Constraint based protein fragment assembly

A. Dal Palù[1], A. Dovier[2], F. Fogolari[3], and E. Pontelli[4]

[1] Dept. of Mathematics, University of Parma, alessandro.dalpalu@unipr.it
[2] Dept. of Maths and Computer Science, University of Udine, dovier@dimi.uniud.it
[3] Dept. of Biomedical Sciences and Technologies, University of Udine, federico.fogolari@uniud.it
[4] Dept. of Computer Science, New Mexico State University, epontell@cs.nmsu.edu

Abstract. We present a tool that predicts the 3D conformation of a protein, based on Constraint Logic Programming and fragment assembly. The fragments are derived from a database of known protein structures with a preprocessor —also developed for this work— that clusters and classifies the fragments according to their similarity and frequency. The fragments are used as building blocks by the CLP program for the generation of the whole conformation.
The first results demonstrate the correctness and efficiency of the method. The declarativeness of the generated code allows future extensions with more detailed energy models and search heuristics for better accuracy.

1 Introduction

Proteins are central components in the way they control and execute the vital functions in living organisms. Their functions are due to their peculiar three-dimensional conformation. Proteins are assembled in cells according to the sequence of DNA coding for them. Adoption of the most stable functional conformation is performed mostly spontaneously.

Knowledge of the three-dimensional conformation of a protein (also known as the *native conformation* or *tertiary structure*) is essential to biomedical investigation. The native conformation represents the functional protein and determines how it can interact with other proteins (e.g., dock) and affect the functions of the organism. It is impossible to clearly understand the behavior and phenotype of an organism without knowledge of the native conformation of the proteins coded in its genome. As a result of advancements in (DNA) sequencing techniques today there is an enormous number of protein sequences (namely, lists of amino acids, also known as *primary structures* of proteins) available in public databases (e.g., the database Swiss-prot contains ca. 500000 protein sequences). Structural information is lagging behind, with a much smaller number of structures deposited in public databases, notwithstanding structural genomics initiatives started worldwide.

For these reasons, one of the most traditional and central problems addressed by research in Bioinformatics deals with the problem of *protein structure prediction* namely, the problem of determining the native conformation of a protein

starting from its primary sequence, using computational methods. Several approaches have been explored to address this problem. At the highest level, it is possible to distinguish between two approaches. The more traditional term *protein prediction* has been commonly used to methods that rely on comparison between known and unknown structures to predict the end result of protein folding. Instead, *protein folding* typically tries to understand the folding path leading to the native conformation, typically using investigations of the potential energy landscape or using molecular dynamics simulations. In both classes of methods, knowledge of "patterns" could be used to restrict the search space—and this is particularly true for the case of so-called *secondary structure* components of a protein (namely local helices or strands). Secondary structure components are important, considering that their formation is believed to represent the earliest phase of the folding process, and their identifications can be relatively simpler (e.g., through low-resolution observations of images from cryo-electron microscopy).

Based on homology modeling or on fold recognition techniques it is possible to predict protein structures based on their sequence, but in general it is difficult to predict a protein structure based *only* on its sequence in the absence of a structural template. Explicit solvent molecular dynamics simulation of protein folding is still beyond current computational capabilities. Already in 1968, Levinthal postulated that the systematic exploration of the space of possible conformations is computationally infeasible [11]. This complexity has been confirmed by theoretical results, showing that even extremely simplified formalizations of the problem lead to computationally intractable problems [5]. Recently, ab-initio methods for generating protein structures given their sequences have been proposed and successfully used [3]. Key elements of these methods are usage of evolutionary information from multiple alignments, simplified representation of proteins and fragment assembly. In other words these methods rely on assembling the structure using simplified representation of fragments (taken from structural database) with favorable conformations for the profile of the given sequence. Three- to nineteen-residue fragments (3–9 for small proteins, 5–19 for large proteins) contain correlations among neighboring residues conformation and therefore most of the computational time is spent for searching the global arrangement of fragment displaying already good local conformations [13, 14].

Several simplified models have been introduced to address the problem. Simplified models abstract several proprieties of proteins and space to lead to a version of the problem that can be solved more efficiently; in turn, the solutions to the simplified problem constitute candidate configurations that can be refined with more computationally intensive methods, e.g., molecular dynamics simulations. Simplifications include introducing fixed sizes of monomers and bond lengths, representing monomers as simple points and viewing the three-dimensional space as a discretized collection of points. In these simplified models, it is possible to view the protein folding problem as an optimization problem, aimed at determining conformations that minimize an energy function. The energy function must be defined according to the simplified model used [4, 10].

Moreover, more simplified energy models have been employed in the context of Constraint Programming approaches to structure prediction [1].

In our previous efforts, we have applied declarative programming approaches, based on the constraint programming paradigm, to solve the problem. Using a crystal lattice organization of the allowable points in the three-dimensional space, that exploits the property that the distance between the $C\alpha$ atoms (a carbon atom that is a convenient representative for the whole amino acid) of two consecutive amino acids is rather constant (precisely, 3.8Å), the problem is viewed as placing amino acids in the allowable points in such a way that constraints encoding the mutual distances of amino acids in the primary sequence are satisfied [6, 7]. The original framework has also been expanded to support several types of *global constraints*, i.e., constraints describing complex relationships among groups of amino acids. One of these constraints is the *rigid structure* constraint—this constraint enables the representation of known substructures (e.g., helices), thus reducing the problem to finding an appropriate placement and rotation of such substructures in the lattice space. The ability to use rigid structure constraints has been shown to lead to dramatic reductions in the search space [8, 2]. However, exploiting knowledge of real rigid substructures in a discrete lattice is hard due to discretization approximations.

Some of the most successful approaches used protein folding build on the principles of using *substructures*. The intuition is that while the complete folding of a protein may be unknown, it is likely that all possible substructures, if properly chosen, can be found among known proteins. The folding is constructed by exploiting relationships among substructures, from substructures composed of a small number of amino acids to complex knowledge of secondary structure components. A notable example of this approach is represented by Rosetta—an ab initio protein structure prediction method that uses simulated annealing search to compose a conformation by assembling substructures extracted from a fragment library; the library, extracted from observed structures stored in the Protein Data Bank [13].

In this work, we follow a similar idea, by developing a database of amino acid chains of length 4; these are clustered according to similarity and the statistics of their frequencies are drawn from the investigation of a relevant section of the protein database. The database contains the data needed to construct the program that assembles the fragments. This program is computationally demanding and specific methods are employed to handle the large search space. The declarative programming paradigm is exploited to generate clean and compact code, and allowed us rapid prototyping. Moreover, the problem of assembling substructures is efficiently tackled using the constraint programming techniques embedded within the Prolog systems.

The paper has the goal of showing that our approach, implemented in a first prototype, is a viable method. The main advantage, respect to a highly engineered and imperative tool, is the modularity of the constraint system, which offers us a convenient framework to test and integrate statistical data from var-

ALA $\mapsto$ 0	LEU $\mapsto$ 1	MET $\mapsto$ 1	ARG $\mapsto$ 2	GLU $\mapsto$ 2	GLN $\mapsto$ 2	LYS $\mapsto$ 2
ASN $\mapsto$ 3	ASP $\mapsto$ 3	SER $\mapsto$ 3	THR $\mapsto$ 4	PHE $\mapsto$ 4	HIS $\mapsto$ 4	TYR $\mapsto$ 4
ILE $\mapsto$ 5	VAL $\mapsto$ 5	TRP $\mapsto$ 5	CYS $\mapsto$ 6	GLY $\mapsto$ 7	PRO $\mapsto$ 8	γ

Table 1. Clustering of amino acids into 9 classes

ious predictors and databases. Moreover, the constrained search technique itself represents an unusual method compared to popular predictors and we believe it can become effective with the development and addition of new energy functions and heuristics.

2 Modeling

We first present how we prepared the statistical data (collections of fragments made of 4 consecutive amino acids, also denoted as 4-tuples) we use for protein assembly. Then we report the main ideas of the CLP($\mathcal{FD}$) implementation, focusing on the constraint definition and on the search heuristics employed.

2.1 Clustering

We focus on proteins made of sequences of amino acids of 20 types (those coded by human genome). We are dealing with the $C\alpha$ modeling, hence every amino acid is identified by the coordinates of its $C\alpha$ atom. Although more than 60,000 protein structures are deposited in `www.pdb.org`, the whole set of known proteins contains too much redundancy (very similar proteins deposited in several variants) to be useful for statistical purposes. Subsets with few redundancy must be chosen. In particular, we have trained our tool with the so-called top-500 set [12] but the C program `tuple_generator` we developed to extract the desired information can work on any given set of known proteins, as well. This set contains 500 proteins with globally $107,138$ occurrences of amino acids which is also roughly the number of 4-tuples. However, the number of different 4-tuples occurring in the set is precisely $62,831$. Since the number of possible 4-tuples of amino acids is $20^4 = 160,000$ most of them do not appear in the set and, even those that appear occur too few times for inferring statistical information. For this reason, we clustered amino acids into 9 classes according to the similarity in the torsion angles about the pseudo bond between two consecutive $C\alpha$'s [10]. Let $\gamma(a) \in \{0, \ldots, 9\}$ be the function assigning a class to each amino acid as defined in Table 1. This way, the majority of the $9^4 = 6561$ 4-tuples have a witness into the set (precisely, there are templates for 5830 of them) and each of them occurs in average more than 60 times.

A second level of approximation is deciding when two occurrences of the same 4-tuple have the "same" form. We put them into the same class when their Root Mean Square Deviation (RMSD) is less than or equal to a given threshold, provided by a global variable `rmsd_thr`, set to 1.0Å or less. The program `tuple_generator` (written in C) returns a set of facts of the form:

$$\texttt{tupla}(\ [a_1, a_2, a_3, a_4],\ [X_1, Y_1, Z_1, X_2, Y_2, Z_2, X_3, Y_3, Z_3, X_4, Y_4, Z_4],\ \texttt{FREQ}, \texttt{ID}, \texttt{PID})$$

where $a_1, \ldots, a_4 \in \{0, \ldots, 8\}$ identify the class of each amino acid, $X_1, \ldots, Z_4$ are the coordinates of the 4 points in the selected template[5], $\texttt{FREQ} \in \{1, \ldots, 1000\}$ is the frequency of that template (expressed with integer numbers with a precision of 1/1000) w.r.t. all occurrences of the tuple $a_1, \ldots, a_4$ in the set top-500, ID is a unique identifier for this *fact* and PID is the first protein found where the template come from (this last information will be printed in the pdb file we produce as output of the computation).

Remark 1. It is rather unlikely that a sequence unknown by the set would occur in a real protein. Anyway, we can run `tuple_generator` using a unique class for amino acids, and then select a number (for instance 10) of the most probable templates and assigning them to all unknown 4-tuplets.

Pairs of `tupla` facts are related by the predicate `next`. $\texttt{next}(\texttt{ID}_1, \texttt{ID}_2, \texttt{Mat})$ holds if the sequence 2–4 of $\texttt{ID}_1$ and 1–3 of $\texttt{ID}_2$ are the same (namely their RMSD differs of at most `rmsd_thr`). `Mat` is the rotation matrix to align the second sequence on the first.

2.2 Constraints

The input is given by a list `Primary` of n amino acids (just to facilitate testing, the PDB protein identifier can also be given and its set of amino acids. In this case, the primary structure is retrieved from the data base).

Two lists of variables are generated: a list `Code` of length $n - 3$ and a list `Tertiary` of length $3n$.

A finite domain is assigned to each variable in `Code`. The i-th variable C_i of `Code` corresponds to the 4-tuple $(a_i, a_{i+1}, a_{i+2}, a_{i+3})$. The possible values for C_i are the IDs of the facts of the form

$$\texttt{tupla}([\gamma(a_i), \gamma(a_{i+1}), \gamma(a_{i+2}), \gamma(a_{i+3})], \_, \texttt{Freq}, \texttt{ID}, \_)$$

This set is also ordered using the frequency information in decreasing order, and stored in a variable $\texttt{ListDom}_i$. The `next` information is then used to impose constraints between C_i and C_{i+1}. Using the combinatorial constraint `table` we allow only pairs of consecutive values supported by the `next` predicate.

Frequency of a tuple will be exploited in the search process. For each possible value of C_i we assign a value B_i depending on the frequency, transformed using logarithms in order to use sums instead of products. This is added as follows:

```
tupla(_,_,Freq,ID,_),
Prob  is integer(log(10,Freq/1000)*1000)+3000,
C_i #= ID #=> B_i #= Prob
```

[5] Without loss of generality, we set $(X_1, Y_1, Z_1) = (0, 0, 0)$.

Since `Freq` is an integer from 1 to 1000, the lowest value is $\log_{10}(\frac{1}{1000}) = -3$, and hence `Val` $= 0$, the highest value is $\log_{10}(\frac{1000}{1000}) = 0$, and hence `Val` $= 3000$.

This list of probabilities is then subject to a series of sum constraints. Basically, after k values have been instantiated, we impose that the sum $B_1 + \cdots + B_k \geq k * \texttt{Thr}$ where `Thr` is a threshold. This ensures that admissible solutions with low probability are cut in badvance.

The list `Tertiary` contains the list of variables X, Y, Z denoting the 3D position of the $C\alpha$ atoms. X, Y, Z will assume integer values with the precision of 10^{-2}Å, and will be instantiated during the search.

2.3 Solution's search

The search is governed by the instantiation of the C_is variables. They are selected leftmost and most probable values are chosen first. Probability constraints allow to skip less frequent foldings.

After C_1 is instantiated, using

$$\texttt{tupla}(\_, [X_1, Y_1, Z_1, X_2, Y_2, Z_2, X_3, Y_3, Z_3, X_4, Y_4, Z_4], \_, C_1, \_)$$

we assign the first four points of the sequence. An auxiliary vector `Rot` is set to be the identity matrix. During the search, this matrix describes the rotation matrix needed to place a new template in the same reference as the one of the last grounded variable.

Assume now that step i is done. Then C_i and $X_i, Y_i, Z_i, X_{i+1}, Y_{i+1}, Z_{i+1}$, $X_{i+2}, Y_{i+2}, Z_{i+2}, X_{i+3}, Y_{i+3}, Z_{i+3}$ are now known (ground) and `Rot` contains the reference for the last instantiated template. Variable C_{i+1} is then instantiated. Using

$$\texttt{tupla}(\_, \texttt{Tuple}, \_, C_{i+1}, \_), \texttt{next}(C_i, C_{i+1}, \texttt{Mat})$$

we have all ingredients to compute the point $X_{i+4}, Y_{i+4}, Z_{i+4}$ (using the rotation matrix `Mat`, the matrix `Rot`, and the known point $X_{i+3}, Y_{i+3}, Z_{i+3}$). The matrix `Rot` is premultiplied with `Mat`, in order to update the new local reference, while `Tuple` is rotated according to the new reference. Finally the rotated `Tuple` is shifted, so that the (ground) point $X_{i+3}, Y_{i+3}, Z_{i+3}$ overlaps with the third point of the rotated `Tuple`. In practice, C_{i+1} is rotated according to the best rotation that overlaps C_i and C_{i+1} and it is placed so that the last point in C_{i+1} is at exactly 3.8Å from the last point in C_i.

Two constraints are added to govern the search:

`all_distant` The list tertiary $X_1, Y_1, Z_1, \ldots, X_n, Y_n, Z_n$ is subject to a constraint aimed to verifying the `all_distant` property: the $C\alpha$ of each pair of non-consecutive amino acids must be distant at least $D = 3.0$Å. Therefore, the constraint:

$$(X_i - X_j) * (X_i - X_j) + (Y_i - Y_j) * (Y_i - Y_j) + (Z_i - Z_j) * (Z_i - Z_j) \ \texttt{\#>=} \ D * D$$

is set for all $i \in \{1, \ldots, n-2\}$ and $j \in \{i+2, \ldots, n\}$.

`diameter` We allow an input parameter that is used to bound the maximum distance between different $C\alpha$s (the diameter of the protein). As explained in [6] where we introduced the `compact factor` constraint, a default value can be expressed by $5.68\, n^{0.38}$ Å. Similarly to the case above, we impose this constraint to all different pairs of $C\alpha$ points.

We associate an energy to each computed structure. Let Pot be the potential matrix that associate a potential value to a contact between two amino acids, as defined in [4]. Then the energy is computed as:

$$\sum_{i=1}^{n-2} \sum_{j=i+2}^{n} \mathsf{Pot}(a_i, a_j)\delta(X_i, Y_i, Z_i, X_j, Y_j, Z_j)$$

where $\delta(X_i, Y_i, Z_i, X_j, Y_j, Z_j) = 1$ if $\sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2 + (Z_i - Z_j)^2} \leq 3.8$Å while it is $\frac{3.8^2}{(X_i-X_j)^2+(Y_i-Y_j)^2+(Z_i-Z_j)^2}$ otherwise.

Each computed structure is saved in `pdb` format. This is a standard format for proteins that can be used by all the viewers (e.g., Rasmol, ViewerLite)

3 Experimental results

The Prolog $CLP(\mathcal{FD})$ program we developed has been run with SICStus prolog 4.0 on a two Intel Core Duo 2.66GHz processors, Ubuntu machine. The images reported in this section were produced with Accelrys ViewerLite 5.0.

The tool can search the first admissible solution (call `pf_id(ID,Tertiary)`), where `ID` is a protein name included in the database (`prot_list.pl`). It is also possible to generate the first `N` solutions and output them as many different models in a single pdb file (`pf_id_all(ID,N)`). Alternatively, it is possible to create different files, one for each model, with `pf_id_enumerate(ID,Tertiary)` and specify a bound on some parameters (energy, diameter, probability etc.). A version of the Prolog code together with a set of experimental tests of the prototype is available at `www.dimi.uniud.it/~dovier/PF/` (off-lattice methods).

We report here the tests on two different sequences: a shorter one (PDB code 1ZDD) made of 34 amino acids and another sequence taken from the CASP 8 competition (target T0397) made of 151 amino acids. We set the probability threshold `thr` equal to 0.5 and diameter less than 50Å.

The goal of the tests using the protein 1ZDD (Figure 1—left represent the deposited structure) is to show that the whole framework is viable.

The first one is a simple test that takes the templates forming the protein (plus some others) and tries to re-assemble the protein. Previously, we generated a template set from a pool of only 10 protein (instead of the `top-500`), including 1ZDD. We then enumerated all solutions obtained by combining the clustered templates and show that the actual protein can be effectively reproduced, even if some of the original 1ZDD templates were merged in classes with other representatives, coming from other proteins. The small variations in the different models in Figure 1 (middle) are due to the choice of different templates forms and to

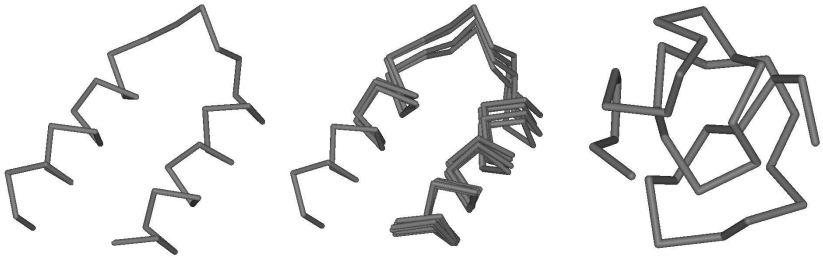

Fig. 1. 1ZDD: deposited (left), reconstruction test (middle), small diameter (right)

the clustering threshold. Increasing the `rmsd_thr` would reduce the number of choices (and of produced models) and at the same time increase the deviation of the model from the actual one.

The second test shows that the reduction of the `diameter` parameter, in order to produce more compact solution with better energy, is a delicate issue. In the example (see Figure 1 at the right), we used the `top-500` database (without 1ZDD) and set `diameter` = 14Å. The resulting protein lost the main characteristics, namely the presence of two alpha helices, because of the restricted space. We tested an additional energy contribution based on preferred torsional angles of specific amino acids, and noted that this conformation would have been strongly penalized by this contribution, which accounts for other chemical properties (such as hydrogen bonds). In future work we plan to integrate torsional energy contributions introduce an integration between energy computation and search heuristics, to guide the search towards the most sensible conformations from the energetic point of view. It is worth noting that in both experiments the time needed to compute the solution was less than 10ms.

The next set of tests shows that the method is able to deal with typical protein size (as in the CASP competition). We launched this test with the `top-500` templates (the `tuple_generator` C code processing time took less than 15 seconds). The CLP program is able to handle the size of the protein in an efficient way: a single solution can be generated in a few milliseconds. We then tested the program with an enumeration of probable solutions. In addition to the constraints described in the previous sections, we added a constraint, verified at each level expansion i, where the number m of templates that are not the most probable ones is limited, namely $m < 1 + 8\sqrt{i/151}$ (151 is the protein length). Basically, this constraint allows to generate conformations mainly made of the most frequent form for each template, while, for a small fraction of templates (in this test at most 8), the form is chosen among the least frequent ones. This way, it is possible to explore the whole search space without getting stuck in a subtree of similar conformations (typical when using leftmost search).

After 12 minutes of computation the search was completed and a set of 500 structures was produced. The RMSD of the best structure (Figure 2 right) is 13.98Å w.r.t. the original one (Figure 2 left).

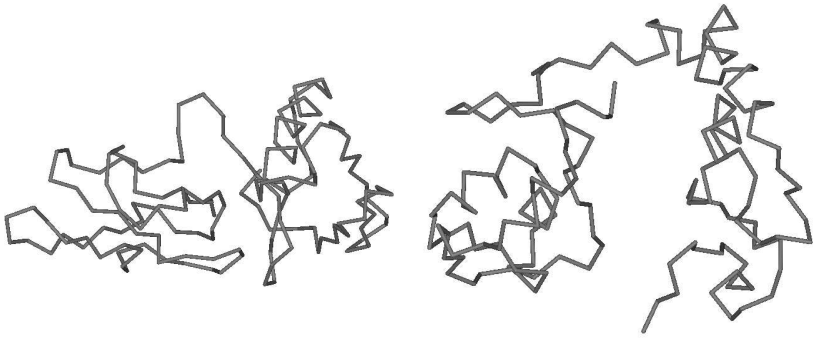

Fig. 2. CASP T0397 target (left) and our result (right)

Our results show that the method can scale well, considering that tools like Rosetta are more computationally demanding for proteins longer than 100 amino acids (due to the simulated annealing process). From the quality point of view, the structure does present some secondary structure feature, but is is not arranged in the correct way. This is an evidence that a more refined energy (like, e.g., [10]) model should be integrated in our tool.

4 Conclusions and future work

In this paper we presented the design and implementation of a constraint logic programming tool that is able to predict the tertiary structure of a protein, given the primary structure and based on fragment assembly. The method retrieves the templates from a protein database, that is clustered according to shape similarity. We used templates of 4 amino acids length, but longer lengths could be easily incorporated, since their statistically significance could carry strong constraints for conserved shapes.

Few constraints are currently considered in our tool (basically: probability, `all_distant`, and `diameter`). Other constraints should be analyzed (the programming style easily allows this kind of extensions). Moreover, some operations that are executed during the search only when variables become ground (e.g., matrix operations) can be replaced by constraint versions that, typically operate a little bit slower but allow constraint propagation that can, in principle, reduce sensibly the size of the search tree. We will work on this topic, also introducing the energy function as a constraint (although we believe that minimizing the energy performs poorly in pruning the search tree). Moreover, as told in Section 3, in this stage we will also substitute the energy function used in this paper with a more precise one.

As future work we plan to investigate more refined models and including more atoms per amino acids (for example two atoms and/or all atoms). This does not affect the code structure, while the number of solutions and the complexity may vary significantly.

Our approach is not computationally limited by the size of the protein, however the quality of the results is to be improved. As future work, we plan to combine quality and efficiency both with new constraints and search heuristics. We noted that from the structural point of view, the models computed are reasonable, but a more realistic energy model and other statistical consideration should be introduced as new constraints. This can be accomplished thanks to the declarativeness of our approach (a main difference from a tool like Rosetta).

References

[1] R. Backofen and S. Will. A Constraint-Based Approach to Fast and Exact Structure Prediction in 3-Dimensional Protein Models. *Constraints* 11(1):5–30, 2006.

[2] P. Barahona and L. Krippahl. Constraint Programming in Structural Bioinformatics. *Constraints* 13(1–2):3–20, 2008.

[3] M. Ben-David, O. Noivirt-Brik, A. Paz, J. Prilusky, J. L. Sussman, Y. Levy Assessment of CASP8 structure predictions for template free targets. Assessment of CASP8 structure predictions for template free targets. *Proteins*, (in press).

[4] M. Berrera, H. Molinari, and F. Fogolari. Amino acid empirical contact energy definitions for fold recognition in the space of contact maps. *BMC Bioinformatics*, 4(8), 2003.

[5] P. Crescenzi, D. Goldman, C. Papadimitriou, A. Piccolboni, M. Yannakakis. On the complexity of protein folding. In *STOC*:61–62, 1998.

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

[7] 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* 37(13):1405–1449, 2007.

[8] A. Dal Palù, A. Dovier, E. Pontelli. Enhancing the Computation of Approximate Solutions of the Protein Structure Determination Problem Through Global Constraints for Discrete Crystal Lattices. Proc. of Computational Structural Bioinformatics Workshop Nov. 4, San Jose, CA. (vol. 1, pp. 38-44) 2007.

[9] F. Fogolari, G. Esposito, P. Viglino, and S. Cattarinussi. Modeling of polypeptide chains as C-α chains, C-α chains with C-β, and C-α chains with ellipsoidal lateral chains. *Biophysical Journal*, 70:1183–1197, 1996.

[10] F. Fogolari, L. Pieri, A. Dovier, L. Bortolussi, G. Giugliarelli, A. Corazza, G. Esposito, and P. Viglino. Scoring predictive models using a reduced representation of proteins: model and energy definition. *BMC Structural Biology*, 7(15), 2007.

[11] C. Levinthal. Are there pathways in protein folding? In *J. Chim. Phys.*, vol 65:44–45, 1968.

[12] S. Lovell, I. Davis, W. Arendall, P. de Bakker, J. Word, M. Prisant, J. Richardson, and D. Richardson, Structure validation by C_α geometry: ϕ, ψ and C_β deviation. *Proteins* 50:437–450, 2003.

[13] S. Raman, R. Vernon, J. Thompson, M. Tyka, R. Sadreyev, J. Pei, D. Kim, E. Kellogg, F. DiMaio, O. Lange, L. Kinch, W Sheffler, B.-H. Kim, R. Das, N. V. Grishin, and D. Baker. Structure prediction for CASP8 with all-atom refinement using Rosetta. *Proteins*, (in press).

[14] S. Wu, J. Skolnick, and Y. Zhang Ab initio modeling of small proteins by iterative TASSER simulations. *BMC Biology*, 5(17), 2007.