A hybrid approach mixing local search and constraint programming applied to the protein structure prediction problem

Raffele Cipriano[1], Alessandro Dal Palù[2], and Agostino Dovier[1]

[1] Univ. di Udine, Dip. di Matematica e Informatica. (cipriano|dovier)@dimi.uniud.it
[2] Univ. di Parma, Dip. di Matematica, alessandro.dalpalu@unipr.it

Abstract. We present a hybrid system that combines local search techniques and constraint solving. We apply it to the ab-initio protein structure prediction problem, modeled in the Face Centered Cubic Lattice with a pairwise contact energy function. In the literature, the problem is successfully solved using constraint programming for proteins with length up to 160 using the HP energy model. In the case of more complex models, w.r.t. energy and structure, current techniques can not be easily extended and the constraint approach is not applicable to proteins of length over 100. The idea described in this paper is based on the alternation of CSP solving phases and local search phases that modify the predicted spatial conformation. The approach is implemented and tested in Gecode and EasyLocal with encouraging results.

1 Introduction

The protein structure prediction problem is recognized to be a challenging problem for computational biology. Even with strong approximations of the spatial model (simple discrete lattices) and of the energy model (simple contact energy function), the problem is proved to be NP-hard. Nevertheless, minimizing simple hydrophobic-polar energy function and using the discrete lattice model FCC (Face Centered Cube), Backofen and Will solve it in seconds for proteins of length 160 and more [1]. Moreover, other researchers (e.g. [9]) approximated the solution to the same problem using local search and refined meta-heuristics.

More complex models have been proposed for the protein structure prediction problem. In [3] the problem have been formalized in the FCC lattice using a 20x20 energy matrix (different contributions for each pair of amino acids) and using information from secondary structure (known and/or predicted presence of α-helices and β-strands). The original implementation in SICStus Prolog CLP(FD) evolved in various directions (e.g., [4]) and an ad-hoc constraint solver on lattices (COLA) has been developed [5]. However, this approach is computationally infeasible when applied to the prediction of protein structures with more than hundred amino acids. Only the presence of other kind of partial information (e.g., known folds for sub-blocks picked from the protein data bank) can speed up significantly the search.

Extended models do not translate into an easy extension of the core computation idea used in [1]. This becomes unapplicable since the presence of different kinds of

contacts generates an explosion of the number of possible cores. Moreover, the definition of optimal core does not account for any complex structural constraints (e.g. secondary structure). Any admissible conformation containing further structural constraints often can only be obtained from a suboptimal core, namely a set of contacts less packed in the space. Since the number of such cores is exponential in the number of cavities in the volume, it is infeasible to precompute them in advance.

In this paper we would like to mix constraint-based and local-search techniques to improve the performance of the above mentioned (FCC 20x20) constraint-based tools. This hybrid system combines local search techniques and constraint solving. During the computation we consider the notion of *conformation*, which is a protein representation mapped to the spatial domain (i.e., FCC). Each conformation represents a possible state of a protein and it is associated to a particular energy, directly derived from the application of the pairwise 20x20 energy function. Each conformation may be constrained to other structural properties (see [3] for a complete list).

The presence of secondary structure information, obtained through neural network prediction, is necessary in order to predict more realistic conformations. In particular, it is shown that the contact energy function is not sufficient to reproduce local arrangements such as helices and/or sheets. The secondary structure information compensates the roughness of the energy model in use. Another advantage from using secondary structure constraints is that the search space reduces, since rigid blocks with no internal degree of freedom are imposed in the conformation.

The idea is to alternate CSP solving phases to local search phases. In the former, given a conformation as input, a CSP is built in order to search a spatially close conformation which respects every structural constraint (e.g., two amino acids may not overlap).

In the latter phase, the conformation is altered by means of a set of moves, which rotate part of the protein using a specific amino acid as pivot of the rotation. The part of the protein rotated is weakly allowed to change shape, in order to satisfy the overall conditions (i.e. the block is not kept fully rigid).

The platform is implemented and tested in Gecode and EasyLocal. Gecode is a recent C++ constraint solving platform with excellent performances [7], while EasyLocal++ is an object oriented, general and configurable, local search tool [6]. We compared the pure constraint programming approach and the one that combines constraint programming and local search on 12 proteins with different length and structure: even without developing particular combination strategies, the conformations found by the hybrid method improve those found with the pure CP approach.

2 Modeling PF in Gecode

As first test, we encoded in GECODE the same model presented in [3], using some enhanced representations for rigid substructures like helices and sheets [4]. We briefly summarize here the essential aspects of the encoding, which is based on the schema presented in [2]. The interested reader can refer to the just cited references.

The *Primary* structure of a protein is a sequence $s = s_1 \dots s_n$, where each s_i is an amino acid identified by a letter of an alphabet $\mathcal{A}$, $|\mathcal{A}| = 20$. The 3D conformation

of the protein is named *Tertiary* structure. *Tertiary* structures often contain *Secondary Structure elements* (e.g., α-helices and β-sheets).

We model the protein on the FCC lattice, namely, each amino acid i occupies a position $\omega(i)$ in the lattice. FCC points are points $\langle x, y, z\rangle \in \mathbb{N}^3$ such that $x + y + z$ is even. Two FCC points $\langle x_1, y_1, z_1\rangle$ and $\langle x_2, y_2, z_2\rangle$ are

- contiguous (or `next`) iff $|x_1 - x_2| \leq 1, |y_1 - y_2| \leq 1, |z_1 - z_2| \leq 1, |x_1 - x_2| + |y_1 - y_2| + |z_1 - z_2| = 2$.
- in `contact` iff they are not contiguous *and* $|x_1 - x_2| + |y_1 - y_2| + |z_1 - z_2| = 2$.

The choice of using the FCC lattice has been often adopted in thermodynamical studies of stability of small proteins (e.g. [10]) and this lattice is able to represent with a certain degree of accuracy the typical backbone angles and the shape of secondary structures. A *folding* of s is a function $\omega : \{1, \ldots, n\} \rightarrow \mathcal{D}$ such that:

1. $\texttt{next}(\omega(i), \omega(i + 1))$ for $i = 1, \ldots, n - 1$, and
2. $\omega(i) \neq \omega(j)$ for $i \neq j$ (namely, ω introduces no loops).

The second property is encoded using the well known `alldifferent` constraint, after a conversion from 3D coordinates to 1D FD variables. The conversion is based on an enumeration of the 3D lattice, using a relation of the kind $V = xM^2 + yM + z$, where M is a sufficiently large number. As shown in [5], using distinct FD variables for each coordinate hampers the propagators effectiveness and thus the `alldifferent` constraint has a limited effect. However, we based this approach on FD variables, instead of 3D box domains an in [5], in order to ease the interaction with GECODE. Let `Pot` be a 20x20 matrix associating an energy contribution measure to each pair of amino acids types s_i and s_j. The contribution is accounted for when $\omega(i)$ and $\omega(j)$ are in contact.

The *protein structure prediction problem* can be modeled as the problem of finding the folding ω of S such that the following energy cost function is minimized:

$$E(\omega, S) = \sum_{1 \leq i < n} \sum_{i+2 \leq j \leq n} \texttt{contact}(\omega(i), \omega(j)) \cdot \texttt{Pot}(s_i, s_j).$$

Let us observe that in the FCC each point is adjacent to 12 neighboring points. However, as explained in [3], we add some extra constraints (e.g. `angles`) that restrict to 90° and 120° the bend angles between three consecutive amino acids.

`secondary_info` constraints encode the Secondary Structure information in the program. The secondary structure is described by a list of elements of the type:

`helix`(i, j)**:** $s_i, s_{i+1}, \ldots, s_j$ form an α-helix. The modeling of α-helices builds on the observation that it is sufficient to constrain the first 4 amino acids of the helix to guarantee its shape—the shape can then be propagated to the rest of the helix via simple vector equalities [4].

`strand`(i, j)**:** $s_i, s_{i+1}, \ldots, s_j$ are in a β-strand. Similar to the case above.

`ssbond`(i, j)**:** presence of a disulfide bridge between s_i and s_j. If $\langle x_1, y_1, z_1\rangle$ and $\langle x_2, y_2, z_2\rangle$ are the variables for the positions of the two amino acids, then we set the constraints $|x_1 - x_2| \leq 4, |y_1 - y_2| \leq 4, |z_1 - z_2| \leq 4$.

We summarized each protein main features in a file format. Every protein is described by: its ID (according to the name used in the protein data bank), the sequence S of amino acids and its secondary structure information (if any).

We run some tests in order to compare the ability of FD solvers to handle the CSP described. We removed on purpose every heuristics and search optimizations described in [4, 5] and we noticed that with the same search parameters Gecode outperforms the running time of the equivalent SICStus Prolog code by a rather constant speedup. This search as well as the hybrid approach produce an output file that can be handled by standard molecular viewers. Complete code is available at www.dimi.uniud.it/dovier/PF/LS.

3 Local Search Moves

In this section we describe the Local Search perturbations that form the second phase in the hybrid technique. The local modifications of a conformation are defined by a set of moves that maps a conformation into another one.

3.1 The pivot move

A convenient move we studied is the *pivot move*, which is proved to be ergodic [8]. The idea of this class of moves is to keep unchanged the first part of the protein (for example the first half) and to rotate the second one. The second part should be rotated in the FCC space as a rigid block while looking for a better associated energy cost. A pivot move is identified by:

- the *pivot* amino acid (s_i), the last amino acid of the part that remains unchanged;
- a *firstfixed* amino acid (s_j), that identifies the rotating part of the protein (thus $i < j$). Below we explain why we require $i + 1 < j$.
- some *rigid block constraints*, that constrain the position of the amino acids of the moving part of the protein (from s_j to s_n).

A good move is influenced by the selection of the *pivot*. In particular is preferable to select an amino acid not involved in α-helices and β-strands: in fact such amino acids are in the middle of a well-structured section of the protein that must not be modified (e.g. it is not possible to break apart a helix).

The firstfixed amino acid identifies a section of the the protein (between s_i and s_j) completely free to move in the FCC lattice (only structural constraints are active, e.g. `next`). A firstfixed amino acid too close to the *pivot* (e.g. $s_j = s_i + 1$) limits the possible rotations of the rigid part of the protein, due to the non overlap constraints and to the poor degree of freedom of the subsequence between s_i and s_j. As this subsequence is enlarged, the possible accommodations of the subsequent rigid block increase exponentially. However, a firstfixed too far from the *pivot* causes the exploration of the huge search space for the subsequence $s_i \dots s_j$.

Lastly, the rigid block constraints must be selected carefully. They are a set of distance constraints between all pairs of amino acids in the rotating block as in the input conformation. The constraints can be relaxed (e.g. distances within a range w.r.t. the original distance, reduced number of pairs) and this case allows multiple solutions

which are spatially close to the original conformation. Once again, the degree of relaxation influences the search space and thus the solution times. On one hand, an exact rigid block set of constraints reproduces exactly the block, but, once rotated, it is not tolerant to local modifications of the block to avoid some overlaps to the first part of the protein. On the other hand the complete absence of rigid block constraints (the second part of the protein is totally free to move in the lattice) causes an inefficient search for the next conformation and every information about the second part of the protein is lost. We have experimentally chosen an intermediate approach, where *some constraints* between the amino-acids in the rigid block are added (obtaining a semi-rigid block). Observe that in this way we naturally mix local search and constraint based search.

We performed various preliminary test, to identify the better combination of these parameters. We decided to select as *pivot* only the amino acids *not* involved in α-helices or β-strands, to select as the firstfixed amino acid the fifth one after the *pivot* (i.e. $s_j = s_i + 5$) and to post distance constraint on the rigid block only between the amino acids of distance six in the primary structure (reduced number of pairs).

3.2 Pivot move implementation

To implement the pivot move we start from a conformation p of the protein encoded into a Gecode object (a Gecode::Space object). We create a new Gecode::Space object representing a protein *nextp*, where we post all the spatial and structural constraint (FCC lattice, `next`, `no loops`, `angles` and `secondary_info` constraints). Then we copy the amino acids $s_1 \dots s_i$ from p to $nextp$; the amino acids from *pivot* + 1 to firstfixed- 1 of *nextp* ($s_{i+1} \dots s_{j-1}$) are only subject to structural constraint; then we post the rigid block constraints on the amino acids of *nextp* from firstfixed to the last one ($s_j \dots s_n$).

Once the new protein object is created and all these constraints are posted, the Gecode search routine is launched for *nextp*. This CP search explores all the possible conformations (with respect to the constraints posted), trying to reach a folding with a better energy cost than the one of *p*. If the search finds such a folding, we iterate the process, starting from the conformation of the protein reached in *nextp*.

3.3 The Local Search algorithm

We inserted the implementation of the pivot move into a basic local search algorithm, using the functionalities provided by the framework EasyLocal++. The main idea of the algorithm is to start from an admissible conformation obtained as the first solution of the constraint programming search, then to randomly select a pivot move and to search a new conformation with better energy, according to the selected move using constraint programming search.

The CSP search (both for the first solution and for the pivot move) invokes a labeling with a *leftmost* variable selection and *median* value selection. We investigated various labeling options and observed experimentally that these ones better fit our problem.

The search on a local move has a timeout, that we call *moveTimeout* (we used a value of 1 minute); if a new conformation with a better energy is discovered before

moveTimeout, it is accepted and it becomes the current one. A *globalTimeout* is selected at the beginning of the execution, so the algorithm iterates until it reaches the *globalTimeout*. At the end, the best solution reached is returned. The iteration of the process depicts a classical hill-climbing algorithm.

When selecting the move, the choice of *pivot* amino acid determines firstfixed amino acid and rigid block constraints). As said above the *pivot* amino acid is randomly selected only among all the amino acids of the protein not involved in α-helices and/or β-strands. During the random moves exploration, we ensure that the same move is not tested many times. We need to keep track of the moves already tested, in order to skip the candidate moves in the history.

Once every move have been tested and no move produces an improvement in the energy cost, the *moveTimeout* is multiplied by a constant *Inc* (we used *Inc*=2) and the process is iterated.

4 Results

After preliminary tests performed with the aim of tuning the various parameters (firstfixed, rigid block constraints, timeouts, search strategies), we tested our hybrid algorithm on 12 proteins of different length and with different secondary structures (the same used in [5]). For each protein we executed 1 run with the pure CP algorithm and 5 runs with the hybrid approach: the selection of a pivot move is randomly guided, so different runs may lead to different solutions and we report results on the best run obtained. In Table 1 we report for each protein the search time in minutes and the energy cost (Ecost) of the best conformation found.

Tests have been performed on a AMD Opteron 280 at 2.2GHz, Linux CentOS machine with a *globalTimeout* of 2 hours. We compare the energy cost of the solution found with the pure constraint programming approach and with the hybrid constraint programming-local search algorithm. The energy costs do not account for the contribution of the internal contacts of the secondary structures, since they are constant during the search process, and thus these results are not comparable to the ones of [3, 4, 5].

We can notice that the energy costs found with the hybrid approach are generally better than the ones found with the only use of constraint programming. This confirms our hypothesis that the hybrid approach leads to better solutions in the same amount of time. With some small proteins the hybrid algorithm stops improving after few minutes: in this case, it falls into a local minimum and neither increasing the *moveTimeout* nor trying different runs can avoid this problem. On the other hand, with some proteins the pure CP search stops finding better solutions in few minutes, because the search space to explore is too big and the search diverges. It must be noticed that the energy values obtained for longer proteins are not yet satisfactory.

Work needs to be done in the local search stage: in fact we noticed that some simple and useful rotations between contiguous secondary structures are not performed; such rotations are probably forbidden by the blocks overlap (the rigidity should be be further relaxed) and by an insufficient number of free amino acids between two contiguous secondary structures.

Protein		CP		CP + LS		Protein		CP		CP + LS	
ID	Length	Time	Ecost	Time	Ecost	ID	Length	Time	Ecost	Time	Ecost
1EDP	17	15	-12279	1	-12140	2IGD	60	70	-7583	87	-16631
1E0N	27	39	-7619	5	-12742	1SN1	62	4	-20764	59	-28853
1VII	36	16	-9194	1	-14402	1L6T	78	107	-23883	52	-7117
2GP8	40	53	-12472	20	-13501	1HS7	96	30	-5797	1	-2067
1ED0	45	7	-10917	8	-14747	1TQG	104	49	-8384	117	-15333
1ENH	54	63	-8928	59	-12386	1SA8	105	67	-14219	120	-26443

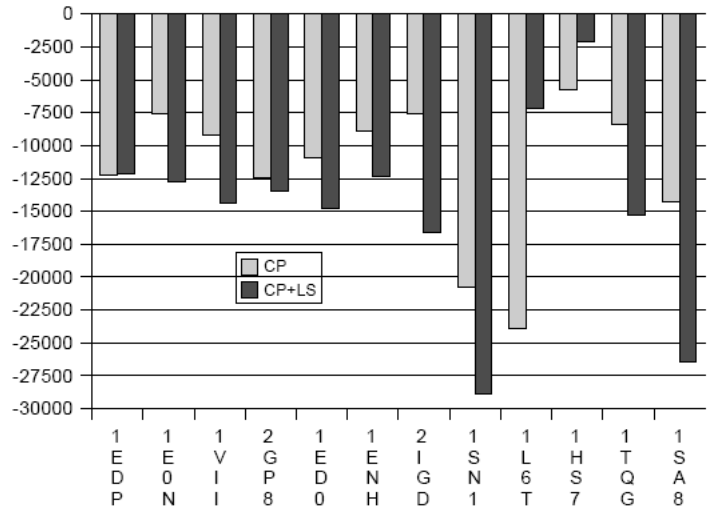

Table 1. Comparison of the solutions obtained by the two approaches on 12 different proteins. Timings are expressed in minutes.

5 Future Work and Conclusions

This is an ongoing work. Our aim was to prove that on the PF problem on FCC lattice with 20x20 energy matrix, the hybrid use of local search and constraint programming outperforms the only use of constraint programming, in terms of quality of solutions and execution time. We first encoded a basic PF model on FCC with 20x20 energy matrix into the constraint programming framework Gecode, without including strong search heuristics (like the ones used in [4, 5]); then we defined a local search move (the pivot move) in the local search framework EasyLocal++, in such a way that the pivot move can interact with the constraint programming model. Our tests confirm that the hybridization of these techniques leads to better solutions with respect to the pure constraint programming model.

Now that we have ensured the feasibility of this idea and the goodness of the results, additional tests and algorithm improvements can be performed. We plan to run the algorithm on other longer and more complex proteins, to refine the parameters with

massive testing, if needed. We can also try to run our algorithm with a longer globalTimeout. Various ideas can be applied to the hybrid algorithm. For example, we plan to embed into the constraint programming model some already tested heuristics (the ones used in [4, 5]): this should improve the performance when searching for a new conformation, and thus speed up the search.

We can refine the local search strategy: the hill-climbing algorithm is very efficient, but it is a local algorithm; the use of more refined strategies (such as tabu search) could avoid falls into local minima. We also think to elaborate more complex local search heuristics and metaheuristics, derivable from the EasyLocal++ framework.

We can speed-up the performance using the COLA solver [5], a constraint solver in C specifically designed for the Protein folding problem: embedding local search routines directly into COLA, instead of using Gecode and EasyLocal++ should outperform the execution time of the present current approach.

Acknowledgements The work is partially supported by MUR FIRB RBNE03B8KK and PRIN projects. We thank Luca Di Gaspero and Andrea Formisano for the useful discussions and the help in installing packages.

References

[1] R. Backofen and S. Will. A Constraint-Based Approach to Fast and Exact Structure Prediction in Three-Dimensional Protein Models, *Constraints*, 11(1):5–30, 2006

[2] P. Clote and R. Backofen. *Computational Molecular Biology: An Introduction*. John Wiley & Sons, 2001.

[3] A. Dal Palù, A. Dovier, and F. Fogolari. Constraint logic programming approach to protein structure prediction. *BMC Bioinformatics*, 5(186), 2004.

[4] A. Dal Palù, A. Dovier, and E. Pontelli. Heuristics, optimizations, and parallelism for protein structure prediction in CLP(FD). Proc. of PPDP 2005: 230-241, 2005.

[5] A. Dal Palù, A. Dovier, and E. Pontelli. A constraint solver for discrete lattices, its parallelization, and application to protein structure prediction. *Software Practice and Experience*, DOI: 10.1002/spe.810, 2007.

[6] L. Di Gaspero and A. Schaerf. EasyLocal++: an object-oriented framework for the flexible design of local-search algorithms. *Software Practice and Experience*, 33(8):733–765, 2003.

[7] Gecode Team. Gecode: Generic Constraint Development Environment. *Available from http://www.gecode.org*, 2006.

[8] N. Madras and A. D. Sokal, The Pivot Algorithm: A Highly Efficient Monte Carlo Method for the Self-Avoiding Walk. *Journal of Statistical Physics*, 50:109–186, 1988.

[9] A. Shmygelska and H. H. Hoos. An ant colony optimisation algorithm for the 2D and 3D hydrophobic polar protein folding problem. *BMC Bioinformatics* 6(30), 2005.

[10] J. Skolnick and A. Kolinski. Reduced models of proteins and their applications. *Polymer*, 45:511–524, 2004.