Received: 29 December 2020 | Accepted: 25 May 2021

DOI: 10.1002/prot.26174

RESEARCH ARTICLE

PROTEINS WILEY

Iterated local search with partition crossover for computational protein design

François Beuvin[1,2] | Simon de Givry[2,3] | Thomas Schiex[2,3] | Sébastien Verel[4] | David Simoncini[1,2]

[1]IRIT UMR 5505-CNRS, Université de Toulouse I Capitole, Toulouse, France

[2]Artificial and Natural Intelligence Toulouse Institute, ANITI, Toulouse, France

[3]MIAT, Université de Toulouse, INRAE, UR 875, Toulouse, France

[4]Université du Littoral Côte d'Opale, Calais, France

Correspondence
David Simoncini, IRIT UMR 5505-CNRS, Université de Toulouse I Capitole, Toulouse, France.
Email: david.simoncini@irit.fr

Sébastien Verel, Université du Littoral Côte d'Opale, Calais, France.
Email: verel@univ-littoral.fr

Funding information
ANITI (Artificial and Natural Intelligence Toulouse Institute)—Institut 3iA (2019), Grant/Award Number: ANR-19-P3IA-0004

ABSTRACT

Structure-based computational protein design (CPD) refers to the problem of finding a sequence of amino acids which folds into a specific desired protein structure, and possibly fulfills some targeted biochemical properties. Recent studies point out the particularly rugged CPD energy landscape, suggesting that local search optimization methods should be designed and tuned to easily escape local minima attraction basins. In this article, we analyze the performance and search dynamics of an iterated local search (ILS) algorithm enhanced with partition crossover. Our algorithm, PILS, quickly finds local minima and escapes their basins of attraction by solution perturbation. Additionally, the partition crossover operator exploits the structure of the residue interaction graph in order to efficiently mix solutions and find new unexplored basins. Our results on a benchmark of 30 proteins of various topology and size show that PILS consistently finds lower energy solutions compared to Rosetta `fixbb` and a classic ILS, and that the corresponding sequences are mostly closer to the native.

KEYWORDS
combinatorial optimization, computational protein design, computational structure biology, energy landscapes, local search methods

1 | INTRODUCTION

Proteins are responsible for a wide range of vital functions in all living organisms, such as cell signaling, transport, regulation, defense against pathogens and catalysis of various chemical reactions. By exploiting the relationship between the sequence of amino acids of a protein, its three-dimensional structure, and its function, it is possible to engineer new proteins for various applications in health, environment, and bio-nanotechnologies.[1-5] The need for efficient computational protein design (CPD) methods emerged from the fact that it is impossible to experimentally test all possible protein sequences corresponding to a target protein structure. CPD, therefore, aims at finding a sequence of amino acids that fold into a target three-dimensional protein structure using purely in silico methods. It can be formalized as a combinatorial optimization problem where variables are amino acid conformations at each sequence position and where an energy function capturing interactions between amino acids within a target three-dimensional structure is to be minimized.[6] In the most common representation, the energy function is pairwise decomposable, the variables take their values in a discrete set of preferred amino acid side chain nature and orientations, and the backbone of the target structure is fixed. Under these assumptions, the CPD problem has been proven to be NP-hard.[7] For this reason, most CPD methods rely on stochastic optimization. For example, the widely used molecular modeling suite Rosetta relies on a simulated annealing algorithm.[8] Other existing methods rely on evolutionary algorithms such as genetic algorithms[9] or estimation of distribution algorithms.[10] Along with local search methods, exact and deterministic methods that can provably identify the global minimum of the energy function (global minimum energy conformation, GMEC) also exist.[11] The state-of-the-art here relies on Cost Function Networks algorithms,[12,13] and has recently been extended to optimize sequences for one or several protein states at the same time.[14] It is available as open source software under the name of "*POMP*[d]." In addition to providing access to the protein sequence of

lowest energy, *POMP*^{d}, which relies on the constraint programming solver ToulBar2,[15] is able to exhaustively enumerate all protein sequences within a threshold to the global minimum. Using this ability, a recent fitness landscape analysis around the optimum of CPD problems showed that the structure of CPD problems can prevent simulated annealing from approaching the GMEC.[16] Indeed, the number and depth of local minima on CPD problems requires simulated annealing methods to accept several unfavorable local moves in a row in order to escape local minima. As a result, the search often gets stuck with infinitesimal chances of finding the GMEC. Besides this, it has also been shown that the gap between the best solutions found with simulated annealing and the GMEC increases with the length of the target proteins. It is thus crucial to develop new local search methods that could shrink this gap once *POMP*^{d} hits the complexity barrier of NP-hardness and cannot provide the GMEC anymore. A careful analysis of CPD fitness landscapes put in evidence that local search methods with high exploration abilities would be more suitable. Here, we study the performance of iterated local search algorithms (ILS).[17] ILS is an iterated process of steepest descent and random solution perturbations. The steepest descent ensures to reach a local minimum, and the perturbation is used to jump off its attraction basin with the hope of reaching a lower local minimum after the next steepest descent. ILS algorithms can be augmented with a so-called partition crossover, which can mix two solutions together by taking advantage of the problem structure, in order to reach better local minima. This kind of algorithms, mixing iterated local search and partition crossover, has already demonstrated good performance for pseudo-Boolean optimization.[18]

In this article, we show the benefits of using explorative local search methods on CPD problems. We generalize partition crossover to combinatorial optimization, combine it with an ILS in a new algorithm that we named PILS, and compare the performances with Rosetta's simulated annealing algorithm as implemented in the `fixbb` protocol and a classical ILS on a benchmark of 30 protein targets.

2 | MATERIALS AND METHODS

2.1 | Energy function

2.1.1 | Definition

Under the assumption that the protein backbone is fixed, the CPD problem can be modeled by only taking into account the effects of the side chain orientation and nature at each residue position. Using this representation, the energy of the system can be decomposed as a sum of unary and binary terms capturing respectively interactions between one residue and the environment and interactions between pairs of residues. The total energy depends on the side chain nature and orientation of each residue in the protein. The continuous space of side chain orientations is discretized by using libraries of statistically preferred orientations called rotamers[19] for every possible amino acid. One solution to the CPD problem is thus represented as a rotamer assignment for all residues in the protein. The energy function is expressed in Equation (1), where $x \in X$ represents a solution of length ℓ from the sequence/conformation space X (which contains all possible rotamer assignments for a given protein structure), and where E_i and E_{ij} are, respectively, unary and binary energy terms whose values are function of rotamer assignments at position x_i (for E_i) and at positions x_i, x_j (for E_{ij}). G is the undirected interaction graph, whose edges represent all interactions between pairs of residues.

$$E(x) = \sum_{i=1}^{\ell} E_i(x_i) + \sum_{(i,j) \in G} E_{ij}(x_i, x_j) \quad (1)$$

In our experiments, we minimize the energy function *beta_nov16*, as provided by the Rosetta modeling software.[20]

2.1.2 | Neighborhood

Local search algorithms rely on the notion of neighborhood. In CPD, we define a neighbor of a solution as an assignment that differs at one position. The neighborhood relation $\mathcal{N}$ is defined as:

$$\mathcal{N}(x) = \{x' \in X : d_{hamming}(x, x') = 1\}$$

where $d_{hamming}$ is the Hamming distance defined over residues: the distance is 1 if only one rotamer differs between the two solutions. Let $op_{i,v}(x) = x'$ be such that $\forall j \neq i, x'_j = x_j$, and $x'_i = v$. We have:

$$\mathcal{N}(x) = \{op_{i,v}(x) : i \in \{1,...,\ell\}, v \in \{1,...,n_i\} \setminus \{x_i\}\}$$

The size of the neighborhood is then $|\mathcal{N}| = \sum_{i=1}^{\ell}(n_i - 1) = \sum_{i=1}^{\ell} n_i - \ell$, where i is a variable index, n_i is the domain size of variable i (the number of available rotamers) and ℓ the number of variables.

2.1.3 | Local update

The time complexity to evaluate $E(x)$ can be reduced using incremental evaluation. Let $l_i = j \mid (i,j) \in G$ be the set of variables interacting with i.

Let $\delta_{(i,v)}(x)$ be the difference of energy between the neighbors x and $op_{i,v}(x)$:

$$\delta_{(i,v)}(x) = E(op_{i,v}(x)) - E(x)$$

Only a few terms from Equation (1) are modified in order to compute $\delta_{(i,v)}(x)$:

$$\delta_{(i,v)}(x) = E_i(v) - E_i(x_i) + \sum_{j \in l_i} (E_{i,j}(v, x_j) - E_{i,j}(x_i, x_j))$$

The time complexity of the incremental evaluation is then $|l_i| + 1$ which is bounded by ℓ. This time complexity is linear instead of quadratic using Equation (1).

The time complexity can be further reduced using double incremental evaluation. Similar to second derivative computation, let be $\delta^2_{(i,v),(k,w)}(x)$ the variation of $\delta_{(i,v)}(x)$ when x move to $op_{k,w}(x)$.

$$\delta^2_{(i,v),(k,w)}(x) = \delta_{i,v}\left(op_{k,w}(x)\right) - \delta_{i,v}(x)$$

By definition, with $x' = op_{k,w}(x)$,

$$\begin{aligned}\delta^2_{(i,v),(k,w)}(x) = {} & E_i(v) - E_i\left(x'_i\right) - \left(E_i(v) - E_i(x_i)\right) \\ & + \sum_{j \in I_i}\left[E_{i,j}\left(v,x'_j\right) - E_{i,j}\left(x'_i,x'_j\right) - \left(E_{i,j}(v,x_j) - E_{i,j}(x_i,x_j)\right)\right]\end{aligned}$$

$\delta^2_{(i,v),(k,w)}(x)$ can be rewritten as:

$$\begin{aligned}\delta^2_{(i,v),(k,w)}(x) = {} & E_i(x_i) - E_i\left(x'_i\right) \\ & + \sum_{j \in I_i}\left[E_{i,j}(x_i,x_j) - E_{i,j}\left(x'_i,x'_j\right) + E_{i,j}\left(v,x'_j\right) - E_{i,j}(v,x_j)\right]\end{aligned}$$

The computation complexity of $\delta^2_{(i,v),(k,w)}(x)$ may look similar to that of $\delta_{i,v}(x)$. However, according to the values of i and k, some simplifications reduce the complexity. When $i = k$, then $x'_i = w$, and $\forall j \in I_i, x'_j = x_j$,

$$\delta^2_{(i,v),(i,w)}(x) = E_i(x_i) - E_i(w) + \sum_{j \in I_i}\left[E_{i,j}(x_i,x_j) - E_{i,j}(w,x_j)\right]$$

This case is the worst case. The time complexity is the same as for $\delta_{i,v}(x)$ computation. When $i \neq k$, and $k \notin I_i$, then, $x'_i = x_i$, and $\forall j \in I_i, x'_j = x_j$,

$$\delta^2_{(i,v),(k,w)}(x) = 0$$

This case is the best case, and if $|I_i|$ is bounded, it is the most common case. The complexity is 0. When $i \neq k$, and $k \in I_i$, then, $x'_i = x_i$, $\forall j \in I_i \setminus \{k\}, x'_j = x_j$,

$$\delta^2_{(i,v),(k,w)}(x) = E_{i,k}(x_i,x_k) - E_{i,k}\left(x_i,x'_k\right) + E_{i,k}\left(v,x'_k\right) - E_{i,k}(v,x_k)$$

In this case, the time complexity to compute $\delta^2_{(i,v),(k,w)}(x)$ is only three operations.

Overall, to update the $\delta_{i,v}(x)$ values for all neighbors (i,v), the complexity is $(|I_i|+1)+3|I_k|$. $|I_i|$ and $|I_k|$ are bounded by $\ell - 1$, the complexity of the update of all δ values is bounded by 4ℓ, which is a linear complexity.

2.2 | Algorithms

2.2.1 | ILS

Iterated local search consists in building a sequence of locally optimal solutions by iteratively perturbing the current local minimum and applying a local search operator.[17] The ILS used in this work is given in Algorithm S1. The main components of the algorithm are the perturbation operator and the local search operator. A perturbation of strength k consists in randomly modifying the value of k variables in the solution. The local search operator used is a steepest descent. This cycle of perturbations and local searches iterates while keeping track of the best solution until a limit number of solution evaluations is reached, or if no improvement occurs in a predetermined number of iterations.

The steepest descent algorithm used as local search operator is given in Algorithm S2. Double incremental evaluation is used to find the best neighbor at each iteration of the algorithm. A Binary Search Tree is used to select one of the best neighbor with complexity $\mathcal{O}(\log(|\mathcal{N}|))$.

2.2.2 | PILS

The algorithm of PILS (see Algorithm 1) is built on the generic form of the ILS. It introduces a second perturbation operator: the partition crossover. The partition crossover exploits the structure of the variable interaction graph in order to efficiently mix two solutions.[21] A description of this crossover operator is presented in Algorithm S3. The partition crossover operator takes two parent solutions as input. First, it removes from the interaction graph all edges linking variables with equal values in both solutions. These values are preserved in the child solution. Then, it evaluates each parent solution on the set of connected components in the variable interaction graph. For each variable in each connected component, the values from the best parent are used in the child solution. The main loop in PILS algorithm generates two local minima solutions (*sol*1 and *sol*2) by perturbation and steepest descent. The two local minima are then combined using partition crossover to produce a new solution. A local minimum is reached from this new solution and stored in *sol*1. The loop iterates until a limit number of evaluations is reached, or if no improvement occurs in a predetermined number of iterations.

Algorithm 1. PILS algorithm

1: **Inputs:** e_{max}: maximum number of "evaluations"
steady_max: maximum number of evaluations without improvement
2: *sol*1 ← random_init(*sol*1)
3: *sol*1 ← steepest-descent(*sol*1), update(e_{tot})
4: *abort* ← *False*
5: repeat
6: *sol*2 ← perturbation(*sol*1, *k*)
7: *sol*2 ← steepest-descent(*sol*2), update(e_{tot})
8: *sol*1 ← partition-crossover(*sol*1, *sol*2)
9: *sol*1 ← steepest-descent(*sol*1), update(e_{tot})
10: If no improvement in *steady_max* iterations: *abort* ← *True*
11: **until** $e_{tot} \geq e_{max}$ or *abort*
12: **Output:** *sol*1: best solution found

3 | RESULTS AND DISCUSSION

We evaluated the performance of PILS compared to ILS and Rosetta `fixbb` protocol on a benchmark of 30 proteins of size ranging from 53 to 159 residues, representing all folding classes ($\alpha, \beta, \alpha/\beta, \alpha+\beta$). All calculations were run on the high-performance computing center of the University of Toulouse (CALMIP). Each method was run 100 times on each protein target. All calculations were performed on Skylake 6140 CPUs on the HPC cluster CALMIP. The time needed for Rosetta `fixbb` protocol to complete the benchmark was used as reference, and the stopping criterion of PILS and ILS were calibrated in order to obtain comparable total CPU times. Rosetta `fixbb` protocol needed 9 days and 4 h to complete the benchmark, ILS needed 9 days and 16 h, and PILS needed 8 days and 23 h. Both ILS and PILS were allowed 60 000 energy function evaluations per run. ILS and PILS runs were stopped if no improvement was observed during respectively 1500 and 500 iterations. The perturbation strength was set to 2 for ILS and 1/3 of the protein sequence length for PILS. The perturbation strengths are set differently for the two methods because of their different nature: ILS uses perturbations in order to escape local minima whereas PILS needs diverse solutions for efficient breeding with the crossover operator. In this section, we present a statistical analysis of

TABLE 1 Median and best performance of PILS, ILS, and Rosetta

	Median			Best		
Target	PILS	ILS	Rosetta	PILS	ILS	Rosetta
erw	−124.01	−123.42	−122.64	−124.01*	−124.01	−123.82
cmp	−142.24	−140.24	−140.94	−143.75*	−142.96	−142.71
ku3	−147.03	−145.57	−145.35	−147.75	−147.5	−147.03
f94	−159.67	−158.27	−157.85	−159.86	−159.86	−159.45
cjj	−144.74	−141.52	−141.03	−145.4*	−144.93	−145.0
orc	−188.96	−188.03	−186.92	−189.27*	−189.27	−189.14
uoy	−173.16	−171.99	−171.72	−173.16*	−173.16	−173.16
hcs	−179.84	−178.65	−175.67	−180.02*	−180.02	−179.9
pgx	−198.14	−194.27	−196.46	−198.21*	−198.09	−197.88
hoe	−211.47	−208.45	−208.45	−211.75*	−211.46	−210.78
k3v	−196.52	−190.56	−189.27	−196.63	−196.5	−194.57
x3o	−238.62	−235.08	−235.98	−239.17*	−238.71	−238.08
ckx	−184.94	−179.86	−181.51	−186.77	−186.54	−186.09
vjk	−295.58	−290.88	−292.4	−295.65*	−294.9	−295.42
dsl	−280.6	−271.88	−278.29	−281.39*	−280.32	−281.31
x6j	−212.99	−207.89	−209.3	−213.09	−212.99	−212.27
fna	−268.79	−263.54	−264.73	−269.76*	−267.65	−268.11
yxm	−307.3	−302.5	−304.92	−307.42*	−307.13	−307.16
cqy	−302.06	−294.03	−297.8	−302.3*	−302.04	−301.32
pcy	−299.02	−292.97	−294.9	−299.27*	−298.91	−298.27
fqt	−355.78	−351.54	−351.9	−356.01*	−355.84	−355.44
a0b	−322.85	−316.32	−316.35	−324.84	−324.07	−322.6
pnd	−377.85	−368.87	−372.04	−378.96	−378.18	−377.83
mvo	−361.06	−350.11	−351.73	−365.56	−364.06	−364.25
qlc	−373.0	−359.51	−361.81	−377.54	−372.07	−370.6
aqt	−423.93	−411.3	−416.34	−426.64	−422.5	−423.93
z3v	−409.61	−402.46	−403.89	−410.55	−408.85	−408.68
f04	−414.42	−403.38	−409.53	−417.8	−415.01	−415.86
tzv	−420.67	−408.78	−414.38	−425.75	−422.26	−421.84
z2u	−411.05	−401.17	−402.31	−413.0	−411.51	−410.34

Note: Solutions annotated with * are global optimum proven by POMPd. Only best solutions of PILS are annotated since they are consistently superior to solutions from other methods. For all targets, PILS systematically outperforms both other methods on median energy values according to Mann-Whitney *U* test.

each problem instance, results in terms of energy minimization and native sequence recovery rates, and provide some statistics on the search dynamics of PILS.

3.1 | Benchmark description and statistics

The PDB code and length of each protein in the benchmark, as well as main statistics extracted from the interaction graph associated with each protein are shown in Table S1. These statistics reflect the topology of the proteins. The neighborhood size, which is the cumulative number of rotamers at each position in the protein sequence, naturally grows with the length of the proteins, as well as the number of links that sums up all edges in the interaction graph. The average number of rotamers per position ($|\mathcal{N}|/\ell$) remains stable, ranging from approximately 250 to 300. The density of the interaction graph, computed as the number of edges in the graph divided by the number of possible edges, tend to decrease with the length of the proteins. The average number of edges per residue grows with the length, translating the fact that buried residues are more connected, and that the number of buried residues increases with the length for globular proteins.

3.2 | Energy minimization

PILS performs consistently better than ILS and Rosetta `fixbb` in terms of energy minimization. Table 1 shows the median energy and the best energy out of 100 runs for each method. The median energy achieved by PILS is significantly better on all instances according to Mann-Whitney *U* test. Furthermore, the energy gap between median values of PILS and respectively ILS and Rosetta `fixbb` seems to increase linearly with the size of the problems (see Figure 1). The linear regression line has a steeper slope for the energy gap between PILS and ILS, but the same tendency is observed in both cases. This result put in evidence the benefits of using PILS to solve difficult CPD problems. When looking at the best energy values obtained for each method on each target, PILS either outperforms both other methods or achieved equal performance. PILS and ILS find the same best value on five protein targets, PILS and Rosetta `fixbb` on one protein target. In order to check where PILS stands in the energy landscape, we attempted to compute the global optimum of the energy function for each target protein with POMP[d.14] POMP[d] could identify the GMEC on 18 instances out of 30 within a time limit of 100 h. PILS reaches the GMEC on 16 instances out of 18 (indicated with a star in Table 1), whereas ILS and Rosetta `fixbb` were respectively able to locate the GMEC on four and one instances. As pointed out in a previous study, the energy gap between sequences predicted with Rosetta `fixbb` and the GMEC increases with the size of the problem.[12] PILS closes this gap on almost all instances for which the GMEC could be found, showing that such enhanced iterative local search methods are preferable to simulated annealing for solving CPD instances when the size of the problems prevent global optimization methods from returning the global optimum. They could also speed up these global optimization methods by providing better initial upper bounds.

3.3 | Native sequence recovery

Solutions returned by PILS are closer to the native sequences on average in comparison with ILS and Rosetta `fixbb`. Native sequence recovery is a well-known and accepted measure for CPD in silico assessment. This test relies on the relationship between protein sequence and structure: the more two sequences are similar, the more they tend to fold into the same three-dimensional structure. Thus, if a computationally designed sequence is close to the native sequence of the target structure, it has good chances of adopting the correct fold. Table 2 shows average native sequence recovery rates over 100 runs for PILS, ILS and Rosetta `fixbb` on all protein targets. Significantly, better results according to Mann-Whitney *U* test are highlighted in bold. PILS outperforms other methods on 18 protein targets out of 30, whereas ILS is better on one target and Rosetta `fixbb` is better on five targets. Mann-Whitney *U* test was not conclusive for six protein targets. Putting these statistics in perspective with PILS results on energy minimization (Table 1), sequences of lower energy are closer on average to native sequences. This suggests that the all atom energy function developed by Rosetta has become accurate enough so that improved energy minimization is more beneficial than sequence space sampling with simulated annealing.

3.4 | PILS search dynamics

Figure 2 (top) shows the average best solution per iteration on 100 runs for protein targets 2erw and 1z2u. These two proteins are respectively the smallest and the largest in our benchmark. The slope of the curves is steep in the first iterations, showing that PILS is able

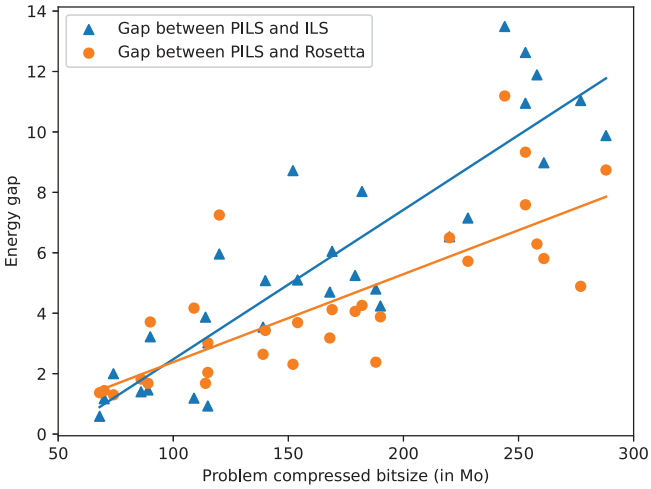

FIGURE 1 Gap between median energies achieved by PILS, ILS, and Rosetta `fixbb` against compressed problem size. Gap between PILS and ILS is shown in blue, gap between PILS and Rosetta `fixbb` is shown in orange. The size of a problem is defined as the amount of bits needed to store the CPD residue interaction graph of a protein target in compressed JSON format. The energy gap is computed as the difference in energy between the two median values. Linear regression lines are plotted with same colors as data points

Target	PILS	ILS	Rosetta	Target	PILS	ILS	Rosetta
erw	**53.86**	52.61	51.8	1x6j	**40.11**	36.48	37.12
cmp	38.19	38.61	37.14	1fna	**41.98**	37.77	37.52
ku3	**43.58**	42.52	41.66	2yxm	**47.61**	46.24	46.42
f94	**37.27**	34.94	34.84	1cqy	**54.09**	47.94	49.37
cjj	**39.97**	36.74	37.41	2pcy	51.6	50.34	**52.03**
orc	41.01	42.53	**43.98**	1fqt	**56.79**	53	52.89
uoy	44	**45.15**	44.45	2a0b	**38.69**	35.99	37.81
hcs	**53.62**	49.94	49.62	2pnd	39.68	37.99	39.3
pgx	**47.41**	40.06	43.93	1mvo	**44.99**	42.76	42.3
hoe	47.65	47.58	**49.66**	2qlc	**46.72**	42.97	45.1
k3v	29.32	30.75	30.72	1aqt	44.91	41.05	44.49
x3o	**50.97**	49.13	49.98	2z3v	50.12	48.93	**50.61**
ckx	40.66	39.59	40.92	3f04	**43.74**	39.05	40.82
vjk	**62.69**	58.14	60.94	1tzv	41.72	39.96	41.49
dsl	43.07	41.9	**45.6**	1z2u	**43.01**	40.56	40.8

TABLE 2 Mean native sequence recovery percentages for PILS, ILS, and Rosetta

Note: Statistically significant results according to Mann-Whitney *U* test are in bold.

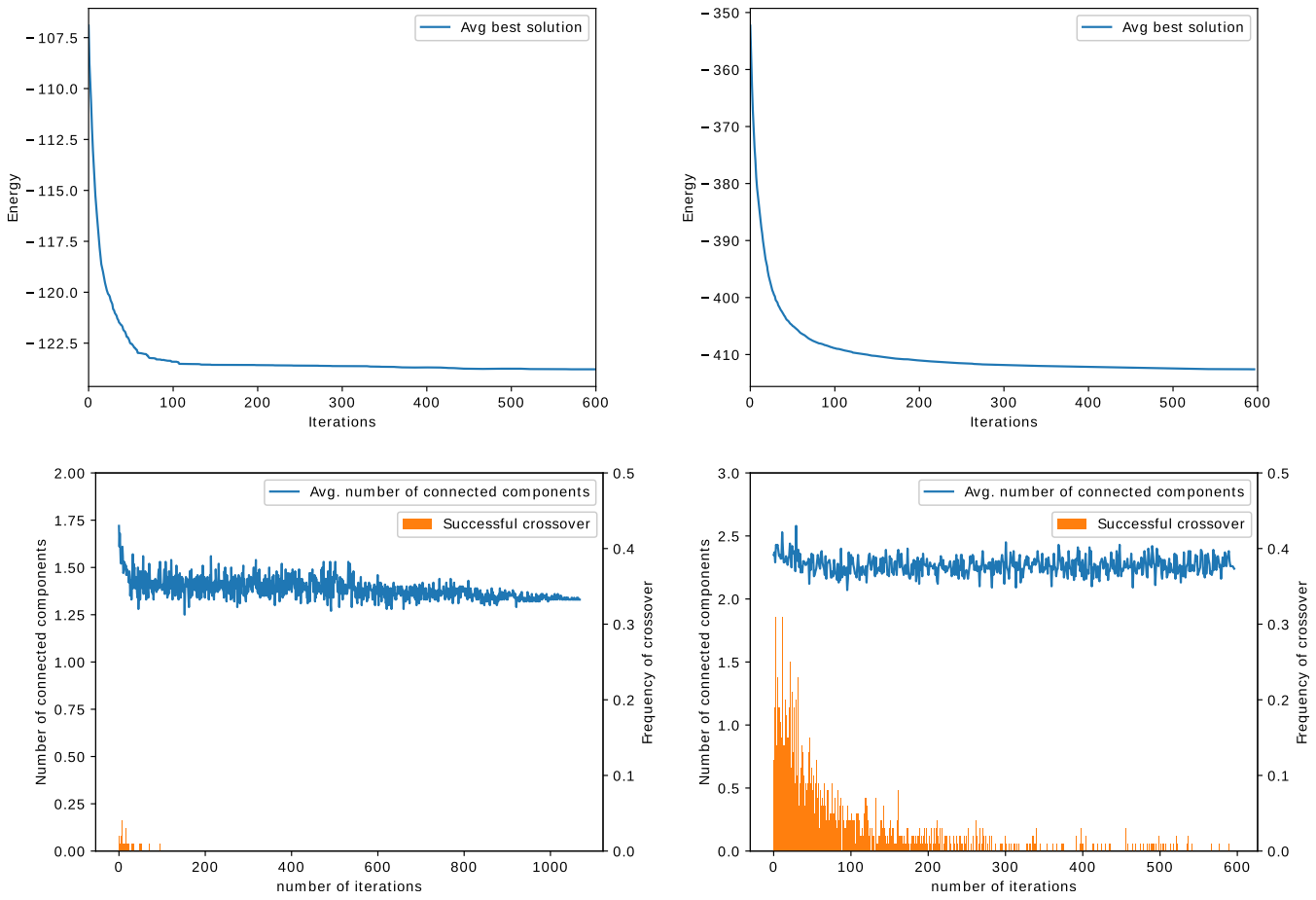

FIGURE 2 Average energy of the best solution per iteration on targets 2erw (top left) and 1z2u (top right). Average number of connected components and successful partition crossovers per iteration on targets 2erw (bottom left) and 1z2u (bottom right) [Color figure can be viewed at wileyonlinelibrary.com]

to quickly identify good solutions. The curve flattens a bit faster in the case of 2erw, which is an easier target. In both cases, we observe that PILS converges to a minimum in less than 600 iterations.

We then looked at the average number of connected components, and the average number of successful crossovers on the same two protein targets (Figure 2, bottom). We consider a crossover as successful if it allows to reach a solution with a lower energy than that of its two parent solutions. We observe different behaviors depending on the protein target. The average number of connected components seems to be constant across iterations in both cases, but is smaller in the case of 2erw. As a consequence, there are few successful crossovers in the first iterations and no successful crossover happens after 50 iterations. In the case of 1z2u, the average number of connected components is higher and allows more successful crossovers. Although the frequency decreases with the iterations, the crossover still has some impact near the end of the runs. 2erw is an easy target on which PILS and ILS could find the global optimum. On this target, the iterated local search on its own is sufficient to quickly find good solutions. The smaller number of connected components translates the fact that the search quickly focuses on the correct region in the search space. It leaves no room for solution diversity that is essential for connected components to appear. On the other side, 1z2u is the longest protein target, with the biggest neighborhood size (Table S1), and could be considered as the most difficult target in the benchmark. In this case, the partition crossover in PILS helps finding better solutions. PILS can be seen as a hybrid evolutionary algorithm with a population of two solutions, partition crossover as a stochastic operator and ILS as a local search operator. From that perspective, increasing the size of the population and defining an appropriate selection operator (selecting solutions for crossover so that the number of connected components is maximized) would probably increase the number of successful crossovers, and as a consequence improve the performance of the algorithm. The local search operator can run independently on each solution and could easily be parallelized to benefit from the computing power of high-performance computing centers.

4 | CONCLUSION

CPD has become a major tool in protein engineering and fundamental structural biology. The massive amount of sequence data and the continuously growing number of structures in the Protein Data Bank combined with nowadays-computational resources and experimental validation techniques have greatly contributed to our understanding of the determinants of protein design modeling. Efficient optimization algorithms become crucial in order to fully benefit from the increasing accuracy of energy functions and progress in protein design modeling. Recent findings on the properties of the energy landscapes describing CPD problems shed light on some misfit of classically used local search optimization techniques such as simulated annealing. The energy landscapes appear particularly rugged, and methods able to better escape local minima are needed.

The algorithm presented in this article, PILS, combines an iterated local search algorithm with a partition crossover operator. This algorithm relies on a fast double incremental steepest descent algorithm and on a crossover, which exploits the structure of the residue interaction graph to find new solutions. Our results on a benchmark of 30 proteins demonstrate the efficiency of PILS in terms of energy minimization and native sequence recovery. The energy gap between PILS and the other tested methods increases linearly with the size of the problems. Additionally, PILS was able to locate the GMEC on 16 out of 18 targets on which the global optimum could be proven by a global optimization method. Solutions of better energy found by PILS often correspond to sequences having a higher sequence identity with the native sequences. The role of the partition crossover has been identified as preponderant on difficult targets, and could be used in a variety of parallel population-based optimization methods for CPD.

ACKNOWLEDGMENTS

This work was funded by ANITI (Artificial and Natural Intelligence Toulouse Institute)—Institut 3iA (2019) (ANR-19-P3IA-0004). We thank CALMIP for computational resources.

CONFLICT OF INTERESTS

You may be asked to provide a conflict of interest statement during the submission process. Please check the journal's author guidelines for details on what to include in this section. Please ensure you liaise with all co-authors to confirm agreement with the final statement.

PEER REVIEW

The peer review history for this article is available at https://publons.com/publon/10.1002/prot.26174.

DATA AVAILABILITY STATEMENT

The source code of PILS is freely available (https://forgemia.inra.fr/david.simoncini/pils). Datasets will be provided upon request.

ORCID

David Simoncini https://orcid.org/0000-0002-3772-5473

REFERENCES

1. Strauch E, Bernard S, La D, et al. Computational design of trimeric influenza-neutralizing proteins targeting the hemagglutinin receptor binding site. *Nat Biotechnol* 2017 06;35:667-671.
2. Noguchi H, Addy C, Simoncini D, et al. Computational design of symmetrical eight-bladed β-propeller proteins. *IUCrJ*. 2019;6(1):46-55.
3. Palm G, Reisky L, Böttcher D, et al. Structure of the plastic-degrading Ideonella sakaiensis MHETase bound to a substrate. *Nat Commun*. 2019;04:10.
4. Ng AH, Nguyen TH, Gómez-Schiavon M, et al. Modular and tunable biological feedback control using a de novo protein switch. *Nature*. 2019;572(7768):265-269.
5. Tournier V, Topham C, Gilles A, et al. An engineered PET depolymerase to break down and recycle plastic bottles. *Nature* 2020 04;580:216-219.

6. Allouche D, André I, Barbe S, et al. Computational protein design as an optimization problem. *Artif Intell*. 2014;212:59-79.
7. Pierce NA, Winfree E. Protein design is NP-hard. *Protein Eng*. 2002;15 (10):779-782.
8. Leaver-Fay A, Tyka M, Lewis SM, et al. ROSETTA3: an object-oriented software suite for the simulation and design of macromolecules. *Methods Enzymol*. 2011;487:545-574.
9. Chowdry AB, Reynolds KA, Hanes MS, Voorhies M, Pokala N, Handel TM. An object-oriented library for computational protein design. *J Comput Chem*. 2007;28(14):2378-2388.
10. Simoncini D, Zhang KYJ, Schiex T, Barbe S. A structural homology approach for computational protein design with flexible backbone. *Bioinformatics* 2018 11;35(14):2418–2426.
11. Hallen MA, Donald BR. Protein design by provable algorithms. *Commun ACM*. 2019;62(10):76-84.
12. 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(12):5980-5989.
13. Traoré S, Roberts KE, Allouche D, et al. Fast search algorithms for computational protein design. *J Comput Chem*. 2016;37(12): 1048-1058. https://doi.org/10.1002/jcc.24290
14. Vucinic J, Simoncini D, Ruffini M, Barbe S, Schiex T. Positive multistate protein design. *Bioinformatics*. 2019;36(1):122-130.
15. Hurley B, O'Sullivan B, Allouche D, et al. Multi-language evaluation of exact solvers in graphical model discrete optimization. *Constraints*. 2016;21(3):413-434.
16. Simoncini D, Barbe S, Schiex T, Verel S. Fitness landscape analysis around the optimum in computational protein design. *Proceedings of the Genetic and Evolutionary Computation Conference GECCO '18*. New York, NY: Association for Computing Machinery; 2018: 355-362.
17. Lourenço H, Martin O, Stützle T. Iterated local search: framework and applications. 2010;146:363-397.
18. Chicano F, Whitley D, Ochoa G, Tinós R. Optimizing one million variable NK landscapes by hybridizing deterministic recombination and local search. *Proceedings of the Genetic and Evolutionary Computation Conference GECCO '17*. New York, NY: Association for Computing Machinery; 2017:753-760. https://doi.org/10.1145/3071178.3071285
19. Shapovalov MV, Dunbrack RL. A smoothed backbone-dependent rotamer library for proteins derived from adaptive kernel density estimates and regressions. *Structure*. 2011;19(6):844-858.
20. Alford R, Leaver-Fay A, Jeliazkov J, et al. The Rosetta All-Atom Energy Function for macromolecular modeling and design. *J Chem Theory Comput*. 2017;4:13.
21. Tinós R, Whitley D, Chicano F. Partition crossover for pseudo-boolean optimization. *Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII FOGA '15*. New York, NY: Association for Computing Machinery; 2015:137-149. https://doi.org/10.1145/2725494.2725497

SUPPORTING INFORMATION

Additional supporting information may be found online in the Supporting Information section at the end of this article.

How to cite this article: Beuvin F, de Givry S, Schiex T, Verel S, Simoncini D. Iterated local search with partition crossover for computational protein design. *Proteins*. 2021; 1-8. https://doi.org/10.1002/prot.26174