Variable Neighborhood Search with Cost Function Networks To Solve Large Computational Protein Design Problems

Antoine Charpentier, David Mignon, Sophie Barbe, Juan Cortés, Thomas Schiex, Thomas Simonson, David Allouche

Variable Neighborhood Search with Cost Function Networks to Solve Large Computational Protein Design Problems

Antoine Charpentier,[†] David Mignon,[‡] Sophie Barbe,[¶] Juan Cortes,[$\mathcal{S}$]

Thomas Schiex,[*,†] Thomas Simonson,[‡] and David Allouche[*,†]

[†]*MIAT, Université de Toulouse, INRA, Castanet-Tolosan, France*

[‡]*Laboratoire de Biochimie (CNRS UMR 7654), École Polytechnique, 91128 Palaiseau, France*

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

[$\mathcal{S}$]*LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France*

E-mail: thomas.schiex@inra.fr; david.allouche@inra.fr

Abstract

Computational Protein Design (CPD) aims to predict amino acid sequences that fold to a specific structure and perform a desired function. CPD depends on a rotamer library, an energy function and an algorithm to search the sequence/conformation space. Variable Neighborhood Search (VNS) with Cost function networks is a powerful framework that can provide tight upper bounds on the global minimum energy. We propose a new CPD heuristic based on VNS, where a subset of the solution space is explored (a "neighborhood"), whose size is gradually increased with a dedicated probabilistic heuristic. The algorithm was tested on 99 protein designs with fixed backbones, involving nine proteins from the SH2, SH3, and PDZ families. The number of mutating positions was 20, 30, or all amino acids, while the rest of the protein explored side chain rotamers. VNS

was more successful than Monte Carlo (MC), Replica Exchange MC and a heuristic steepest-descent energy minimization (SDH), providing solutions with equal or lower best energies in most cases. For complete protein redesign, it gave solutions 2.5 to 11.2 kcal/mol lower than the other approaches. VNS is implemented in the toulbar2 software. It could be very helpful for large and/or complex design problems.

Introduction

Computational protein design (CPD) is an important developing tool in biotechnology, with possible applications in drug design, green chemistry, biomaterials, bioenergy and beyond. It is also an attractive approach to understand protein structure-function relationships.

The method starts from a particular protein sequence and structure and amino acid positions are iteratively selected and allowed to mutate so as to optimize a desired property, such as protein stability or ligand binding. In recent years, CPD has led to proteins with optimized thermostability and solubility,[1,2] novel ligand-binding,[3–5] novel enzyme activity,[6–8] and completely redesigned sequences.[9]

CPD methods are characterized by an energy function, a description of conformational space, and a search method to explore sequences and conformations. If many amino acid positions are allowed to mutate, the problem becomes enormous, and so efficient energy functions and powerful search methods are required. Many applications rely on a molecular mechanics energy function and a simple implicit solvent model, such that the energy function takes the form of a sum over amino acid pairs. While recent works try to better represent the flexibility of proteins at the cost of increased computational complexity,[10] conformational space is usually defined by a fixed protein backbone and a discrete library of rotamers for each amino acid side chain. The unfolded state is usually treated implicitly, through a set of amino acid chemical potentials. Even with these drastic simplifications, the space to explore can be huge and the optimization problem remains NP-hard. Thus, with n mutating positions, twenty allowed amino acid types, an average of 12 rotamers per

side chain type as in the Tuffery rotamer library,[11] and a total of N amino acids, the number of possible states is $12^N \times 20^n$.

The goal of CPD is usually to identify a small collection of low energy solutions, which can be experimentally tested. For some applications, it is important to sample large numbers of low energy states, or to enumerate all the states within a given energy window. This allows thermodynamic properties to be estimated, such as ligand binding free energies or acid/base constants. This can be achieved using ε-guaranteed algorithms such as TopN[12] and K^*,[4,13] later improved with CFN algorithms[14–16] or extended to affinity-based sequence optimization.[17] Or it can be achieved using Monte Carlo based approaches[18–21] with only asymptotic guarantees in the limit of infinite sampling. For other applications, a few solutions are sufficient, which can then be refined or modified manually using expert knowledge and/or experiments. To evaluate the quality of sampled sequences and conformations, an important goal is to identify, when possible, the global minimum energy sequence and conformation, referred to as the GMEC. The GMEC is an important candidate solution in its own right, and it also serves as a reference, indicating the energy range that should be sampled by good solutions.

With a given energy function and conformational space, the quantity and quality of sampled states are determined by the exploration method. Several classes of methods exist. Simulated annealing with Monte Carlo (MC) is very common; by performing multiple runs one expects to sample a collection of low energy solutions, and to reach the GMEC in the limit of a very long run.[22] However, in practical situations, there are no guarantees and few theoretical results on convergence and the nature of the sampled ensemble. Related methods with similar properties include genetic algorithms[23] and a steepest descent heuristic (SDH[24–26]) with multiple restarts. A second class of methods uses MC or Replica Exchange MC (REMC).[19,27–29] These methods efficiently sample very large numbers of states according to a Boltzmann distribution, which provides rigorous thermodynamic properties. Although convergence to the Boltzmann distribution is only guaranteed in the infinite limit,[30] it is common to observe numerical convergence over long runs for many small- and medium-sized problems. Assuming convergence, one then obtains tight confidence intervals

for average properties such as rotamer probabilities, binding free energies or sequence profiles.

A third class of exploration methods aims to provably identify the GMEC and possibly states within a small energy window above it. Cost function networks[31–34] are one framework that can provably find the GMEC. However, the problem being NP-hard, success is not guaranteed within a reasonable computing time. In practice, state-of-the-art CFN algorithms use branch-and-bound search and tight incremental bounds to solve large design problems. The search is separated in two phases: in a first phase, increasingly good solutions are found until the optimal solution is found. In a second phase, search is proving that this solution is optimal (no better solution exists). When the search is interrupted because of a cpu-time limit, one usually says that optimality was not proven: for sure the last phase was not finished (possibly not even started). With the Rosetta energy function[35,36] and the Dunbrack rotamer library,[37] the full design problem, where all amino acids were allowed to mutate, could be solved for proteins of about 100 amino acids.[38] Comparable performances were obtained with a molecular force field and an EEF1 solvent model.[32] In recent experiments with the same force field but a different solvent model and rotamer library, the GMEC was usually found and proven for problems where 20 positions could mutate and up to 96 other could change rotamers.[29] For larger problems, CFN algorithms were not able to prove optimality.

In this work, we propose a new approach for large protein design problems that combines the capacity of CFN algorithms with a so-called *Variable Neighborhood Search* (VNS): a stochastic optimization algorithm originally defined in Operations Research.[39] The idea is to select a subset of amino acid positions and identify the GMEC for these using CFN algorithms, with the others held fixed. A new subset is then chosen, and so on, until the energy cannot be improved in a reasonable time. We will refer to the subset of positions as the *mutator subset* or simply the *mutators*. Given a current selection of mutators, its *neighborhood* designates its design space: the collection of designs that can be reached by changing the amino acid identity or rotamer of one or more mutators. The mutator optimizations are done with CFN. With a mutator number of one or two, VNS can be seen as MC in the low temperature limit. We will typically use larger mutator numbers, and will compare several methods to choose the mutators, which can be deterministic,

probabilistic, or structure-based.

Below, we describe the CPD context and the VNS procedure, including the CFN framework and the VNS search. Next, we evaluate several variants of the VNS method on nine small proteins from the SH3, SH2, and PDZ families. The VNS variants are compared to each other and to results obtained here and earlier with SDH, MC, and REMC.[29] Finally, we discuss the potential impact of the method in CPD.

Theoretical framework

The CPD problem

In this work, we assume a "pairwise decomposable" energy function,[40] a fixed protein backbone, a discrete library of amino acid side chain rotamers. The pairwise energy function describes the energy of the molecular system of interest as a sum of terms, each of which depends on the relative positions of at most two atoms or atom groups. In the CPD case, this reduces to pairs of side-chain conformations. In this context, CPD typically searches for sequences that optimize the *folding* energy, namely the difference between the energies of the folded and unfolded states. The unfolded state is usually not explicitly modeled but captured by a sum of fitted terms, each of which depends on the amino acid type at one position. With these assumptions, a sequence-conformation is defined by a vector r whose components r_i are integer numbers that indicate the side chain type and rotamer at amino acid position i. The folding energy of a sequence-conformation can be written as:

$$E(r) = E_{bb} + \sum_i E(r_i) + \sum_{i<j} E(r_i, r_j) \tag{1}$$

E_{bb} is the energy of the protein backbone. $E(r_i)$ describes intra side chain and side chain-backbone interactions, and contains a contribution from the unfolded state. $E(r_i, r_j)$ is a side chain-side chain interaction. All these terms can be pre-computed and stored in an energy matrix. In some applications, the backbone term E_{bb} may also include contributions from side chains that are not

allowed to move or mutate.

Cost Function Networks

Cost function networks were introduced in Constraint Programming for discrete optimization. The problem is to minimize a cost function that depends on many discrete variables,[41] a problem known as Weighted Constraint Satisfaction. The total cost is a sum of simpler terms, also called cost functions, that each depend on just a few variables. The problem of finding the GMEC can be reduced to this problem, as shown by the form of the CPD energy, Eq. (1). The CFN variables represent side chain types and rotamers. The CFN cost functions represent energy contributions, and the total cost function represents the folding energy. Formally, using the terminology of protein design, a CFN can be defined as follows:

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

- *a set R of rotamer variables r_i, where i is an amino acid position; $i \in \{1, \ldots, n\} \stackrel{\text{def}}{=} I$. Each r_i takes its values in a finite domain D_i (a list of side chain types and rotamers).*
- *a set W of cost functions w_S, each involving a set of rotamer variables $\{r_i | i \in S\}$, $S \subseteq I$.*

We denote $D^S = \prod_{i \in S} D_i$ the Cartesian product of the domains of all the variables indexed in $S \subseteq I$. A vector of D^I represents an assignment of all the variables (types and conformation), in other words a sequence-conformation. It will be denoted r_I or simply r. A vector of D^S will be denoted r_S.

The total cost $w(r)$ of a sequence-conformation r is defined as the sum over all cost functions:

$$w(r) = \sum_W w_S(r_S) \tag{2}$$

Here, r_S is the projection of r onto S, while $w(r)$ could also be written $w_I(r_I)$. The total cost can be identified with the CPD folding energy. A cost function that depends on a single variable, say $w_{\{i\}}$, is called a unary cost function; it corresponds to a term $E(r_i)$ in the folding energy. A

cost function that depends on two variables r_i, r_j is called a binary function; it corresponds to a term $E(r_i, r_j)$. The backbone energy E_{bb} can be viewed as a cost function that has no variable-dependence or equivalently depends on an empty set of variables: $E_{bb} \equiv w_{\varnothing}$. The optimization problem is to identify a sequence-conformation r that gives the minimum total cost.

Guaranteed CFN solving

The Weighted Constraint Satisfaction problem can be solved using various algorithms, the most usual being a guaranteed branch-and-bound algorithm[42,43] that combines best and depth-first searches with tight incremental lower bounds. These algorithms rely on the representation of the space D^I of sequence-conformations as a binary tree. The root of the tree is associated with the original CFN. Any node in the tree has two children obtained by choosing both a variable r_i and a value ρ in its domain. The left child is a new CFN with r_i set to ρ (D_i becomes $\{\rho\}$). The right child is a CFN where ρ has been removed (D_i becomes $D_i \backslash \{\rho\}$). The leaves of this tree correspond to CFNs with all variables assigned to a distinct state $r_I \in D^I$. Together, they represent the complete set D^I and their energies. Because this tree has guaranteed exponential size, it is never explored fully. Instead, the algorithm relies on the energy of the best solution (leaf) found so far as a global upper bound on the energy of the GMEC. It computes a lower bound on the energy of the CFN associated with the current node using so-called "local consistency-enforcing algorithms".[44] These algorithms reformulate a CFN in an equivalent CFN that gives direct access to a non-naive lower bound on the optimum energy. If at a given node, this local lower bound is equal to or larger than the global upper bound, the exploration of the branch is stopped. Depth-first search will then backtrack and explore the most recent unexplored branch, with the guarantee that the optimal solution cannot be missed. Figure 1 shows a complete binary tree of depth 4.

Partial tree search with limited discrepancies

The amino acid position i and the rotamer value ρ chosen for testing at each node determine the precise tree structure. In practice, the branch-and-bound algorithm spends only a minor fraction

of its time to find the optimal leaf and the rest to prove optimality. Because we are not focused on guaranteed optimization, we used a partial-tree search mechanism called "Limited Discrepancy Search" (LDS)[45] which we now rapidly describe: LDS assumes that for every position i, a rotamer value $\rho_0 \in D_i$ that most likely participates in a low energy solution can be determined heuristically. A branch in the tree that sets r_i to a rotamer different from ρ_0 is called a *discrepancy*. LDS is parameterized by a bound δ on the maximum number of discrepancies that are allowed in any explored branch. With $\delta = 0$, a single branch defined by the heuristically chosen rotamers is explored. With larger δ bounds, an increasingly large fraction of the tree can be explored. When $\delta = n$ (the number of positions in the protein sequence), all branches can potentially be explored. To obtain good solutions, Fontaine et al.[46] proposed setting the number δ of allowed discrepancies to three or less. This gave a good balance between computational cost and quality on several large benchmark problems. The size of the partial tree that can be explored scales as $(k+\delta)^{\delta+1}$,[47] where k is the number of mutators. In practice, because of the tight lower bounds provided by CFN local consistencies, only a tiny unpredictable fraction of this is explored. Beyond the lower bound, local consistencies also provide, for free, the preferred rotamer $\rho_0 \in D_i$ that is needed by LDS: this is the so-called "support" as defined by Cooper et al.,[44] in Definition 4.5. It is at least guaranteed to have a minimum one-body energy.

Variable Neighborhood Search overview

The fundamental mechanism driving VNS is a local search within the neighborhood of a given state, or sequence-conformation $r \in D^I$. Neighboring states are ones obtained by changing one or more types or rotamers of r. The amino acid positions allowed to mutate are the mutators. If the best state r' within the neighborhood has a better energy than r, it is adopted and the process is repeated. Such a local search will eventually be trapped in a local minimum. VNS then restarts, using a new, larger set of mutators and thus a larger neighborhood.

VNS iteratively performs three steps. In the first step, a subset K of $k(i)$ amino acid positions are selected randomly to be mutators. The number $k(i)$ is drawn from a sequence of integers, in-

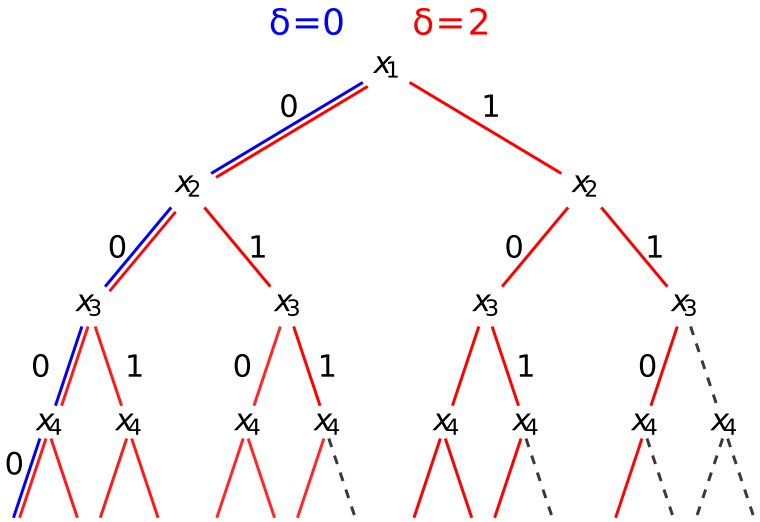

Figure 1: A binary search tree with four amino acid positions with two possible rotamer values each. At the root node, an unassigned position (r_1) is chosen and either assigned a chosen rotamer (left branch) or this rotamer removed from the domain (right branch). If one uses $\delta = 2$ in LDS, the dashed branches on the rightmost parts of the tree and subtrees can never be explored. For $\delta = 0$, the only branch explored always follows left branches, defining an initial "greedy" candidate solution, which energy already provides an upper bound on the GMEC energy.

dexed by i, which depends on the current iteration number. For example, $k(i)$ could be drawn from a Luby sequence, defined below. In the second step, a search in the corresponding neighborhood yields a new solution r'. In the third step, if r' has a lower cost, it replaces r and the number of mutators is reset to its initial value $k(0)$; otherwise, i is incremented and the next mutator number in the sequence will be used. The algorithm is shown in Fig. 2. For neighborhood exploration, we use a partial LDS, depth-first, branch-and-bound algorithm that exploits local consistencies for both energy bounds and the heuristic choice of rotamers.

Neighborhood growth

At the end of each VNS iteration, the number of mutators is modified according to the sequence $k(i)$. If the latest iteration was successful (energy improvement), i is reset to 0 and the algorithm shifts back to the beginning of the mutator number sequence. If not, i is increased. The simplest mutator number sequence is defined by $k(i+1) = k(i) + 1$. This is called the k^{++} sequence. Because LDS complexity can rapidly increase with the mutator number, it seems more suitable

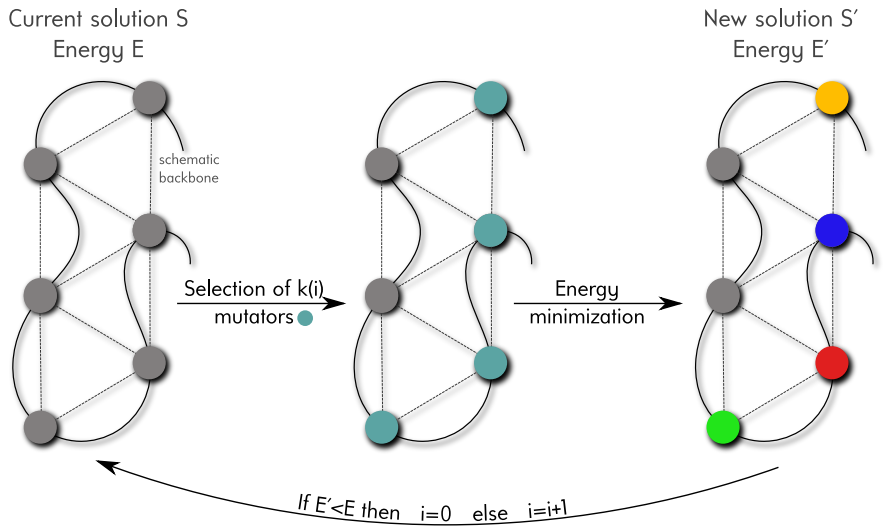

Figure 2: Variable Neighborhood Search procedure with mutator sequence $k(i)$. The protein backbone is shown on the left as a chain of gray beads, representing amino acid residues. In step (1), we choose a subset of $k(i)$ mutators (blue positions in the middle). In step (2), their amino acid types and rotamers are optimized, with the other positions fixed. The new types/rotamers are colored on the right. In step (3), if the energy has improved, the next iteration will use a mutator number $k(0)$. Otherwise, the next mutator number in the sequence will be used.

Algorithm 2: The VNS schema: an initial sequence-conformation r is generated by a greedy assignment of variables (line 1). VNS iterates until the mutator number k reaches n or the runtime exceeds a limit (line 3). $k(i)$ mutators are selected (line 4). The corresponding rotamer variables are optimized (the other rotamers being fixed) using limited discrepancy search. If the energy was improved, we shift back to the start of the mutator number sequence; otherwise, the next mutator number will be used.

Function `VNS/LDS+CFN` (R, W, δ)

1 $r \leftarrow$ `genInitSol()` `// initialize sequence-conformation`
2 $i \leftarrow 0$ `// i: progress in the mutator sequence`
3 **while** $(k \leq n) \wedge (not\ TimeOut)$ **do**
4 select a subset $K \subset I$ of $k(i)$ mutators `// their rotamers are` r_K
5 use `LDS+CFN.Search` to select new rotamers r'_K
`// The sequence-conformation is now` $r' = r'_K + r_{I \setminus K}$
6 **if** $w(r') < w(r)$ **then**
7 $r \leftarrow r'$
8 $i \leftarrow 0$ `// Shift back to initial mutator number`
9 **else**
$i \leftarrow i + 1$ `// next mutator number`
10 **return** r

to only increase it if smaller neighborhoods have been sufficiently explored. The simplest way to achieve this would be to increase the mutator number less frequently. With this aim, we have explored a new update rule based on the Luby series,[48] defined by:

$$\text{If } i+1 \text{ is a power of 2, and } i = 2^t - 1, \text{ then} \quad luby(i) = 2^{t-1}$$
$$\text{If not, and } 2^{t-1} \leq i < 2^t - 1, \text{ then} \quad luby(i) = luby(i - 2^{t-1} + 1)$$

where t is the solution of the equation in the left part of either of the conditions above. This expression defines the following sequence for $luby(i)$: {1,1,2,1,1,2,4,1,1,2,1,1,2,4,8,1,...} In this series, powers of 2 are spaced increasingly and irregularly. We then defined a mutator sequence $k_m(i)$, parameterized by its initial value $k_m(0)$ and a constant integer m. If $luby(i) = 2^m$, we increase the mutator number by one; otherwise, we keep the previous mutator number:

$$\text{If } luby(i) = 2^m, \quad k(i) = k(i-1) + 1$$
$$\text{Otherwise,} \quad k(i) = k(i-1)$$

With the smallest m value (m = 0), the increase of the mutator number is maximum, but it is already slower than the k^{++} rule, with a sequence of increments {1,1,0,1,1,0,0,1,1,0,1,1,0,0,0,1,...}. With a larger m, the mutator number increases even more slowly. With a slower increase, the probability is higher that any amino acid position will be considered as a mutator before larger neighborhoods are considered. Thus, local energy improvements are more likely to be found earlier.

Structure-based versus random mutator sets

The default VNS approach draws the k mutator positions from a uniform random distribution. In the CPD context, we expect that the protein 3D structure can guide a more efficient heuristic. The first mutator was always taken from a uniform distribution. The remaining mutators were chosen using a deterministic or a probabilistic scheme. The deterministic scheme used the *nearest* neighbors of the first mutator, based on the distances between side chain C_β atoms. In the probabilistic schemes, the remaining mutators were sampled either uniformly at *random* or with a probability

p_i that decreased with distance in the 3D protein structure:

$$p_i \propto \frac{1}{d_i} \tag{3}$$

where d_i is the distance of position i from the first mutator. This last heuristic will be denoted *d-prob*.

VNS heuristics

By combining a specific mutator number sequence with neighborhood growth heuristics, we obtained six different VNS heuristics. The mutator number sequence was given either by the k^{++} or the Luby-based sequences. The neighborhood growth was done either with a uniform *random* method, a *nearest* neighbor method, or a *d-prob* method. The various heuristics are represented as leaves of the tree in Figure 3.

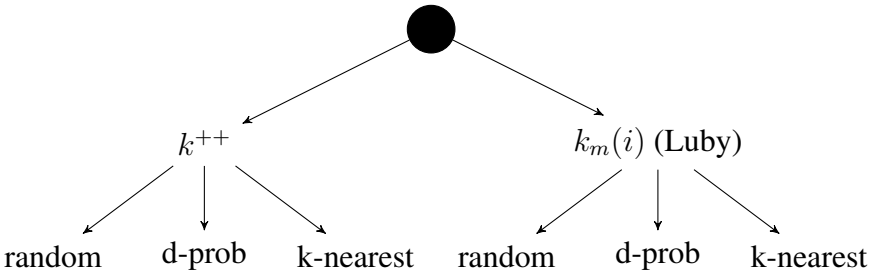

Figure 3: Representation of six different VNS heuristics defined by the mutator number sequence used and the neighborhood growth method used.

Numerical methods

Models and test cases

To evaluate VNS in the CPD context, we used test problems from earlier work.[29] They were based on nine protein domains, from the SH3, SH2 and PDZ structural families in Table 1. For each domain, the energy matrix was computed with the Proteus CPD package,[26] using the Amber ff99SB

molecular mechanics energy function, a Coulomb Accessible Surface Area (CASA) implicit solvent model[29,49] and the Tuffery rotamer library.[50] The model represents a good balance between accuracy and computational effort.[51] The first nine test problems (Full tests) corresponded to the full redesign of each protein, except for GLY and PRO residues, which were not mutated. Additional test problems corresponded to the redesign of 20 or 30 amino acid positions in each protein (N20 and N30 tests, respectively). Five choices of mutating positions were considered in each case. The positions were close together in the 3D structure. amino acids that were not allowed to mutate kept their wild type amino acid types, but were allowed to explore all possible rotamers. The test set contained $9 \times (5 + 5 + 1) = 99$ problems in total. The search space size for each problem is the product of the total number of types and rotamers allowed at each amino acid position. These sizes ranged from 10^{61} to 10^{229}.

Table 1: Protein set for testing.

Type	Name	PDB	Length	Vars.[a]
PDZ	NHERF	1G9O	91	76
PDZ	Syntenin	1R6J	82	72
PDZ	DLG2	2BYG	97	82
SH3	Abl	1ABO	56	48
SH3	Crk	1CKA	57	49
SH2	Syk	1A81	108	92
SH2	Grb2	1BM2	98	88
SH2	Zap70	1M61	109	88
SH2	Src	1O4C	104	96

[a] The number of amino acids, excluding Gly and Pro.

Numerical protocols

The six VNS heuristics were implemented in the toulbar2 CFN solver. The C++ code of toulbar2 is available under an MIT license at https://github.com/toulbar2/toulbar2. For each test problem, we applied VNS with the six heuristics described above (Fig. 3). The initial number of mutators was $k(0) = 4$ with a fixed discrepancy value of 3, a value set according to results previously obtained on a large collection of problems (not including these) in The Probabilistic Inference Challenge (PIC2011)(http://www.cs.huji.ac.il/project/PASCAL/). For the Luby based variants,

the parameter m was set to 3, based on short test runs on 20 instances randomly picked in the FULL, N20 and N30 set of the 1CKA and 1ABO proteins (over 990 overall). We imposed a CPU time limit of two hours. To estimate variability, ten runs were done for each test problem, with different initial random seeds.

The results were compared to those obtained by three other search methods:[29] Monte Carlo (MC), Replica Exchange Monte Carlo (REMC), and a multi-start, steepest-descent minimization heuristic (SDH). The results for the MC runs and the N20 and N30 REMC runs were presented in previous work. The SDH and the Full REMC runs are reported here. The SDH method uses a large number of "minimization cycles". In each cycle, a random sequence-conformation is chosen, then the energy is minimized, one residue at a time, with the others fixed in their current state. The entire sequence is scanned repeatedly this way until the energy no longer improves, ending the cycle.[29]

The new REMC runs used an improved selection of two-position moves (which provides better results than the selection strategy used earlier[29]). The new SDH runs used a slightly modified treatment of histidine residues, allowing multiple protonation states. All these results were obtained with the Proteus software,[26] using the following settings:

- **MC:** The simulation temperature was constant and corresponded to a thermal energy of kT = 0.2 kcal/mol. The simulations lasted 10^9 steps and never exceeded 9 hours.

- **Steepest descent heuristic (SDH):** Each run included 110,000 minimization cycles, except for a few cases that were extended to 330,000 or 990,000 cycles; run times did not exceed 24 hours.

- **REMC:** The REMC runs used eight replicas and 0.75 x 10^9 MC steps per replica. The thermal energies of the replicas were spaced in a geometric progression between 0.125 and 2 or 3 kcal/mol. The total CPU time per core was never more than 3 hours, for a total cpu-time on an 8-core machine of 24 hours at most.[29]

For each test we computed the difference $\Delta E = E_{\mathrm{Proteus}} - E_{\mathrm{VNS}}$ between the best energy obtained

with the Proteus protocols and the best VNS energy. A positive value indicates an improvement with VNS. For the nine Full tests, new REMC runs were performed here. The new runs used an improved selection of two-position moves (which provides better results than the selection strategy implemented in previous work[29]).

Results

Qualitative analysis

For a given VNS heuristic and category of problems (Full, N30, or N20), Table 2 reports the frequency of improvement ($\Delta E > 0$) and stagnation ($\Delta E = 0$) over all runs and test problems. All six VNS variants provided energies of the same or better quality than MC+REMC+SDH for most of the tests, especially for the larger tests. On the easiest category (N20), the energy was unchanged for about 2/3 of the tests and improved for about 30%, with a mean improvement of about 0.1 kcal/mol. Indeed, for most of the N20 tests, MC+REMC+SDH found the GMEC,[29] which cannot be improved. On the most difficult, Full category, the best VNS methods improved over MC+REMC+SDH in 94–99% of the cases, with a mean energy improvement of about 3 ± 1 kcal/mol. The best VNS energies, overall, were given by the Luby/d-prob heuristic.

Table 2: Frequency of energy improvement with VNS: each row represents a VNS heuristic and reports the percentage of runs in which the energy improved ($+$) or was unchanged (0) compared to SDH+MC+REMC.

	Full		N30		N20	
VNS	$+$	0	$+$	0	$+$	0
Luby/d-prob	98%	0%	65%	31%	31%	68%
Luby/random	99%	0%	63%	31%	31%	69%
Luby/nearest	94%	0%	60%	31%	28%	66%
k^{++}/d-prob	80%	0%	56%	27%	27%	66%
k^{++}/nearest	80%	0%	54%	21%	21%	62%
k^{++}/random	76%	0%	56%	25%	25%	67%

Quantitative analysis

For each VNS heuristic H and test problem P, we averaged the energy gain over the ten runs; the result is denoted $\overline{\Delta E}(H,P)$. The standard deviation over the runs is denoted $\sigma(\Delta E)(H,P)$:

$$\overline{\Delta E}(H,P) = \frac{1}{10}\sum_{r=1}^{10} \Delta E \qquad \sigma(\Delta E)(H,P) = \sqrt{\sum_{r=1}^{10} \frac{(\Delta E(r) - \overline{\Delta E}(H,P))^2}{9}} \tag{4}$$

For each heuristic H and problem category C, we computed the average of $\overline{\Delta E}(H,P)$ and $\sigma(\Delta E)(H,P)$ over the corresponding tests, denoted $\langle\overline{\Delta E}\rangle_C$ and $\langle\sigma(\Delta E)\rangle_C$. The results are given in Table 3 and shown as a scatter plot in Figure 4.

Table 3: Averages $\langle\overline{\Delta E}\rangle_C$ of the mean energy improvement, compared to MC+REMC+SDH, over the test category C, and average $\langle\sigma(\Delta E)\rangle_C$ of its standard deviation.

	Full		N30		N20	
VNS	$\langle\overline{\Delta E}\rangle_C$	$\langle\sigma(\Delta E)\rangle_C$	$\langle\overline{\Delta E}\rangle_C$	$\langle\sigma(\Delta E)\rangle_C$	$\langle\overline{\Delta E}\rangle_C$	$\langle\sigma(\Delta E)\rangle_C$
Luby/d-prob	3.17	1.03	0.77	0.35	0.15	0.01
Luby/random	2.95	1.06	0.76	0.32	0.15	0.01
Luby/nearest	2.78	1.36	0.69	0.33	0.13	0.07
k^{++}/d-prob	0.77	2.40	0.59	0.47	0.13	0.11
k^{++}/nearest	1.03	2.21	0.43	0.59	0.10	0.20
k^{++}/random	0.55	2.47	0.58	0.48	0.14	0.05

Luby/d-prob is the best method, offering both quality and reliability in most test categories and especially in the most challenging "Full" category. In this category, the results of k^{++} and Luby are clearly separated. Luby outperform k^{++}, providing better energies and smaller standard deviations, especially with the structure-based neighborhood growth heuristic d-prob.

For the N20 set, we used toulbar2 to identify proven GMECs using options `-dee: -d: -l=3 -m=2`.[32] Using the best solution provided by the VNS heuristics as an initial upper bound, toulbar2 was able to prove the optimality of 36 out of the 45 instances within a 72 hours time-out but never improved over the best VNS solution. Over this set of 36 instances with a proven GMEC, VNS with Luby/d-prob found the GMEC in 357 of 360 runs (99.1%). For the 3 remaining designs, the best solution could not be improved and no optimality proof reached within the 72

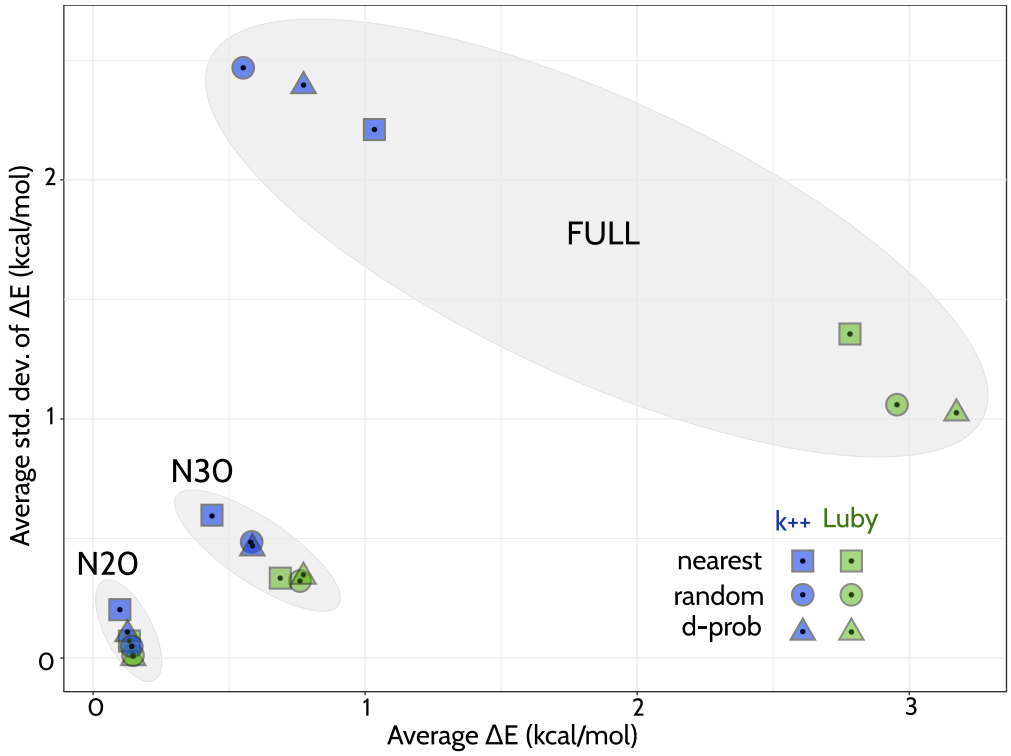

Figure 4: Average and standard deviation of the average energy improvement using different VNS heuristics and test categories. The average and standard deviation for a VNS method H and a test category C (Full, N30, N20) were computed over all the tests in C. Blue and green colors are respectively associated with k^{++} and Luby mutator growth methods. Squares, circles and triangles are respectively associated with k-nearest, random, and d-prob position selection. Each category is highlighted by an ellipsoid.

hours timeout. SDH reached the GMEC in 25 of the 36 cases. When all N20 instances are considered, VNS improved over (SDH+MC+REMC) in 14 cases, with improvements ranging from 0.04 to 1.39 kcal/mol. While the average improvement was very small over N20, there were no runs with $\Delta E < 0$.

VNS Luby/d-prob also gave the best results on N30, offering improved energies for almost 2/3 of the N30 tests, including nine improvements by more than 1.5 kcal/mol. The best energy improvement obtained over all runs and problems was 5.4 kcal/mol.

To see how the results of the best VNS (Luby/d-prob) distribute over the different test problems, we show a box-plot of the distribution of ΔE over the nine Full tests in Figure 5 (boxplots for the N20 and N30 datasets, on a per instance basis, are available as Figures S1 and S2). If we consider the best VNS runs (among the 10 runs per test), the energy improvements ranged from 1.3 to 11.2 kcal/mol. In 8 cases out of 9, the median energy obtained with VNS Luby/d-prob also improved over the best energy obtained by MC+SDH+REMC with a maximum of 11.2 kcal/mol for 1G90. The worst median ΔE was obtained on 1A81, with a value -0.30 kcal/mol. In this case, both the average and maximum improvements remained positive (respectively, 0.2 and 2.5 kcal/mol).

Optimality proof for side chain positioning

VNS does not provide an optimality proof for its best solution, which may not be the GMEC. As a partial optimality test, we considered the Full tests and asked whether for the obtained sequence, the obtained side chain rotamers were indeed optimal. Using Luby/d-prob, for every test problem, an optimal side chain positioning was found among the 10 runs. In the MC+REMC+SDH tests,[29] the optimal side chain rotamers were obtained only for five of the nine Full instances.

Over all problem categories and all runs, Luby/d-prob appears to improve conformational sampling, since 91% of all lowest-energy sequences corresponded to optimal side chain positions. As another test of reliability, we restarted sequence optimization from the best VNS solutions using the Hybrid best first search algorithm[42] (toulbar2 command line with no options) and with a previously published Depth First Search protocol[32] (both available in toulbar2). After 24 hours timeout

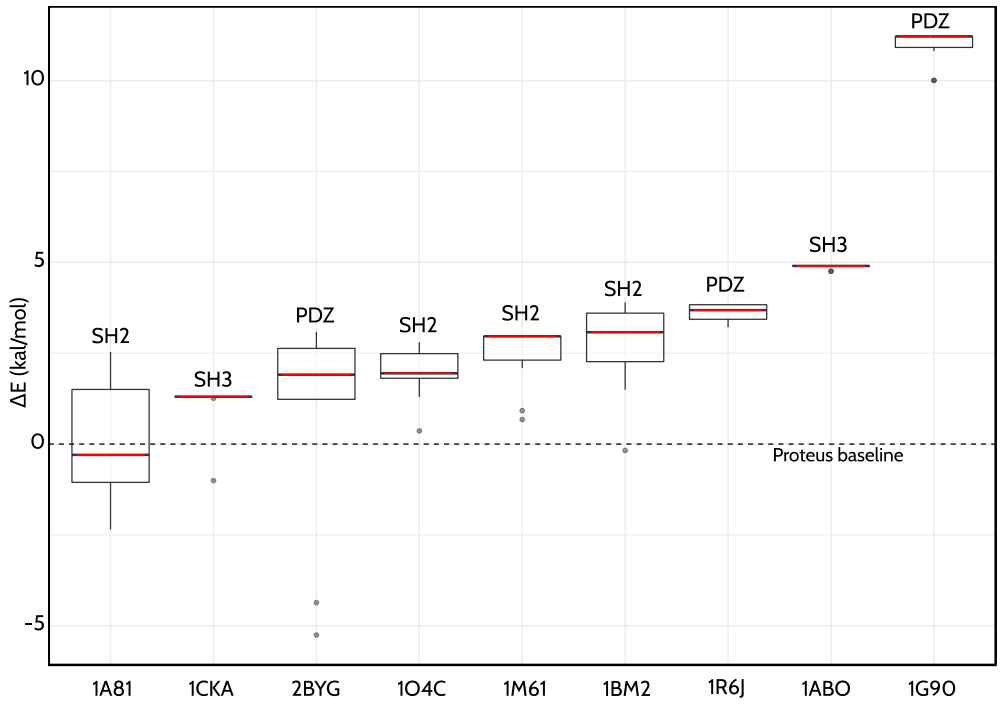

Figure 5: Box plot of ΔE values with VNS Luby/d-prob from ten runs on each Full test.

with each protocol, none of these did provide any energy improvement, suggesting that the VNS energies were already very good. No optimality proof could be obtained either.

Neighborhood growth speed

We compared the k^{++} and Luby neighborhood growth sequences during the VNS searches. We considered the nine Full problems. The histogram of mutator numbers used in the VNS searches is shown in Figure 6. With k^{++}, the mutator number increased rapidly to sizes that define very hard subproblems. Using Luby, with the parameter m = 3, neighborhood growth was slower and energy exploration more effective. The largest mutator sets included about 40 amino acids, which is fewer than the average number of residues within 14 Å of a given amino acid residue. With Luby, seven times more subproblems (or nodes) were explored on average. The largest mutator numbers were 49 and 40, respectively, with k^{++} and Luby. The median mutator number was 8 with both methods.

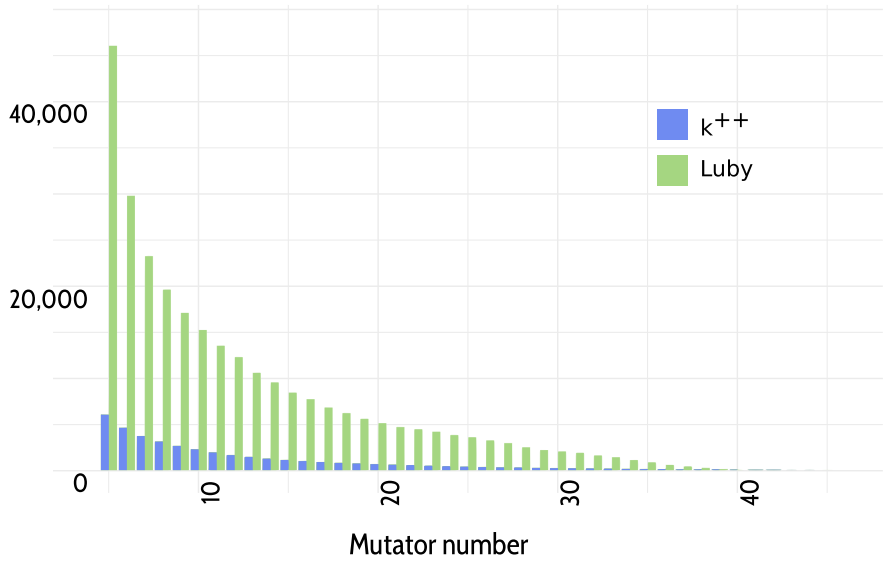

Figure 6: Histogram of the mutator numbers explored during the Full design tests using the k^{++} (blue) or Luby (green) sequences.

Computational efficiency

As shown in Figure 7, the average time by run is similar for all the VNS heuristics and is around 10, 15 and 50 minutes for respectively the N20, N30 and FULL cases. The use of the Luby-based heuristic has a limited effect on the run time. However, it does increase the probability of identifying the GMEC (without proving optimality). An indicative rate of best solution recovery is provided in Table S1.

With the Luby/d-prob strategy, and considering only "FULL" instances, Table 4 shows that the average time to obtain the best solution for the considered proteins ranges between 14 and 88 minutes. The observed variability is clearly related to the different sizes of the proteins and is very likely also influenced by the interaction terms defined by the energy functions for each of the considered proteins.

In comparison,[29] SDH and REMC methods maximum run times reached almost 24 hours of sequential cpu-time (REMC using at most 3 hours on 8 cores). MC maximum run time was close to nine hours.

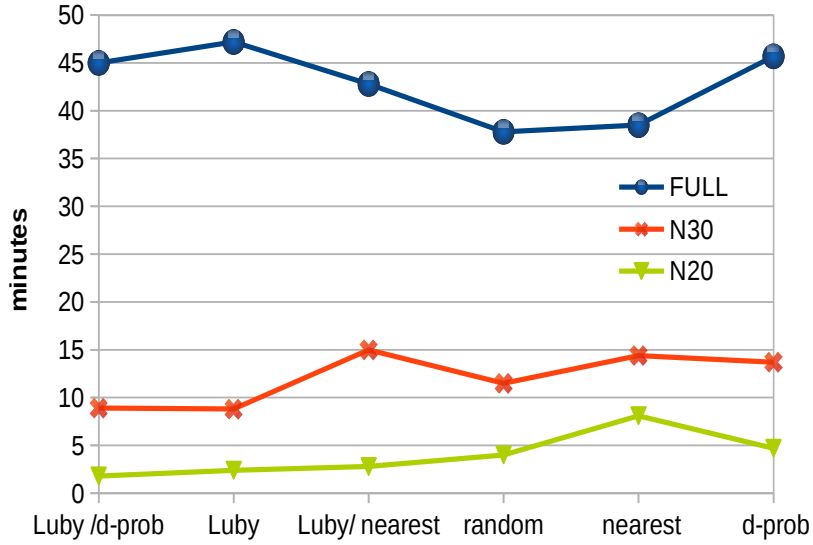

Figure 7: Average run times in minutes on FULL (blue circle), N30 (orange cross) and N20 (green triangle) for each considered heuristic.

Table 4: Family, PDB identified, problem size (number of variables) and CPU times (in minutes) for the full redesign of each protein, averaged over ten runs using Luby d-prob.

Family	PDZ	SH2	SH2	SH2	SH2	PDZ	PDZ	SH3	SH3
PDB	1A81	2BYG	1O4C	1M61	1BM2	1R6J	1G9O	1ABO	1CKA
#. vars	92	97	96	88	88	72	76	48	49
$\langle$ cpu $\rangle$	74±24	70±34	55±14	50±18	49±18	45±23	37±20	15±7	11±3

Concluding discussion

Cost function networks and their algorithms, implemented in Toulbar2, define a powerful framework for large CPD problems, able to provably obtain the GMEC for large problems in short times. For other problems that are too large for the GMEC to be found, we would nevertheless like to exploit the power of CFNs. This is done here by applying CFN optimization to a series of subproblems, whose size is gradually increased until the energy does not improve further. This iterative approach is referred to as Variable Neighborhood Search, and has been extensively used for other discrete optimization problems.[52,53] The key parameters of the method are the sequence of subproblem sizes and the detailed method for choosing the subproblems. Here, we defined a subset

of mutator positions and we increased their number with the help of a Luby integer sequence.

The best strategy for choosing the mutators was a random one (d-prob) that choses with a high probability positions close together in the 3D protein structure. This strategy appears to balance the need to distribute mutators throughout the structure and the need for concerted rearrangements of groups of nearby amino acids in order to optimize packing and lower the folding energy. In this respect, our VNS strategy contrasts with heuristic schemes like SDH, where individual positions are optimized, or Monte Carlo where one or two amino acids are changed at a time. The largest energy improvements in our tests occurred when there were 4–8 mutators. Thanks to the CFN optimization, these positions are rigorously optimized, and all relevant concerted rearrangements are taken into account.

The largest CPD problems where Toulbar2 identified the GMEC[38] used the Talaris force field distributed with Rosetta, where different distance cutoffs are applied to different energy terms, between 6 and 10 Å. The problems considered here used a larger energy cutoff, which presumably makes the energy surface more rugged and complex, hence the need for a complex, multi-position exploration. It may be that a simplified energy function with an aggressive cutoff could be used to guide the VNS method, for example to choose the detailed structure of the search tree. The full energy function would then be optimized with the help of the resulting tree. For non-globular proteins that have an elongated structure, a smaller energy cutoff could make the problem decomposable into subproblems that interact only through few amino acids. The VNS heuristics developed in this paper could be extended to exploit such situations.

Because of its increased efficiency, our method can be highly useful in various CPD contexts. The benchmarked problems used Tuffery's rotamer library[50] with 248 rotamers for a fully mutable residue. Since intensive CFN algorithms are only applied on relatively small subproblems in VNS, there is no visible reason that would preclude the use of VNS and the proposed heuristic with denser rotamer libraries[37] or energy-based conformer libraries.[54]

More generally, whether it is for a better modeling of the protein flexibility or to account for multiple desirable or undesirable states of the protein being designed,[55,56] computational protein

design will require to process increasingly harder and larger problems. In this context, our approach could be highly useful to handle the greatly increased search space of such refined models. This is also supported by the fact that VNS was recently parallelized.[57]

The VNS algorithm is not designed for the extensive enumeration of low-energy states, unlike Monte Carlo. Nevertheless, the visited sequences give information about local minima and the energy landscape of the system, through the collection of suboptimal solutions explored during the search. Another limitation of the VNS method is that the folding energy is optimized without considering alternative objectives, such as protein solubility or specific binding of other proteins. This defines multi-objective optimization problems, which are computationally difficult and often addressed with simple, heuristic methods. For example, to design ligand binding while maintaining protein stability, it is common to optimize the energy of the protein-ligand complex. A more rigorous approach would optimize the ligand binding free energy while imposing a limit on the folding energy. Such an approach was discovered and applied recently, with the help of adaptive importance sampling Monte Carlo.[21] Another example of a multi-objective optimization would be to optimize folding energy while also optimizing the folding of the associated messenger RNA, or while enforcing a DNA coding sequence that overlaps with another protein coding sequence.[58] The VNS framework offers possibilities along these lines, since it can take advantage of "constraint programming expressivity"[41] in order to enrich the CPD model with new types of constraints.

In conclusion, we have adapted the Variable Neighborhood Search algorithm to large protein design problems. We have shown that a slower growth of the mutator set and search space and a probabilistic mutator selection significantly improve the performance. Complex, concerted structure rearrangements are explored for each mutator thanks to the CFN framework. As CPD models become more realistic and complex, finding the GMEC and proving its nature will become increasingly intractable. In the context of very large problems and/or more complex physical models, the search approach described in this paper should be highly useful.

Acknowledgements

This work was granted access to the HPC resources on the TGCC-Curie supercomputer, the Computing mesocenter of Région Midi-Pyrénées (CALMIP, Toulouse, France) and the Geno-Toul (Toulouse) Bioinformatic platform. This work was supported by the EMERGENCE program of IDEX Toulouse (E-CODE project), the French National Research Agency (ANR-12-MONU-0015-03 for JC, TSi, SB, ANR-16-C40-0028 for TSc and DA) and the French National Institute for Agricultural Research (INRA).

Associated Contents

Supplementary materials are provided that describe more detailed information on the per-instance energy gap as well as the frequency of recovery of the best energy on each problem category. This information is available free of charge via the Internet at http://pubs.acs.org.

References

(1) Korkegian, A.; Black, M. E.; Baker, D.; Stoddard, B. L. Computational thermostabilization of an enzyme. *Science* **2005**, *308*, 857–860.

(2) Diaz, J. E.; Lin, C.-S.; Kunishiro, K.; Feld, B. K.; Avrantinis, S. K.; Bronson, J.; Greaves, J.; Saven, J. G.; Weiss, G. A. Computational design and selections for an engineered, thermostable terpene synthase. *Protein Sci.* **2011**, *20*, 1597–1606.

(3) Rudicell, R. S.; Kwon, Y. D.; Ko, S.-Y.; Pegu, A.; Louder, M. K.; Georgiev, I. S.; Wu, X.; Zhu, J.; Boyington, J. C.; Chen, X.; Shi, W.; Yang, Z.-y.; Doria-Rose, N. A.; McKee, K.; O'Dell, S.; Schmidt, S. D.; Chuang, G.-Y.; Druz, A.; Soto, C.; Yang, Y.; Zhang, B.; Zhou, T.; Todd, J.-P.; Lloyd, K. E.; Eudailey, J.; Roberts, K. E.; Donald, B. R.; Bailer, R. T.; Ledgerwood, J.; Mullikin, J. C.; Shapiro, L.; Koup, R. A.; Graham, B. S.; Nason, M. C.; Con-

nors, M.; Haynes, B. F.; Rao, S. S.; Roederer, M.; Kwong, P. D.; Mascola, J. R.; Nabel, G. J. Enhanced Potency of a Broadly Neutralizing HIV-1 Antibody In Vitro Improves Protection against Lentiviral Infection In Vivo. *J. Virol.* **2014**, *88*, 12669–12682.

(4) Roberts, K. E.; Cushing, P. R.; Boisguerin, P.; Madden, D. R.; Donald, B. R. Computational Design of a PDZ Domain Peptide Inhibitor that Rescues CFTR Activity. *PLoS Comput. Biol.* **2012**, *8*, 1–12.

(5) 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 Catal.* **2015**, *5*, 1186–1198.

(6) Smith, B. A.; Hecht, M. H. Novel proteins: from fold to function. *Curr. Opin. Chem. Biol.* **2011**, *15*, 421 – 426, Molecular Diversity.

(7) Jiang, L.; Althoff, E. A.; Clemente, F. R.; Doyle, L.; Röthlisberger, D.; Zanghellini, A.; Gallaher, J. L.; Betker, J. L.; Tanaka, F.; Barbas, C. F.; Hilvert, D.; Houk, K. N.; Stoffard, B. L.; Baker, D. De novo computational design of retro-aldol enzymes. *Science* **2008**, *319*, 1387–1391.

(8) Khare, S. D.; Kipnis, Y.; Takeuchi, R.; Ashani, Y.; Goldsmith, M.; Song, Y.; Gallaher, J. L.; Silman, I.; Leader, H.; Sussman, J. L. S.; Stoddard, B. L.; Tawfik, D. S.; Baker, D. Computational redesign of a mononuclear zinc metalloenzyme for organophosphate hydrolysis. *Nat. Chem. Biol.* **2012**, *8*, 294–300.

(9) Kuhlman, B.; Dantas, G.; Ireton, G. C.; Varani, G.; Stoddard, B. L.; Baker, D. Design of a novel globular protein fold with atomic-level accuracy. *science* **2003**, *302*, 1364–1368.

(10) 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: Struct., Funct., Bioinf.* **2013**, *81*, 18–39.

(11) Gaillard, T.; Panel, N.; Simonson, T. Protein side chain conformation predictions with an MMGBSA energy function. *Proteins: Struct., Funct., Bioinf.* **2016**, *84*, 803–819.

(12) Henrion, M. Search-based methods to bound diagnostic probabilities in very large belief nets. Seventh conference on Uncertainty in Artificial Intelligence. 1991; pp 142–150.

(13) 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. *J. Comput. Chem. 29*, 1527–1542.

(14) Viricel, C.; SIMONCINI, D.; Allouche, D.; De Givry, S.; Barbe, S.; Schiex, T. Approximate counting with deterministic guarantees for affinity computation. International Conference on Modelling, Computation and Optimization in Information Systems 2015. 2015; pp 165–176, Proceedings of the 3rd International Conference on Modelling, Computation and Optimization in Information Systems and Management Sciences.

(15) Viricel, C.; Simoncini, D.; Barbe, S.; Schiex, T. Guaranteed weighted counting for affinity computation: Beyond determinism and structure. International Conference on Principles and Practice of Constraint Programming. 2016; pp 733–750.

(16) Viricel, C.; de Givry, S.; Schiex, T.; Barbe, S. Cost function network-based design of protein–protein interactions: predicting changes in binding affinity. *Bioinformatics* **2018**, 733–750.

(17) Ojewole, A. A.; Jou, J. D.; Fowler, V. G.; Donald, B. R. BBK* (Branch and Bound Over K*): A Provable and Efficient Ensemble-Based Protein Design Algorithm to Optimize Stability and Binding Affinity Over Large Sequence Spaces. *J. Comput. Biol.* **2018**, *25*, 726–739, PMID: 29641249.

(18) Druart, K.; Bigot, J.; Audit, E.; Simonson, T. A hybrid Monte Carlo method for multibackbone protein design. *J. Chem. Theory Comput.* **2017**, *12*, 6035–6048.

(19) Villa, F.; Panel, N.; Chen, X.; Simonson, T. Adaptive landscape flattening in amino acid sequence space for the computational design of protein:peptide binding. *J. Chem. Phys.* **2018**, *149*, 072302.

(20) Polydorides, S.; Simonson, T. Monte Carlo simulations of proteins at constant pH with generalized Born solvent, flexible sidechains, and an effective dielectric boundary. *J. Comput. Chem.* **2013**, *34*, 2742–2756.

(21) Villa, F.; Mignon, D.; Polydorides, S.; Simonson, T. Comparing pairwise-additive and many-body generalized Born models for acid/base calculations and protein design. *J. Comput. Chem.* **2017**, *38*, 2396–2410.

(22) Van Laarhoven, P. J.; Aarts, E. H. *Simulated annealing: Theory and applications*; Springer, 1987; pp 7–15.

(23) 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.* **2007**, *28*, 2378–2388.

(24) Wernisch, L.; Héry, S.; Wodak, S. Automatic protein design with all atom force fields by exact and heuristic optimization. *J. Mol. Biol.* **2000**, *301*, 713–736.

(25) Schmidt am Busch, M.; Lopes, A.; Mignon, D.; Simonson, T. Computational protein design: software implementation, parameter optimization, and performance of a simple model. *J. Comput. Chem.* **2008**, *29*, 1092–1102.

(26) Simonson, T.; Gaillard, T.; Mignon, D.; Schmidt am Busch, M.; Lopes, A.; Amara, N.; Polydorides, S.; Sedano, A.; Druart, K.; Archontis, G. Computational protein design: The proteus software and selected applications. *J. Comput. Chem.* **2013**, *34*, 2472–2484.

(27) Yang, X.; Saven, J. G. Computational methods for protein design and protein sequence variability: biased Monte Carlo and replica exchange. *Chem. Phys. Lett.* **2005**, *401*, 205–210.

(28) Druart, K.; Palmai, Z.; Omarjee, E.; Simonson, T. Protein:ligand binding free energies: a stringent test for computational protein design. *J. Comput. Chem.* **2016**, *37*, 404–415.

(29) Mignon, D.; Simonson, T. Comparing three stochastic search algorithms for computational protein design: Monte Carlo, Replica Exchange Monte Carlo, and a multistart, steepest-descent heuristic. *J. Comput. Chem.* **2016**, *37*, 1781–1793.

(30) Grimmett, G. R.; Stirzaker, D. R. *Probability and random processes*; Oxford University Press, Oxford, United Kingdom, 2001.

(31) 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. Principles and Practice of Constraint Programming. 2012; pp 840–849.

(32) 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* **2013**, *29*, 2129–2136.

(33) 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. *Artif. Intell.* **2014**, *212*, 59–79.

(34) Traoré, S.; Roberts, K. E.; Allouche, D.; Donald, B. R.; André, I.; Schiex, T.; Barbe, S. Fast search algorithms for computational protein design. *J. Comput. Chem.* **2016**, 1048–1058.

(35) 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.* **2015**, *11*, 609–622.

(36) Alford, R. F.; Leaver-Fay, A.; Jeliazkov, J. R.; OaMeara, M. J.; DiMaio, F. P.; Park, H.; Shapovalov, M. V.; Renfrew, P. D.; Mulligan, V. K.; Kappel, K.; Labonte, J. W.; Pacella, M. S.;

Bonneau, R.; Bradley, P.; Dunbrack, R. L.; Das, R.; Baker, D.; Kuhlman, B.; Kortemme, T.; Gray, J. J. he Rosetta All-Atom Energy Function for Macromolecular Modeling and Design. *J. Chem. Theory Comput.* **2017**, *13*, 3031–3048.

(37) Shapovalov, M. V.; Dunbrack, R. L. A Smoothed Backbone-Dependent Rotamer Library for Proteins Derived from Adaptive Kernel Density Estimates and Regressions. *Structure* **2011**, *19*, 844–858.

(38) Simoncini, D.; Allouche, D.; de Givry, S.; Delmas, C.; Barbe, S.; Schiex, T. Guaranteed Discrete Energy Optimization on Large Protein Design Problems. *J. Chem. Theory Comput.* **2015**, *11*, 5980–5989.

(39) Mladenović, N.; Hansen, P. Variable Neighborhood Search. *Comput. Oper. Res.* **1997**, *24*, 1097–1100.

(40) Samish, I. The Framework of Computational Protein Design. *Methods Mol. Biol. (N. Y., NY, U. S.)* **2017**, *1529*, 3—19.

(41) Meseguer, P.; Rossi, F.; Schiex, T. *Handbook of Constraint Programming*; Elsevier, 2006; Chapter 9.

(42) Allouche, D.; De Givry, S.; Katsirelos, G.; Schiex, T.; Zytnicki, M. Anytime hybrid best-first search with tree decomposition for weighted CSP. International Conference on Principles and Practice of Constraint Programming. 2015; pp 12–29.

(43) 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* **2016**, *21*, 413–434.

(44) Cooper, M.; de Givry, S.; Sanchez, M.; Schiex, T.; Zytnicki, M.; Werner, T. Soft arc consistency revisited. *Artif. Intell.* **2010**, *174*, 449–478.

(45) Harvey, W. D.; Ginsberg, M. L. Limited discrepancy search. Proc. of IJCAI'95. 1995; pp 607–615.

(46) Fontaine, M.; Loudni, S.; Boizumault, P. Exploiting Tree Decomposition for Guiding Neighborhoods Exploration for VNS. *RAIRO OR* **2013**, *47*, 91–123.

(47) Larrosa, J.; Rollon, E.; Dechter, R. Limited Discrepancy AND/OR Search and Its Application to Optimization Tasks in Graphical Models. Proc. of IJCAI. 2016; pp 617–623.

(48) Luby, M.; Sinclair, A.; Zuckerman, D. Optimal speedup of Las Vegas algorithms. Proc. of TCS. 1993; pp 128–133.

(49) Lopes, A.; Aleksandrov, A.; Bathelt, C.; Archontis, G.; Simonson, T. Computational sidechain placement and protein mutagenesis with implicit solvent models. *Proteins: Struct., Funct., Bioinf.* **2007**, *67*, 853–867.

(50) Tuffery, P.; Etchebest, C.; Hazout, S.; Lavery, R. A New Approach to the Rapid Determination of Protein Side Chain Conformations. *J. Biomol. Struct. Dyn.* **1991**, *8*, 1267–1289, PMID: 1892586.

(51) Gaillard, T.; Simonson, T. Full Protein Sequence Redesign with an MMGBSA Energy Function. *J. Chem. Theory Comput.* **2017**, *13*, 4932–4943, PMID: 28886244.

(52) Hansen, P.; Mladenović, N.; Moreno Pérez, J. A. Variable neighbourhood search: methods and applications. *4OR* **2008**, *6*, 319–360.

(53) Jarboui, B.; Sifaleras, A.; Rebai, A. 3rd International Conference on Variable Neighborhood Search (VNS'14). *Electronic Notes in Discrete Mathematics* **2015**, *47*, 1 – 4, The 3rd International Conference on Variable Neighborhood Search (VNS'14).

(54) Subramaniam, S.; Senes, A. Backbone dependency further improves side chain prediction efficiency in the Energy-based Conformer Library (bEBL). *Proteins: Struct., Funct., Bioinf. 82*, 3177–3187.

(55) Davey, J. A.; Chica, R. A. Multistate approaches in computational protein design. *Protein Sci* **2012**, *21*, 1241–1252.

(56) St-Jacques, A. D.; Gagnon, O.; Chica, R. A. *Modern Biocatalysis: Advances Towards Synthetic Biological Systems*; Royal Society of Chemistry, 2018; Vol. 32; p 88.

(57) Ouali, A.; Loudni, S.; Loukil, L.; Boizumault, P.; Lebbah, Y. Replicated parallel strategies for decomposition guided VNS. *Electronic Notes in Discrete Mathematics* **2015**, *47*, 93–100.

(58) Opuu, V.; Silvert, M.; Simonson, T. Computational design of fully overlapping coding schemes for protein pairs and triplets. *Scientific Reports* **2017**, *7*, 15873.

Graphical TOC Entry