Computational Protein Design as a Cost Function Network Optimization Problem

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

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

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

1 Introduction

A protein is a sequence of basic building blocks called amino acids. Proteins are involved in nearly all structural, catalytic, sensory, and regulatory functions of living systems [11]. Performance of these functions generally requires the assembly of proteins into well-defined three-dimensional structures specified by their amino acid sequence. Over millions of years, natural evolutionary processes have shaped and created proteins with novel structures and functions by means of sequence variations, including mutations, recombinations and duplications. Protein engineering techniques coupled with high-throughput automated procedures offer today the possibility to mimic the evolutionary process on a greatly accelerated time-scale, and thus increase the odds to identify the proteins of interest for technological uses [30]. This holds great interest for medicine, biotechnology, synthetic biology and nanotechnologies [28, 33, 15].

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

With a choice among 20 naturally occuring amino acids at every position, the size of the combinatorial sequence space is however clearly out of reach of current experimental methods, even for small proteins. Computational protein design (CPD) methods therefore try to intelligently guide this process by producing a collection of proteins, intended to be rich in functional proteins and whose size is small enough to be experimentally evaluated. The challenge of choosing a sequence of amino acids to perform a given task is formulated as an optimization problem, solvable computationally. It is often described as the inverse problem of protein folding [29]: the three-dimensional structure is known and we have to find amino acid sequences that folds into it. It can also be considered as a highly combinatorial variant of side-chain positioning [36] because of possible amino acid changes.

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

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

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

2 The Computational Protein Design approach

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

(Global Minimum Energy Conformation). The energy of a conformation can be directly computed from the amino acid sequence and rotamers by introducing substitutions within the native structure.

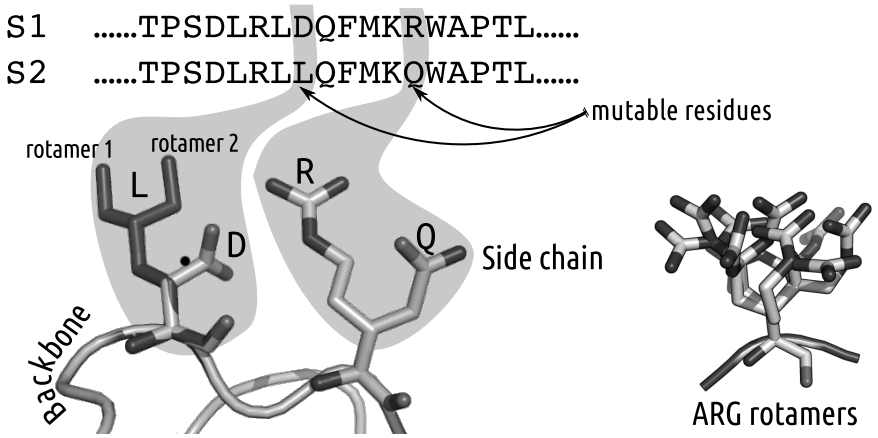

Fig. 1. A local view of combinatorial sequence exploration considering a common backbone. Changes can be caused by amino acid identity substitutions (*for example D/L* or *R/Q*) or by amino acid side-chain reorientations (rotamers) for a given amino acid. A typical rotamer library for one amino acid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

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

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

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

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

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

3 Existing approaches for the CPD

The protein design problem as defined above, with a rigid backbone, a discrete set of rotamers, and pairwise energy functions has been proved to be NP-hard [32]. Hence, a variety of meta-heuristics have been applied to it, including Monte Carlo simulated annealing [21], genetic algorithms [34], and other algorithms [10]. The main weakness of these approaches is that they may remain stuck in local minima and miss the GMEC without notice.

However, there are several reasons motivating the exact solving of the problem. First, because they know when an optimum is reached, exact methods may stop before metaheuristics. Voigt et al. [37] reported that the accuracy of metaheuristics also degrades as problem size increases. More importantly, the use of exact search algorithms becomes crucial in the usual experimental design cycle that goes through CPD modeling, solving, protein synthesis and experimental evaluation: when unexpected experimental results are obtained, the only possible culprit lies in the CPD model and not in the algorithm.

Current exact methods for CPD mainly rely on the dead-end-elimination (DEE) theorem [9, 8] and the A^* algorithm [25, 13]. From a constraint satisfaction perspective, the DEE theorem can be seen as an extension of neighborhood substitutability [7, 20, 2]. DEE is used as a pre-processing technique and removes rotamers that are locally dominated by other rotamers, until a fixpoint is reached. The rotamer r at position i is removed if there exists another rotamer u at the same position such that [9]:

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

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

Extensions to higher orders have been considered [14, 31, 26, 12]. These DEE criteria preserve the optimum but may remove suboptimal solutions.

This DEE preprocessing is usually followed by an A^* search method. After DEE pruning, the A^* algorithm allows to expand a sequence-conformation tree, so that sequence-conformations are extracted and sorted on the basis of their energy values. At depth d of the tree, the lower bound used by $A*$ [13] is exactly the PFC-DAC lower bound [38, 23] used in WCSP and later obsoleted by soft arc consistencies [35, 22, 5]:

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

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

4 Cost Function Network model

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

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

5 Integer Linear programming model

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

is equal to 1 iff the pair (i_r, j_s) is used. Constraints enforce that a pair is used iff the corresponding values are used. Then, finding a GMEC reduces to the following ILP:

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

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

6 Experimental Results

We used a set of 12 protein design cases to evaluate the performance of `toulbar2`, `cplex` and compare them with the DEE/A* approach implemented in `osprey` (open source dedicated Java CPD software). This set comprises 9 protein structures derived from the PDB which were chosen for the high resolution of their 3D-structures and their distribution of sizes and types. Diverse sizes of sequence-conformation combinatorial spaces were considered, varying by the number of mutable residues, the number of alternative amino acid types at each position and the number of conformations for each amino acid (Table 1). The *Penultimate* rotamer library was used [27].

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

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

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

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

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

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

6.1 Explaining the differences

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

If we consider `osprey` first, it uses an obsolete lower bound instead of the more recent incremental and stronger lower bounds offered by soft local consistencies such as EDAC [22]. This, together with the associated informed value ordering provided by these local consistencies, may explain why `toulbar2` outperforms `osprey`. Similarly, the LP relaxation lower bound used in ILP is known (by duality) to be the same as the Optimal Soft AC lower bound (when no upper bounding occurs, i.e. when $k = +\infty$). Since OSAC dominates all other local consistencies at the arc level, this provides an explanation for the efficiency of `cplex` compared to `osprey`. Finally, the problem is deeply non linear. It can be concisely formulated as a CFN but the ILP formulation is much more verbose. This probably contributes, together with the upper bounding (provided by node consistency) and value ordering heuristics of `toulbar2`, to the efficiency of `toulbar2` compared to `cplex`.

7 Conclusion

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

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

The tendency in CPD is to solve more sophisticated formulations of this already challenging problem by relaxing assumptions such as a fixed backbone, or by considering sub-optimal conformations. Existing CFN algorithms still need to be extended and adapted to tackle such problems.

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

References

1. Anfinsen, C.: Principles that govern the folding of protein chains. Science 181(4096), 223–253 (1973)
2. Bistarelli, S., Faltings, B., Neagu, N.: Interchangeability in soft csps. In: International Workshop on Constraint Solving and Constraint Logic Programming. pp. 31–46 (2002)
3. Boas, F., Harbury, P.: Potential energy functions for protein design. Current opinion in structural biology 17(2), 199–204 (2007)
4. Case, D., Darden, T., Cheatham III, T., Simmerling, C., Wang, J., Duke, R., Luo, R., Merz, K., Pearlman, D., Crowley, M., Walker, R.C., Zhang, W., Wang, B., Hayik, S., Roitberg, A., Seabra, G., Wong, K.F., Paesani, F., Wu, X., Brozell, S., Tsui, V., Gohlke, H., Yang, L., Tan, C., Mongan, J., Hornak, V., Cui, G., Beroza, P., Mathews, D.H., Schafmeister, C., Ross, W.S., Kollman, P.A.: Amber 9. University of California, San Francisco (2006)
5. Cooper, M., de Givry, S., Sanchez, M., Schiex, T., Zytnicki, M., Werner, T.: Soft arc consistency revisited. Artificial Intelligence 174, 449–478 (2010)
6. Cooper, M.C., de Givry, S., Schiex, T.: Optimal soft arc consistency. In: Proc. of IJCAI'2007. pp. 68–73. Hyderabad, India (Jan 2007)
7. Cooper, M.: Fundamental properties of neighbourhood substitution in constraint satisfaction problems. Artificial Intelligence 90(1-2), 1–24 (1997)
8. Dahiyat, B., Mayo, S.: Protein design automation. Protein Science 5(5), 895–903 (1996)
9. Desmet, J., Maeyer, M., Hazes, B., Lasters, I.: The dead-end elimination theorem and its use in protein side-chain positioning. Nature 356(6369), 539–542 (1992)
10. Desmet, J., Spriet, J., Lasters, I.: Fast and accurate side-chain topology and energy refinement (FASTER) as a new method for protein structure optimization. Proteins: Structure, Function, and Bioinformatics 48(1), 31–43 (2002)
11. Fersht, A.: Structure and mechanism in protein science: a guide to enzyme catalysis and protein folding. WH Freemean and Co., New York (1999)
12. Georgiev, I., Lilien, R., Donald, B.: Improved pruning algorithms and divide-and-conquer strategies for dead-end elimination, with application to protein design. Bioinformatics 22(14), e174–e183 (2006)
13. Georgiev, I., Lilien, R., Donald, B.: The minimized dead-end elimination criterion and its application to protein redesign in a hybrid scoring and search algorithm for computing partition functions over molecular ensembles. Journal of Computational Chemistry 29(10), 1527–1542 (2008)
14. Goldstein, R.: Efficient rotamer elimination applied to protein side-chains and related spin glasses. Biophysical Journal 66(5), 1335–1340 (1994)
15. Grunwald, I., Rischka, K., Kast, S., Scheibel, T., Bargel, H.: Mimicking biopolymers on a molecular scale: nano (bio) technology based on engineered proteins. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 367(1894), 1727–1747 (2009)
16. Harvey, W.D., Ginsberg, M.L.: Limited discrepency search. In: Proc. of the 14^{th} IJCAI. Montréal, Canada (1995)
17. Hawkins, G., Cramer, C., Truhlar, D.: Parametrized models of aqueous free energies of solvation based on pairwise descreening of solute atomic charges from a dielectric medium. The Journal of Physical Chemistry 100(51), 19824–19839 (1996)
18. Khare, S., Kipnis, Y., Takeuchi, R., Ashani, Y., Goldsmith, M., Song, Y., Gallaher, J., Silman, I., Leader, H., Sussman, J., et al.: Computational redesign of a

mononuclear zinc metalloenzyme for organophosphate hydrolysis. Nature Chemical Biology 8(3), 294–300 (2012)
19. Kingsford, C., Chazelle, B., Singh, M.: Solving and analyzing side-chain positioning problems using linear and integer programming. Bioinformatics 21(7), 1028–1039 (2005)
20. Koster, A., van Hoesel, S., Kolen, A.: Solving frequency assignment problems via tree-decomposition. Tech. Rep. RM/99/011, Universiteit Maastricht, Maastricht, The Netherlands (1999)
21. Kuhlman, B., Baker, D.: Native protein sequences are close to optimal for their structures. Proceedings of the National Academy of Sciences 97(19), 10383 (2000)
22. Larrosa, J., de Givry, S., Heras, F., Zytnicki, M.: Existential arc consistency: getting closer to full arc consistency in weighted CSPs. In: Proc. of the 19^{th} IJCAI. pp. 84–89. Edinburgh, Scotland (Aug 2005)
23. Larrosa, J., Meseguer, P., Schiex, T., Verfaillie, G.: Reversible DAC and other improvements for solving max-CSP. In: Proc. of AAAI'98. Madison, WI (Jul 1998)
24. Leach, A., Lemon, A., et al.: Exploring the conformational space of protein side chains using dead-end elimination and the A* algorithm. Proteins Structure Function and Genetics 33(2), 227–239 (1998)
25. Looger, L., Hellinga, H.: Generalized dead-end elimination algorithms make large-scale protein side-chain structure prediction tractable: implications for protein design and structural genomics1. Journal of molecular biology 307(1), 429–445 (2001)
26. Lovell, S., Word, J., Richardson, J., Richardson, D.: The penultimate rotamer library. Proteins: Structure, Function, and Bioinformatics 40(3), 389–408 (2000)
27. Nestl, B., Nebel, B., Hauer, B.: Recent progress in industrial biocatalysis. Current Opinion in Chemical Biology 15(2), 187–193 (2011)
28. Pabo, C.: Molecular technology: designing proteins and peptides. Nature 301, 200 (1983)
29. Peisajovich, S., Tawfik, D.: Protein engineers turned evolutionists. Nature methods 4(12), 991–994 (2007)
30. Pierce, N., Spriet, J., Desmet, J., Mayo, S.: Conformational splitting: A more powerful criterion for dead-end elimination. Journal of computational chemistry 21(11), 999–1009 (2000)
31. Pierce, N., Winfree, E.: Protein design is NP-hard. Protein Engineering 15(10), 779–782 (2002)
32. Pleiss, J.: Protein design in metabolic engineering and synthetic biology. Current Opinion in Biotechnology 22(5), 611–617 (2011)
33. Raha, K., Wollacott, A., Italia, M., Desjarlais, J.: Prediction of amino acid sequence from structure. Protein Science 9(6), 1106–1119 (2000)
34. Schiex, T.: Arc consistency for soft constraints. In: Principles and Practice of Constraint Programming - CP 2000. LNCS, vol. 1894, pp. 411–424. Singapore (Sep 2000)
35. Swain, M., Kemp, G.: A CLP approach to the protein side-chain placement problem. In: Principles and Practice of Constraint Programming—CP 2001. pp. 479–493. Springer (2001)
36. Voigt, C., Gordon, D., Mayo, S.: Trading accuracy for speed: a quantitative comparison of search algorithms in protein sequence design. Journal of Molecular Biology 299(3), 789–803 (2000)
37. Wallace, R.: Directed arc consistency preprocessing. In: Meyer, M. (ed.) Selected papers from the ECAI-94 Workshop on Constraint Processing, pp. 121–137. No. 923 in LNCS, Springer, Berlin (1995)