

Finding and enumerating sequence-conformations

Within a fixed time limit of 100 hours, our algorithm identified and proved the optimality of the GMEC for 98 protein design problems out of 107, using a single core of an Intel Xeon CPU E5-2690@2.90GHz (dated Q1-2012) and a maximum of 66GB of memory on the largest problem (with a 1.7 GB energy matrix file, see SI).

As expected, the computation time needed to find the GMEC with this algorithm grows exponentially with the size of the problem. Figure 4 shows a plot of the running time (with a logarithmic scale) as a function of the problem bit-size. We define the problem bit-size as the number of bits needed to represent the energy matrices with eight digits after the decimal point, following compression. We use it here instead of the size of the sequence-conformation space, as it gives a better linear regression coefficient with cpu-time ($r^2 = 0.661$ instead of 0.478).

The observed exponential time complexity defines a computational barrier after which GMECs cannot be found and proved optimal in reasonable time using current algorithmic technology. This limit is however far enough so that most of the proteins in our benchmark could be handled efficiently on one CPU core. A few years ago, sizes of up to 10^{78} sequence-conformations had already been reached by BROMAP,[13,14] a software that does not seem to be publicly available. Then sizes of up to 10^{98} were reached in less than 100 hours.[15]

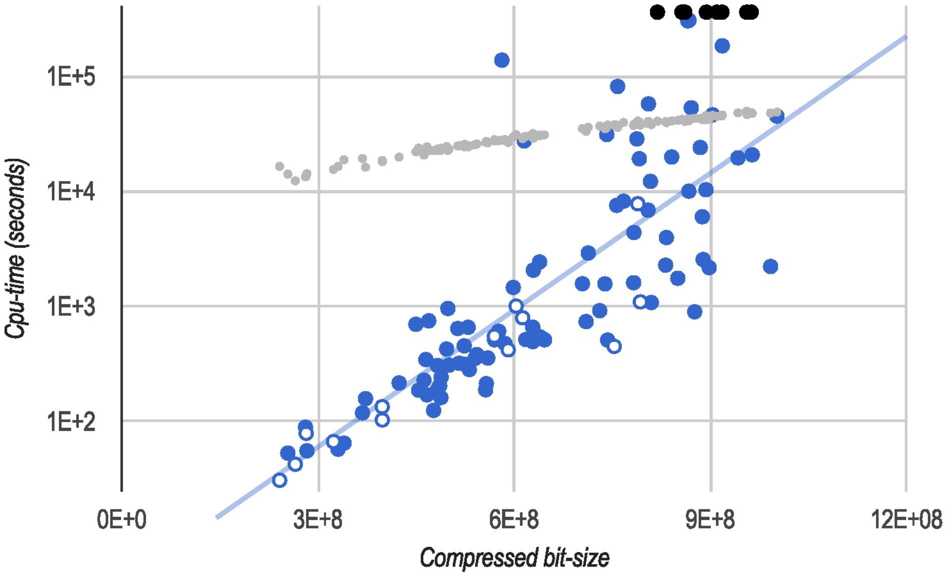


Figure 4: Execution times in seconds (Y logscale) as a function of problem bit-size. In blue: protein design problems that were solved to optimality (deterministic method). In black: design problems that could not be solved in 100 hours (deterministic method). In light gray, cpu-time for the 1,000 runs of Rosetta fixbb protocol. The exponential regression ($r^2 = 0.661$) over provenly solved protein designs (blue dots) is consistent with the expected worst-case exponential run-time of deterministic methods. Hollow blue dots are design cases for which Rosetta fixbb protocol could also find the optimum in 1,000 runs. The largest problem solved requires 15 Gbits uncompressed or 1 Gbits after strong Lempel-Ziv-Markov chain (LZMA2) lossless compression, as implemented in xz.[50]

Within the same time limit, on a single core of a 2012 generation CPU, the largest problem provenly solved in this paper has a sequence-conformation space size of 10^{234}. More than 80% of the solved problems needed less than 3 hours each and the smallest problem for which we could not prove optimality of the best conformation found in 100 hours has size 10^{206}. In the rest of the paper, we only consider those 98 design problems for which we could find the GMEC and prove its optimality.

This efficiency is the result of a variety of techniques inside our optimization tool. The pruning provided by the combination of Arc Consistency filtering both as pre-processing and during search and the good initial upper bound k on the GMEC energy are essential.

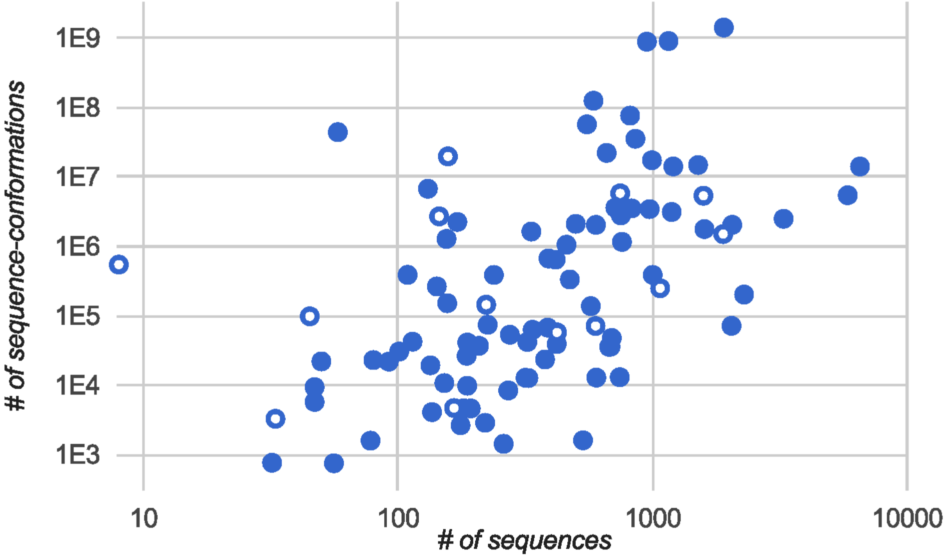


Figure 5: For each protein design for which the exhaustive enumeration of all sequence-conformations within 0.2 Rosetta units of the optimum finished in less than 100 hours, we give the number of sequences (X axis) and sequence-conformations (Y-axis) found. Hollow dots represent protein backbones for which Rosetta fixbb protocol could find the GMEC in 1,000 runs.

To explore the capabilities of our fixed-VAC-DFBB(EDAC,stat-k) enumeration algorithm, we tried to exhaustively enumerate all sequence-conformations within 0.2 Rosetta units of the energy of the GMEC. Because of the size of the problems, this small gap may already contain a very large number of sequence-conformations. In the same time-limit as above, the set of all sequence-conformations in this gap could be enumerated on 92 of the 98 designs. The number of different sequences in this gap ranges from 8 to 6,527, while the total number of sequence-conformations ranges from 752 to 1.42 10^{9}. This variety of behaviors is represented as a scatter-plot in Figure 5. The mean number of conformations per sequence varied from 3 to close to 1 million, indicating very different densities of states and levels of geometrical constrainedness on rotamers in the different backbones (see SI).

Assessment of stochastic methods

The ability to exactly solve problems of sizes beyond what was previously considered as feasible offers an unprecedented opportunity to evaluate the reliability of stochastic methods on challenging problems.[51] In the context of CPD, Rosetta fixed-backbone design protocol (fixbb) can be considered as a reference.[6,52] With the initial aim of estimating a probability of finding the optimum, we performed this protocol with default parameters for 1,000 runs on each protein backbone. For each run, we computed the distance of the resulting sequence-conformation to the optimum in terms of energy and in terms of sequence, in Hamming distance (number of mutations).

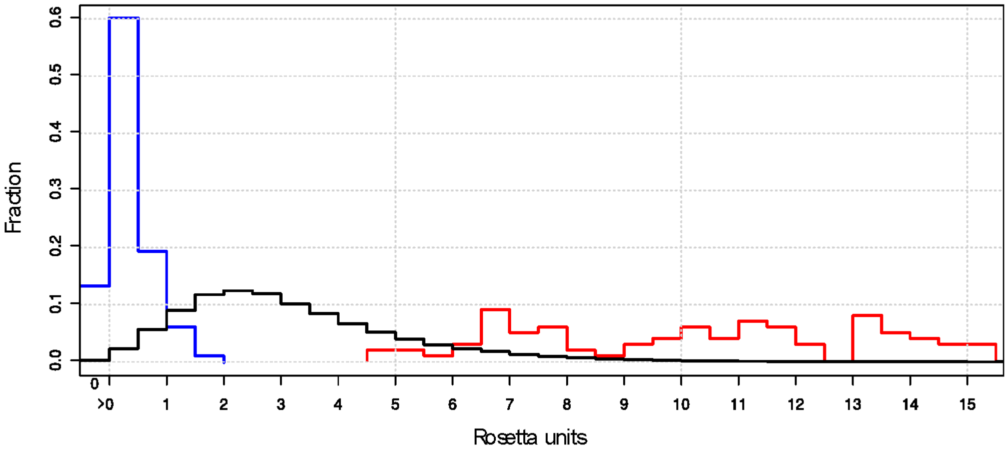


Figure 6: Histograms giving the distribution of the solutions found by the fixbb protocol by distance to the optimal energy. The first bucket in the histogram represents optimal solutions, with energy E^*. In blue, the optimistic set of the best energy solution obtained for each design problem over 1,000 runs of the fixbb protocol. In black, all solutions for all runs and all backbones. In red, the pessimistic set of all worst energy solutions obtained for each design problem over all 1,000 runs of the fixbb protocol. The X-scale has been truncated to 15 Rosetta-units but the worst difference to the optimum observed was equal to 37.5 Rosetta units.

With 1,000 runs, the Rosetta protocol could find the GMEC for 13 targets out of 98.

These 13 problems took a total of 36 hours to be solved to optimality by our guaranteed algorithm. The corresponding fixbb runs took a total of more than 90 hours, with no optimality proof.

In one additional case, the optimal sequence was found with a sub-optimal conformation. The distribution of the best, mean and worst energy among the 1,000 conformations found by Rosetta, in terms of distance to the optimum is presented in Figure 6. It shows that 1,000 runs of this protocol are not sufficient to get a reliable access to the optimal solution on these full redesigns. For one backbone, the best Rosetta solution in the 1,000 runs was almost 2 Rosetta units away from the optimum.

The strength of simulated annealing search is that it always provides a result, despite the exponentially increasing conformation space and the problem NP-hardness. Is there a price to pay for this ability? In Figure 7, we present the best and mean distance to the optimum for each protein backbone over the same 1,000 runs, as a function of the size of the sequence-conformation space. One can observe a clear tendency for this mean gap to grow with the log-size of the sequence-conformation space and thus the number of residues to design. From this point of view, the price paid for the efficiency of simulated annealing is an empirically growing expected optimality gap.

The initial aim of this experiment was to empirically estimate the probability of the fixbb protocol to find the GMEC on our set of benchmarks. Figure 8 represents all benchmarks ordered according to their hardness for the Rosetta fixbb protocol, as indicated by the empirical frequency of finding the optimum, when non zero, or as the remaining gap to optimality after 1,000 runs otherwise (see the SI for a more detailed representation). On our problems of gradually increasing sizes from 50 to 100 residues, we observe a very rapid drop in frequency, reaching 0.001 after just 7 problems. This is followed by a steadily increasing gap to optimality. An extrapolation of this trend to the right region would imply that the probability of finding the GMEC on the hardest problems becomes rapidly extremely small.

We more specifically focused on two hard protein design problems maximizing either the

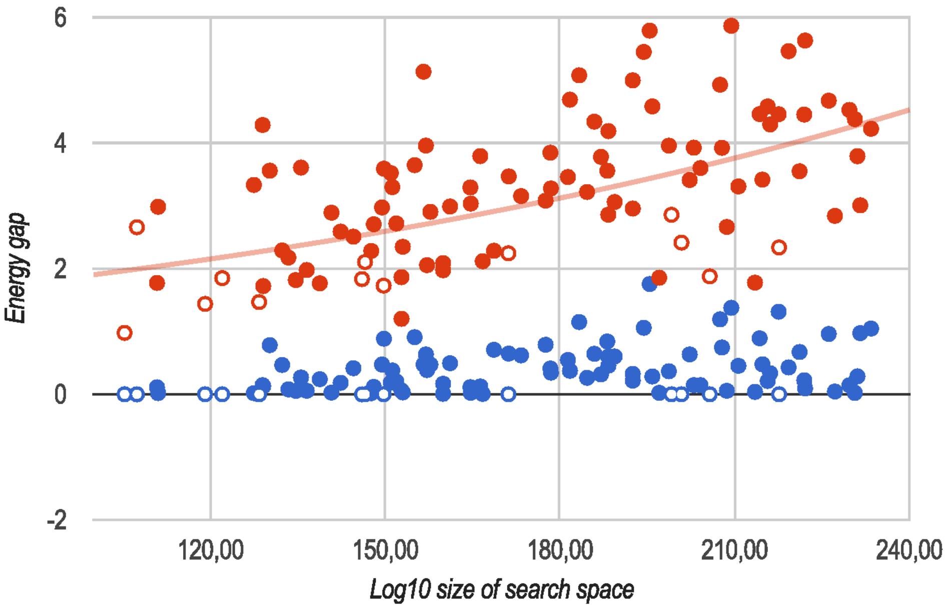


Figure 7: Distance to the optimum energy for the best (blue) and mean (red) energy over 1,000 fixbb runs, as a function of the $\log_{10}$-conformation space. An exponential regression ($r^2 = 0.318$) fit is shown (marginally better than a linear – $r^2 = 0.309$ – also increasing fit). Hollow dots represent protein backbones for which the optimum was found in 1,000 runs.

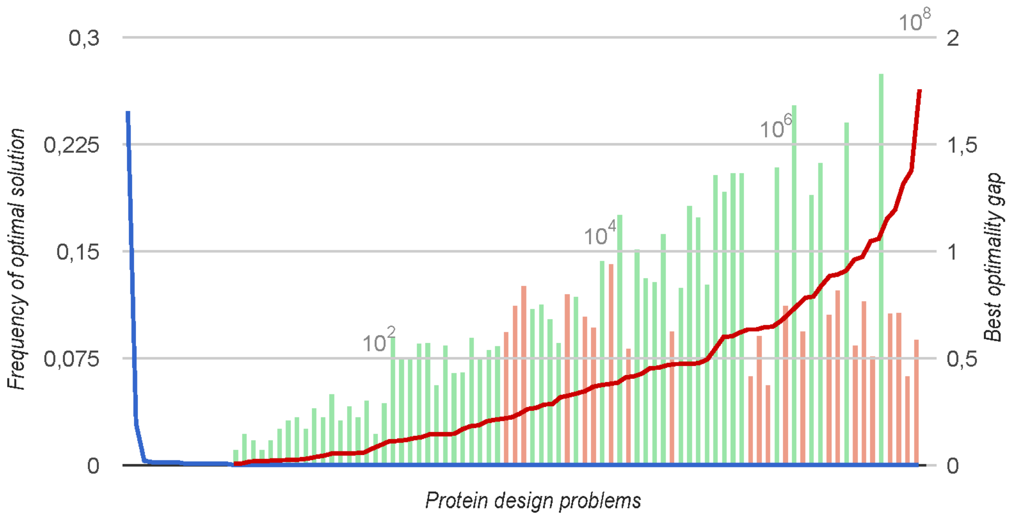


Figure 8: In blue, we give the empirical frequency of correct identification of the optimum by fixbb (left scale). In red (right scale), we indicate the smallest gap to optimality of all remaining protein backbones over 1,000 fixbb runs. Design problems are ordered in decreasing order of frequency and then increasing order of energy gaps. In the background, we represent lower bounds or exact number of unique sequences with energy between the GMEC and fixbb best solution using a logarithmic scale (in gray). A green box indicates an exact count, a red box the lower bound provided by the 0.2 Rosetta units enumeration.

mean or the minimum energy gap over $1,000$ fixbb samples (PDB ID: 2CJJ and 2CKX). We executed fixbb $1,000,000$ times on these (representing around 2 years of CPU-time). We used these samples to compute the empirical cumulative density function $cdf(\theta) = p(E(X) < \theta)$ as the fraction of samples with energy less than θ in the sample. Exploiting the fact that the minimum of p i.i.d. variables sampled with a given $cdf(\theta)$ has a cumulative density function defined by $(1-(1-cdf(\theta))^p)$, it is possible to compute the empirical average energy gap that would be obtained after p iterations of fixbb. These averages are represented in Figure 9 and show that the improvement in energy rapidly decreases to almost 0. Since the curves are convex, a linear extrapolation of the curves will yield an optimistic lower bound on the time needed to get the GMEC. For `2CJJ`, with a final slope of $1.6\ 10^{-2}$ Rosetta units per million samples, this bound is equal to $1.8\ 10^7$, representing more than 20 years of cpu-time. It is only of $5.6\ 10^6$ samples for `2CKX`. This is probably caused by the optimistic linear extrapolation: the curve being further away from the optimum, it decreases faster, by $9.3\ 10^{-2}$ Rosetta unit per million of samples.

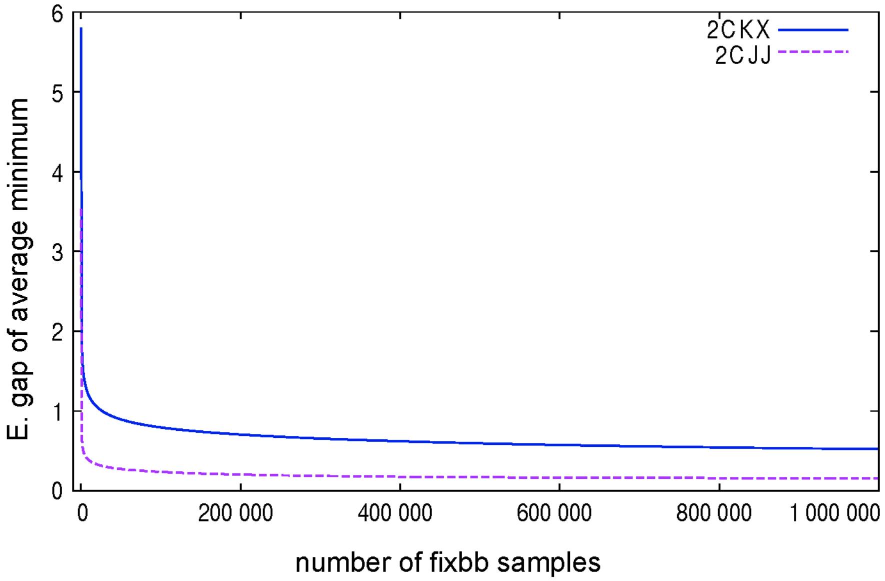


Figure 9: The average value of the gap to the optimum of the minimum of p fixbb samples (Y axis) as p increases (X-axis), on the two hard protein design problems 2CJJ and 2CKX, empirically estimated from 1,000,000 fixbb samples.

Sequence analysis

The Rosetta fixbb protocol also defines a distribution over the distance of a predicted sequence to the optimal sequence. To evaluate this induced distribution at the sequence level, we computed the Hamming distance of the predicted sequences with the optimal sequence for each backbone on either the minimum energy fixbb design, on all designs and on the design maximizing the Hamming distance. This is represented in Figure 10. If we consider only the best Rosetta model over 1,000 runs (in terms of energy), we can see that the designed sequence may exceptionally be up to 30% away from the optimum sequence. Over all design problems, the best model sequence disagrees with the GMEC for approximately 8% of all residues. The core of all proteins is however well conserved, with a difference of 2.4% in the core, 7% in the boundary and close to 10% on the surface. Core residues (as defined in Rosetta) only account for a small fraction of all residues in our benchmark (10%).

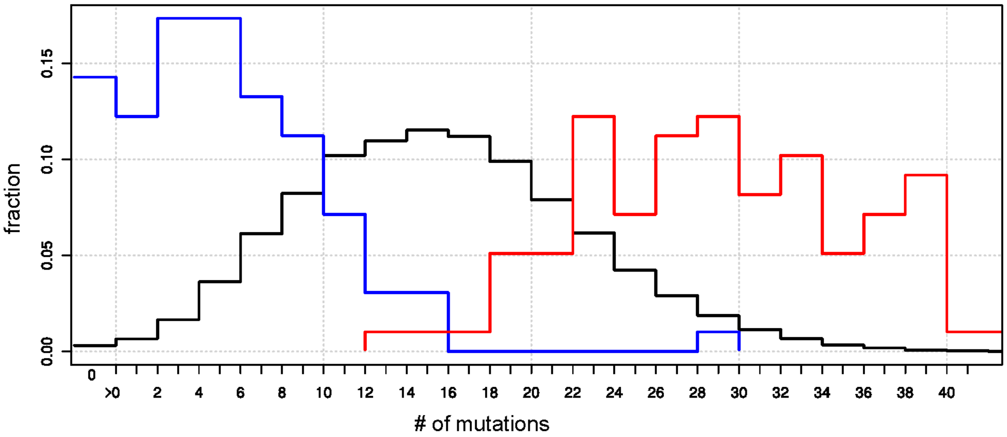


Figure 10: Histogram of the optimal sequence recovery rate of Rosetta fixbb protocol for sequences with the best energy among the 1,000 runs (blue, best case), among all sequences (black, mean case) and among sequences with the largest Hamming distance to the optimal sequence (red, worst-case).

The guaranteed optima give a direct access to the implicit side-effects of the scoring function. Table 1 shows the number of hydrophobic, polar, charged and aromatic amino acids contained in the wild type, in the best Rosetta model and in the GMEC. Our interest in the wild type sequence follows from the use of its sequence as a guide to improve energy functions.[2,53]

Table 1: Distribution of different amino acid types in the native sequence, in Rosetta best solution, in all Rosetta solutions, in the GMEC sequence, and in all sub-optimal sequences within 0.2 Rosetta units of the GMEC on the 92 designs where the enumeration finished. Non mutable cysteines involved in disulfide bridges are excluded from these counts.

Type	native	fixbb best	fixbb all	GMEC	sub-optimal
Charged	1,795	1,996	1,818.33	2,097	2,090.13
Aromatic	585	616	611.235	622	624.79
Polar	1,817	1,730	1,864.46	1,662	1,662.84
Hydrophobic	2,585	2,440	2,487.99	2,401	2,404.23

One can see that compared to the native sequence, Talaris2014 favors charged and aromatic amino acids at the expense of the polar and hydrophobic residues.[54] Removing the effect of stochastic sampling, the GMEC shows these biases are stronger than estimated otherwise and remain essentially unchanged in the enumerated sub-optimal solutions. Without this noise, it should be easier to improve the energy weights as it has been done before.[2]

Sub-optimal sequences are clearly much closer to the GMEC than to the native sequence and this is confirmed by an asymptotic test that cannot reject the assumption that the proportion of each class in each protein comes from the same distribution in the GMEC and sub-optimal sequences (see SI).

Conclusion

Often, drastic progress in the ability to solve problems can unlock access to new questions. This progress may come from new hardware, as recently highlighted by the creation of the Anton computer for molecular dynamics.[55] In this paper, we have shown how recent algorithmic progress in Artificial Intelligence optimization techniques allows us to provide a guaranteed global minimum energy conformation for much larger redesign problems than was previously possible. On most of the solved problems, we have also been able to exhaustively enumerate all conformations within the neighborhood of the optimum, providing a precise local image of the density of states around the GMEC.[23]

Full protein redesign has previously been used as a basis for tuning the energy function.[21] Since the GMEC is the true meaning of the energy function, and given the qualitative differences we observed between it and the best fixbb sampled sequences (e.g., in term of compositional bias), these algorithmic advances could be a new cornerstone of energy function tuning.

Our benchmark uses rigid backbones and discrete rotamers. But protein are continuously flexible molecules. Accounting for continuous rotamers and/or backbone flexibility in the computational model has been shown to be more accurate.[21] Such flexible models are however even more challenging to solve. Existing provable algorithms for such models also directly rely on discrete pairwise decomposable energy optimization problems and should directly benefit from these results.

CFN algorithms are not limited to solving discrete pairwise decomposable optimization problems. They can deal with non-pairwise cost functions, including so-called global cost functions that involve an arbitrary number of variables[56] and can handle constraints. These features will probably be specifically handy for multi-state protein design.[22] On a longer term, in the right hands, they may find numerous other usages we cannot envision.

Because the GMEC identification problem is NP-hard, it has become usual and accepted to rely on stochastic optimization algorithms, such as simulated annealing, to identify ”close to optimum” solutions. For the first time, we have been able to measure the rate of success of the simulated annealing based fixbb protocol of Rosetta protein modeling suite on very large problems, with previously unreachable guaranteed optima. These results show that the probability of finding the GMEC drops very quickly close to 0 as problems get harder. Even with 1,000 runs, the Monte Carlo algorithm was sometimes unable to get within 1 Rosetta units of the optimum, leading to design sequences that could differ from the optimal sequence by more than 30% of their amino-acids.

Over the full set of all solved backbones, we observed that the CPU-time of the deter-

[1] The 1PGA backbone used in a previous study[2] is close to the 1PGX backbone in our benchmark. Its GMEC is provenly identified but not produced in 1,000 runs of fixbb protocol.

ministic method tends to increase exponentially, defining a clear limit to the maximum size of the problems that can be currently solved. Over hundreds of thousands of fixbb samples, we similarly observed that the mean energy gap to optimality tends to increase with the number of designed residues, putting a limit on the size of systems for which a reasonably good solution can be found with confidence. For the hardest problems, millions of samples did not allow to close the gap.

For a large fraction of significant discretized protein design problems, it may now be the case that guaranteed optimal sequence-conformations can be produced more efficiently than heuristic solutions.

Acknowledgement

This work was partially supported by the Agreenskill program (D. Simoncini PostDoc funding). We are grateful to the Genotoul (Toulouse) Bioinformatic platform for providing us computational support for this work.

Supplementary Information (SI) is available with this paper. It includes (1) an exhaustive spreadsheet including, for each protein design problem, all the different characteristics of the problem, its representation, and the results of CFN and fixbb algorithms (2) an associated document with precise references to source repositories where the algorithms are available, Talaris2014 weight parameters, description of the spreadsheet, statistical test definition and results on all classes and samples, proof of the cumulative density function of minimum, and a more detailed box-plot based representation of the contents of the fixbb samples. This information is available free of charge via the Internet at http://pubs.acs.org.

References

(1) Khoury, G. A.; Smadbeck, J.; Kieslich, C. A.; Floudas, C. A. *Trends Biotechnol.* **2014**, *32*, 99–109.

(2) Alvizo, O.; Mayo, S. L. *Proc. Natl. Acad. Sci. U.S.A.* **2008**, *105*, 12242–12247.

(3) Shapovalov, M. V.; Dunbrack, R. L. *Structure* **2011**, *19*, 844–858.

(4) Pierce, N. A.; Winfree, E. *Protein Eng.* **2002**, *15*, 779–782.

(5) Kirkpatrick, S.; Gelatt, C. D.; Vecchi, M. P. *Science* **1983**, *4598*, 671–680.

(6) 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. *Methods Enzymol.* **2011**, *487*, 545–574.

(7) Chowdry, A. B.; Reynolds, K. A.; Hanes, M. S.; Voorhies, M.; Pokala, N.; Handel, T. M. *J. Comput. Chem.* **2007**, *28*, 2378–2388.

(8) Desmet, J.; De Maeyer, M.; Hazes, B.; Lasters, I. *Nature* **1992**, *356*, 539–542.

(9) 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.; Donald, B. R. *Methods Enzymol.* **2013**,*523*, 87–107

(10) Kuhlman, B.; Baker, D. *Proc. Natl. Acad. Sci. U.S.A.* **2000**, *97*, 10383–10388.

(11) Kuhlman, B.; Dantas, G.; Ireton, G. C.; Varani, G.; Stoddard, B. L.; Baker, D. *Science (New York, N.Y.)* **2003**, *302*, 1364–1368.

(12) Siegel, J. B.; Zanghellini, A.; Lovick, H. M.; Kiss, G.; Lambert, A. R.; St Clair, J. L.; Gallaher, J. L.; Hilvert, D.; Gelb, M. H.; Stoddard, B. L.; Houk, K. N.; Michael, F. E.; Baker, D. *Science (New York, N.Y.)* **2010**, *329*, 309–313.

(13) Hong, E.-J.; Lippow, S. M.; Tidor, B.; Lozano-Pérez, T. *J. Comput. Chem.* **2009**, *30*, 1923–1945.

(14) Biddle, J. C. Methods and Applications in Computational Protein Design. M.Sc. thesis, Massachusetts Institute of Technology, 2010.

(15) Traoré, S.; Allouche, D.; André, I.; de Givry, S.; Katsirelos, G.; Schiex, T.; Barbe, S. *Bioinformatics* **2013**, *29*, 2129–2136.

(16) Zhou, Y.; Wu, Y.; Zeng, J. Computational Protein Design Using AND/OR Branch-and-Bound Search. In *Proceedings of the International Conference on Research in Computational Molecular Biology*, Warsaw, Poland, 2015; Przytycka, TM. Ed.; Springer Verlag: Berlin, 2015; 354–366.

(17) Allouche, D.; André, I.; Barbe, S.; Davies, J.; de Givry, S.; Katsirelos, G.; O'Sullivan, B.; Prestwich, S.; Schiex, T.; Traoré, S. *Artif. Intell.* **2014**, *212*, 59–79.

(18) Klepeis, J.; Floudas, C.; Morikis, D.; Tsokos, C.; Lambris, J. *Ind. Eng. Chem. Res.* **2004**, *43*, 3817–3826.

(19) Zhu, Y. *Ind. Eng. Chem. Res.* **2007**, *46*, 839–845.

(20) 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. *J. Chem. Theory Comput.* **2015**, *11*, 609–622.

(21) Hallen, M. A.; Keedy, D. A.; Donald, B. R. *Proteins* **2013**, *81*, 18–39.

(22) Ambroggio, X. I.; Kuhlman, B. *J. Am. Chem. Soc.* **2006**, *128*, 1154–1161.

(23) Silver, N. W.; King, B. M.; Nalam, M. N.; Cao, H.; Ali, A.; Kiran Kumar Reddy, G.; Rana, T. M.; Schiffer, C. A.; Tidor, B. *J. Chem. Theory Comput.* **2013**, *9*, 5098–5115.

(24) Meseguer, P.; Rossi, F.; Schiex, T. Soft Constraint Processing. *Handbook of Constraint Programming*; Rossi, F., van Beek, P., Walsh, T., Eds.; Elsevier: Amsterdam, The Netherlands, 2006; 281–328.

(25) Goldstein, R. F. *Biophys. J.* **1994**, *66*, 1335–1340.

(26) Georgiev, I.; Lilien, R. H.; Donald, B. R. *J. Comput. Chem.* **2008**, *29*, 1527–1542.

(27) Cooper, M.; de Givry, S.; Sanchez, M.; Schiex, T.; Zytnicki, M.; Werner, T. *Artif. Intell.* **2010**, *174*, 449–478.

(28) Lawler, E.; Wood, D. *Oper. Res.* **1966**, *14*, 699–719.

(29) Jégou, P.; Terrioux, C. *Artif. Intell.* **2003**, *146*, 43–75.

(30) de Givry, S.; Schiex, T.; Verfaillie, G. Exploiting Tree Decomposition and Soft Local Consistency in Weighted CSP. In *Proceedings of the National Conference on Artificial Intelligence*, Boston, MA, USA, 2006; Gil, Y.; Mooney RJ. Eds; AAAI Press: Palo Alto, California, 2006; 22–27.

(31) Dechter, R.; Mateescu, R. *Artif. Intell.* **2007**, *171*, 73–106.

(32) Cooper, M. C.; Schiex, T. *Artif. Intell.* **2004**, *154*, 199–227.

(33) Schiex, T. Arc consistency for soft constraints. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming*, Singapore, 2000; Dechter, R. Ed.; Springer Verlag: Berlin, 2000; 411–424.

(34) Larrosa, J.; de Givry, S.; Heras, F.; Zytnicki, M. Existential arc consistency: getting closer to full arc consistency in weighted CSPs. In *Proceedings of the International*

Joint Conference on Artificial Intelligence, Edinburgh, Scotland, 2005; Kaelbling, LP.; Saffiotti, A. Eds; Professional Book Center: Denver, Colorado, USA, 2005; 84–89.

(35) van Beek, P. Bactrack Search Algorithms, *Handbook of Constraint Programming*; Rossi, F., van Beek, P., Walsh, T., Eds.; Elsevier: Amsterdam, The Netherlands, 2006; 85–134.

(36) Dechter, R. Tractable Structures for Constraint Satisfaction Problems, *Handbook of Constraint Programming*; Rossi, F., van Beek, P., Walsh, T., Eds.; Elsevier: Amsterdam, The Netherlands, 2006; 209–244.

(37) Bertelé, U.; Brioshi, F. *Nonserial Dynamic Programming*; Academic Press: London, 1972.

(38) Dechter, R. *Constraints* **1997**, *2*, 51–55.

(39) Brooks III, C. L.; Pettitt, B. M.; Karplus, M. *J. Chem. Phys.* **1985**, *83*, 5897–5908.

(40) Larrosa, J. Boosting search with variable elimination. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming*, Singapore, 2000; Dechter, R. Ed.; Springer Verlag: Berlin, 2000; 291–305.

(41) Robertson, N.; Seymour, P. D. *J. Algorithms* **1986**, *7*, 309–322.

(42) Xu, J.; Berger, B. *J. ACM* **2006**, *53*, 533–557.

(43) Leaver-Fay, A.; Kuhlman, B.; Snoeyink, J. An adaptive dynamic programming algorithm for the side chain placement problem. In *Proceedings of the Pacific Symposium on Biocomputing*, Fairmont Orchid, Big Island of Hawaii, 2005; Altman, RB.; Dunke, AK.; Hunter, L. Eds; World Scientific Press: Singapore, 2005; 16–27.

(44) Jou, J. D.; Jain, S.; Georgiev, I.; Donald, B. R. BWM*: A Novel, Provable, Ensemble-Based Dynamic Programming Algorithm for Sparse Approximations of Computational

Protein Design. In *Proceedings of the International Conference on Research in Computational Molecular Biology*, Warsaw, Poland, 2015; Przytycka, TM. Ed.; Springer Verlag: Berlin, 2015; 154–166.

(45) Kjærulff, U. *Triangulation of graphs–algorithms giving small total state space*; R-90-091990; Univ. of Aalborg, Denmark, 1990.

(46) Boussemart, F.; Hemery, F.; Lecoutre, C.; Sais, L. Boosting systematic search by weighting constraints. In *Proceedings of the European Conference on Artificial Intelligence*, Valencia, Spain, 2004; De Mntaras, L.; Saitta, R. Eds.; IOS Press: Amsterdam, 2004; 146–150.

(47) Lecoutre, C.; Saïs, L.; Tabary, S.; Vidal, V. *Artif. Intell.* **2009**, *173*, 1592,1614.

(48) Berman, H. M.; Westbrook, J.; Feng, Z.; Gilliland, G.; Bhat, T.; Weissig, H.; Shindyalov, I. N.; Bourne, P. E. *Nucleic Acids Res.* **2000**, *28*, 235–242.

(49) Chaudhury, S.; Lyskov, S.; Gray, J. J. *Bioinformatics* **2010**, *26*, 689–691.

(50) Ziv, J.; Lempel, A. *IEEE Trans. Inf. Theory* **1977**, *23*, 337–343.

(51) Voigt, C. A.; Gordon, D. B.; Mayo, S. L. *J. Mol. Biol.* **2000**, *299*, 789–803.

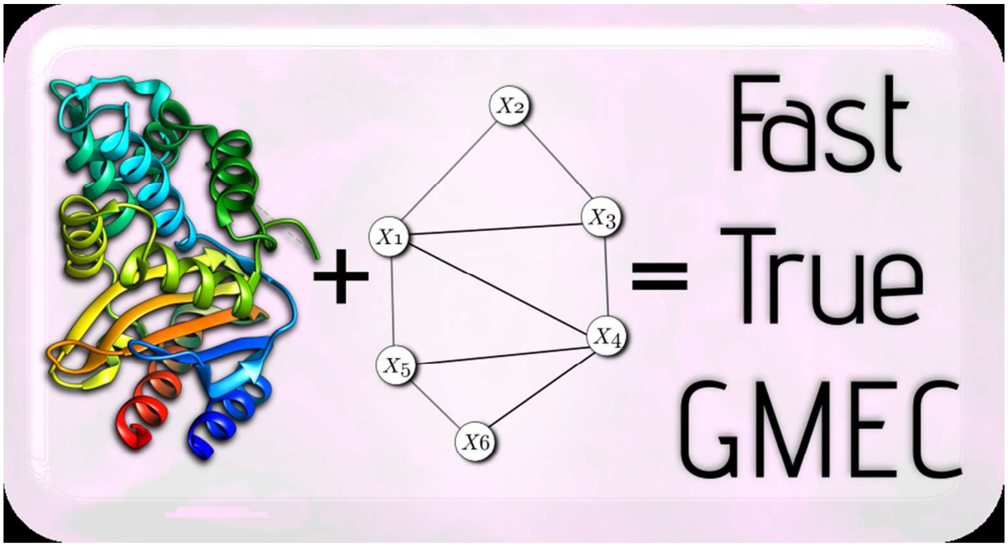
(52) Dantas, G.; Kuhlman, B.; Callender, D.; Wong, M.; Baker, D. *J. Mol. Biol.* **2003**, *332*, 449–460.

(53) Leaver-Fay, A.; O'Meara, M.; Tyka, M.; Jacak, R.; Song, Y.; Kellogg, E.; Thompson, J.; Davis, I. W.; Pache, R.; Lyskov, S.; Gray, J.; Kortemme, T.; Richardson, J.; Havranek, J.; Snoeyink, J.; Baker, D.; Kuhlman, B. *Methods Enzymol.* **2013**, *523*, 109.

(54) Jackson, E. L.; Ollikainen, N.; Covert III, A. W.; Kortemme, T.; Wilke, C. O. *PeerJ* **2013**, *1*, e211.

(55) Shaw, D. E.; Deneroff, M. M.; Dror, R. O.; Kuskin, J. S.; Larson, R. H.; Salmon, J. K.; Young, C.; Batson, B.; Bowers, K. J.; Chao, J. C.; Eastwood, M. P.; Gagliardo, J.; Grossman, J. P.; Ho, C. R.; Ierardi, D. J.; Kolossváry, I.; Klepeis, J. L.; Layman, T.; McLeavey, C.; Moraes, M. A.; Mueller, R.; Priest, E. C.; Shan, Y.; Spengler, J.; Theobald, M.; Towles, B.; Wang, S. C. Anton, a special-purpose machine for molecular dynamics simulation. In *Proceedings of the Annual International Symposium on Computer Architecture*, San Diego, California, USA, 2007; Tullsen, DM.; Calder, B. Eds.;007, ACM: New York, USA, 2007; 1–12.

(56) Allouche, D.; Bessiere, C.; Boizumault, P.; De Givry, S.; Gutierrez, P.; Loudni, S.; Métivier, J.-P.; Schiex, T. Filtering decomposable global cost functions. In *Proceedings of the National Conference on Artificial Intelligence*, Toronto, Ontario, Canada, 2012; Hoffmann, J.; Selman, B. Eds; AAAI Press: Palo Alto, California, 2012; 7–12.



Graphical Abstract
82x44mm (300 x 300 DPI)