Constraint Based Protein Structure Prediction Exploiting Secondary Structure Information

Alessandro Dal Palù[2], Sebastian Will[1], Rolf Backofen[1], and Agostino Dovier[2]

[1] Jena Center of Bioinformatics, Institute of Computer Science, Friedrich-Schiller-University, Jena. Ernst-Abbe-Platz 2, 07743 Jena (Germany).
[2] Department of Mathematics and Computer Science, University of Udine. Via delle Scienze 206, 33100 Udine (Italy).

Abstract. The protein structure prediction problem is one of the most studied problems in Computational Biology. It can be reasonably abstracted as a minimization problem. The function to be minimized depends on the distances between the various amino-acids composing the protein and on their types. Even with strong approximations, the problem is shown to be computationally intractable. However, the solution of the problem for an arbitrary input size is not needed. Solutions for proteins of length 100–200 would give a strong contribution to Biotechnology. In this paper, we tackle the problem with constraint-based methods, using additional constraints and heuristics coming from the secondary structure of a protein that can be quickly predicted with acceptable approximation. Our prototypic implementation is written using constraints over finite domains in the Mozart programming system. It improves over any previous constraint-based approach and shows the power and flexibility of the method. Especially, it is well suited for further extensions.

1 Introduction

A protein is identified by a finite list of amino-acids, which we can represent as symbols of an alphabet of 20 elements. The *protein structure prediction (PSP)* problem is the problem of predicting the 3D structure of the protein (its *native conformation*) when the list of amino-acids is known. Native conformation determines the Biological function of the protein. It is accepted that the native conformation is the state of minimum free energy. Up to now, though, a definitive energy model has not been yet devised. The energy of a conformation depends partially on the *distances* between all pairs of amino-acids and on their *type*. Thus, the PSP problem can be simplified to that of minimizing a suitable energy function generated by the protein 3D conformations. Although the problem, even with some simplifications, is NP-complete [9], it deserves to be attacked, because a solution for 'small' proteins (100–200 amino-acids) would be anyhow extremely important in Biology and Biotechnology. The problem is approached in several ways (see [7, 14] for a review). Prediction methods make use of statistical information available from more than 24,000 structures deposited in the Protein Data Bank (PDB) [5]. The correct fold for a new sequence can be obtained when *homology* (sequence similarity) is detected with a sequence for

which the structure is available. Another approach tries to superimpose (*thread*) a chain on known structures and evaluates the plausibility of the fold.

Ab-initio methods, instead, try to find the native conformation without direct reference to a structural model. In this context, a constraint-based encoding of the problem is extremely natural. A first abstraction is the choice of a spatial model for the admissible positions of the various amino-acids. *Lattice models* are used to formalize the PSP problem as a minimization problem on finite domains. It has been shown [4, 3, 12] that the Face-Centered-Cubic lattice (FCC) provides the best lattice approximation of proteins. Moreover, it has been shown that the residue neighbors in real proteins are clustered in a rather dense way, occupying positions closely approximating those of a distorted FCC packing. It is believed that this packing is a direct manifestation of the hydrophobic effect. Recall that FCC is the closest packing of spheres, which was proven only recently [8].

The FCC lattice is exploited in [1, 16] for solving proteins of length up to 160 with the further abstraction of splitting the amino-acids in two families (H and P). However, the HP abstraction does not ensure that the result is the native conformation; in particular, the local sub-conformations of the form of α-helices or β-strands (cf. Sect. 2) are often lost. These structures are often generated early in the real folding process and, moreover, the structures can be predicted with good approximation [13]. These two observations suggest to include secondary structures in the constraint-based definition of the (complete, non-HP) problem. This is done in [10] using $CLP(\mathcal{FD})$ constraints in SICStus Prolog; proteins of length 50–60 can be predicted.

In this paper we extend the results of [10] in three ways. First, a precise mathematical formalization of secondary structure is given and results are obtained, which allow for easy breaking of symmetries, cheap computation of energy, and an effective, but computationally inexpensive, search strategy during the constraint search. Then we split the energy function into 4 families and we observe that the energy contribution of one of them is sensibly greater than the others: we use this information for developing heuristics for the solution's search process. Finally, we take advantages from using the constraint propagation of Oz 3.0 language and in particular we implement specially tailored propagators that allow to solve the problem efficiently. The preliminary results obtained show a huge speed-up w.r.t. the results of [10].

The paper is organized as follows: in Section 2 we provide some biological background needed through the paper. In Section 3 we describe the spatial and protein models. In Section 4 we define the problem and introduce our solving strategy. In Sections 5 and 6 we provide technical details to handle secondary structure elements. In Section 7 we describe the constraint framework. In Section 8 we present some preliminary results and we conclude with Section 9.

2 Biological Background

The *Primary* structure of a protein is a finite sequence of linked units (or *residues*), that define uniquely the molecule. Each residue is an amino-acid,

denoted by one of the 20 elements in the alphabet set Σ. A *protein (primary) sequence* is thus a string $s \in \Sigma^*$.

Native conformations are largely built from *Secondary Structure elements (*SSE*s)*, which are local motifs consisting of short, consecutive parts of the amino acid sequence having a very regular conformation. Some of them are α-helices, β-sheets, and $\beta\alpha\beta$ turns. In this paper we model the first two motifs: α-helices are constituted by 5 to 40 residues arranged in a regular right-handed helix with 3.6 residues per turn. This local structure is stabilized by local interactions and can be thought as a rigid cylinder. β-sheets are constituted by extended strands of 5 to 10 residues. Each strand is made of contiguous residues, but strands participating in the same sheet are not necessarily contiguous in sequence. There are algorithms based on neural networks that can predict with high accuracy (75% [7]) the secondary structure of a protein. Formally, a *secondary structure for a protein sequence* $s = s_1 \dots s_n$ is a list sse of k triples $SSE_i = (t_i, b_i, e_i)$, where for $0 \leq i < k$, $t_i \in \{\boldsymbol{\alpha}, \boldsymbol{\beta}\}$, $0 \leq b_i < e_i < n$, and for $0 \leq i < k-1$, $e_i < b_{i+1}$. Later, we will use the notation $|s| = n$, and $|\,\mathrm{sse}\,| = k$. Given a secondary structure sse for a protein sequence s, then the *sub-sequence of the i-th* SSE is the string $s_{b_i} \dots s_{e_i}$.

In nature, each protein always reaches a specific 3D conformation, called native conformation or *tertiary structure*. This conformation determines the function of the protein. The *protein structure prediction problem* is the problem of determining the tertiary structure of a protein given its primary structure. It is accepted that the primary structure uniquely determines the tertiary structure. Due to entropic considerations, it is also accepted that the tertiary structure minimizes the global energy of the protein. Though, is not yet uniquely accepted which energy function describes this phenomenon.

3 Formalizing the Models

3.1 The Lattice Model

In this section, we introduce the spatial model and its properties. The protein is represented as a succession of three dimensional points. Each point corresponds to an amino-acid[3]. In our approach we restrict the point's domain to be in the *face-centered-cubic lattice (FCC)*. Fig 1a) shows its unit cell. The FCC is the set of points $\mathsf{FCC} = \{(x, y, z) | x, y, z \in \mathbb{Z}, x + y + z \text{ is even}\}$.

We discuss now some important properties of this lattice. First, note that the FCC has 48 automorphisms, which are represented by orthogonal matrices M, where the column vectors are a permutation of $(\pm 1, 0, 0), (0, \pm 1, 0)$, and $(0, 0, \pm 1)$. As we now will explain, we use only 24 of these automorphisms, due to the chirality of proteins.

Chirality is an important property of protein structures and sub-structures. Two objects are *chiral* if they are identical except for a mirror reflection (different *handedness*). In nature, a sequence of amino-acids, when folded (e.g. right

[3] In particular, we associate each point to the position of the α-carbon lying in the backbone of the amino-acid, which can be assumed to be its steric center.

handed α-helices), has a specific handedness and usually it can not generate the symmetric folding as well. In our model, when we apply transformations, we are interested in preserving the natural handedness. Since reflections invert the handedness, we restrict to automorphisms with an orthogonal matrix with positive determinant. Note that half of the possible automorphisms are reflections, since the determinant associated to the matrix is negative. This leads us to the following definition of the basic transformation that we apply to points:

Definition 1. *A matrix M is called* rotational*, if it is orthogonal,* $\det(M) = 1$ *and each of its elements is in $\mathbb{Z}$. An* isometric mapping $\langle M, \boldsymbol{t}\rangle$*, where $\boldsymbol{t} \in \mathsf{FCC}$ and M is a rotational matrix, is a function from points to points of the form $\langle M, \boldsymbol{t}\rangle : \boldsymbol{p} \to M\boldsymbol{p} + \boldsymbol{t}$. We identify this function with its extension to sets of points. M is called the* transformation matrix *of the isometric mapping $\langle M, \boldsymbol{t}\rangle$ and $\boldsymbol{t}$ its* translation vector*.*

The composition $A_2 \circ A_1$ of two isometric mappings $A_2 = \langle M_2, \boldsymbol{t}_2\rangle$ and $A_1 = \langle M_1, \boldsymbol{t}_1\rangle$, i.e. the application of A_2 after A_1 equals

$$\langle M_2, \boldsymbol{t}_2\rangle \circ \langle M_1, \boldsymbol{t}_1\rangle = \langle M_2 M_1, M_2\boldsymbol{t}_1 + \boldsymbol{t}_2\rangle.$$

Note that $A_2 \circ A_1$ is again an isometric mapping, since M_2M_1 is rotational. Due to this definition, there is also an inverse for an isometric mapping $\langle M, \boldsymbol{t}\rangle$, namely $\langle M, \boldsymbol{t}\rangle^{-1} \triangleq \langle M^{-1}, -M^{-1}\boldsymbol{t}\rangle$. We define the FCC-norm of a point (x, y, z):

$$\|\boldsymbol{p}\|_{\text{fcc}} = \max\left\{|x|, |y|, |z|, \frac{|x| + |y| + |z|}{2}\right\}.$$

A vector $\boldsymbol{v} \in \mathsf{FCC}$ is called a *unit vector* if $\|\boldsymbol{v}\|_{\text{fcc}} = 1$. The FCC-distance of two points $\boldsymbol{p}, \boldsymbol{q} \in \mathsf{FCC}$ is $\|\boldsymbol{p} - \boldsymbol{q}\|_{\text{fcc}}$. Let us observe that in the FCC lattice, each walk from $(0, 0, 0)$ to $(x, y, z) \in \mathsf{FCC}$, needs at least $\|(x, y, z)\|_{\text{fcc}}$ lattice unit vectors. We also use the standard Euclidean norm $\|\langle x, y, z\rangle\|_2 = \sqrt{x^2 + y^2 + z^2}$.

3.2 The Protein Model

We provide here the formal definition of our 3D model and energy function. The tertiary structure of a protein can be modeled as a function $\omega : [0\,..\,n-1] \to \mathsf{FCC}$. The position of the i-th monomer in the protein corresponds to the point $\omega(i)$.

Definition 2. *The function ω is a* folding *for the primary sequence s, iff*
bond-constraint: $\forall i\,.\,0 \le i < |s| - 1 : \|\omega(i) - \omega(i+1)\|_{fcc} = 1$*, and*
angle-constraint: $\forall i\,.\,0 \le i < |s|-2 : \angle(\omega(i), \omega(i+1), \omega(i+2)) \in \{90°, 120°, 180°\}$.

Note that we allow only angles of 90°, 120°, and 180°, whereas in the FCC lattice, three consecutive monomers can also form angles of 0° and 60°. We exclude the angles less than 90° for sterical reasons, since the first and third amino-acid would be too close. Note that in [10] the 180° angle was excluded as well, since it is unfavorable in real proteins. In our approach we allow this angle, since simpler constraints can be used (especially when linking SSEs to

neighbors). Moreover, when discretizing the protein on FCC lattice, 180° angles can be required to fit more accurately the native state. For example, the modeling of α-helices in the FCC lattice, is not able to represent the typical periodicity of the pattern.

A folding ω *satisfies* a secondary structure $\text{sse} = (t_i, b_i, e_i)_{0 \le i < |\,\text{sse}\,|}$ if and only if for every $i \in [0..|\,\text{sse}\,| - 1]$, the positions $\omega(b_i), \ldots, \omega(e_i)$ approximate a right-handed α-helix on the FCC if $t_i = \boldsymbol{\alpha}$ and approximate a β-strand on the FCC if $t_i = \boldsymbol{\beta}$. We give a formal definition of these approximations in Subsection 5.1.

The modeling of the folding energy is a delicate issue: a large set of phenomena can be included to produce a refined energy function. In this paper we restrict to one kind of interaction between elements, namely the *contact energy* between pairs of amino-acids and we use the matrix developed in [6]. $\text{ppot}(a, b)$ denotes the potential of the amino-acids a and b. The potential is only contributed to the total energy if the two amino-acids are in close contact. If two monomers are too close, they are *clashing*, in this case the total energy is ∞, since for steric reasons two residues repel each other.

Definition 3 (Energy). *For two amino-acids $a, b \in \Sigma$ and two positions $\boldsymbol{p}, \boldsymbol{q} \in$ FCC, we define*

$$\mathrm{E}(a, b, \boldsymbol{p}, \boldsymbol{q}) = \begin{cases} \text{ppot}(a, b) & \|\boldsymbol{q} - \boldsymbol{p}\|_2 = 2 \\ \infty & \|\boldsymbol{q} - \boldsymbol{p}\|_2 < 2 \\ 0 & \textit{otherwise.} \end{cases}$$

The energy of a protein conformation with protein sequence s and folding $\omega : [0\,..\,|s| - 1] \to \text{FCC}$ *is* $\mathrm{E}^{\mathrm{C}}(s, \omega) = \sum_{0 \le i, j < |s|, i+2 < j} \mathrm{E}(s_i, s_j, \omega(i), \omega(j))$.

Note that due to our definition, every sub-sequence up to 3 consecutive amino-acids in the primary structure, does not contribute to the energy. This is an empirical choice. Basically, if three consecutive amino-acids formed an angle of 90° the first and third amino-acid would form a pair contributing to the energy, whereas if they formed an angle of 120° there would be no contribution. The energy contribution of an amino-acid pair is negative on average and thus, angles of 90° would be always favoured. This is true in the lattice, but not in nature, and thus we avoid this a-priori preference by removing the energy contribution of the pair of amino-acids s_i and s_{i+2}.

We report here a part of the potential table of [6], also available at `http://www.dimi.uniud.it/dovier/PF/`.

	CYS	MET	PHE	ILE	LEU	VAL	TRP	TYR	ALA	GLY
CYS	-3.477	-2.240	-2.424	-2.410	-2.343	-2.258	-2.080	-1.892	-1.700	-1.101
MET	-2.240	-1.901	-2.304	-2.286	-2.208	-2.079	-2.090	-1.834	-1.517	-0.897
PHE	-2.424	-2.304	-2.467	-2.530	-2.491	-2.391	-2.286	-1.963	-1.750	-1.034
...	...									

4 Problem Definition and Solving Strategy

We define the protein structure prediction problem when the secondary structure information is taken into account. Given a sequence s and a secondary structure

sse, the problem is to find the folding ω that satisfies *sse* and has minimal energy $\mathrm{E}^{\mathrm{C}}(s, \omega)$. The energy function can be partitioned as a sum of four energy terms, namely the energy contribution by pairs of amino-acids, where

1. both are in the same SSE,
2. both are in SSEs, but not in the same,
3. one is in a SSE and the other is not,
4. both are not in SSEs.

Formally, these contributions, whose sum is $\mathrm{E}^{\mathrm{C}}(s, \omega)$ are defined as

1. $\mathrm{E}^{\mathrm{s}}(s, \mathrm{sse}, \omega) = \sum_{0 \le r < |\,\mathrm{sse}\,|} \sum_{b_r \le i+2 < j \le e_r} \mathrm{E}(s_i, s_j, \omega(i), \omega(j))$
2. $\mathrm{E}^{\mathrm{ss}}(s, \mathrm{sse}, \omega) = \sum_{0 \le r < r' < |\,\mathrm{sse}\,|} \sum_{b_r \le i \le e_r} \sum_{b'_r \le j \le e'_r, i+2 < j} \mathrm{E}(s_i, s_j, \omega(i), \omega(j))$
3. $\mathrm{E}^{\mathrm{sn}}(s, \mathrm{sse}, \omega) = \sum_{i \in \mathrm{D}} \sum_{0 \le j < |s|, j \notin \mathrm{D}, i+2 < j} \mathrm{E}(s_i, s_j, \omega(i), \omega(j))$
4. $\mathrm{E}^{\mathrm{nn}}(s, \mathrm{sse}, \omega) = \sum_{0 \le i < j < |s|, i, j \notin \mathrm{D}, i+2 < j} \mathrm{E}(s_i, s_j, \omega(i), \omega(j))$,

where D is the set of positions in SSE, i.e. $\bigcup_{0 \le r < |\,\mathrm{sse}\,|} [b_r \,..\, e_r]$. The first term $\mathrm{E}^{\mathrm{s}}(s, \mathrm{sse}, \omega)$ is constant for each folding ω that satisfies a given secondary structure sse. Thus, optimizing $\mathrm{E}^{\mathrm{C}}(s, \omega)$ it is equivalent to optimize $\mathrm{E}^{\mathrm{ss}}(s, \mathrm{sse}, \omega) + \mathrm{E}^{\mathrm{sn}}(s, \mathrm{sse}, \omega) + \mathrm{E}^{\mathrm{nn}}(s, \mathrm{sse}, \omega)$.

From a set of 500 selected PDB-proteins, we estimated the average contributions of the last three terms to their sum in the native state, which is given by the following distributions: E^{ss} 49% , E^{sn} 36%, and E^{nn} 15%. Note that half of the energy is contributed by interaction between SSEs. This suggests a heuristic to solve the structure prediction problem. Thus, first we place the SSEs optimally according to E^{ss}. Then, for fixed SSEs, we place the remaining amino-acids while optimizing $\mathrm{E}^{\mathrm{C}}(s, \omega)$.

During the optimization of E^{ss}, the SSEs are placed in the FCC lattice. The elements are treated as rigid blocks that can be shifted and oriented in the lattice. It is fundamental to be independent from global rigid transformations, which give the same isomorphic results. In the next section, we introduce the notion of relative position between two SSEs. This concept is the base to abstract from the symmetries of the problem. Note that all energy terms are based on the notion of distance between amino-acids, thus they are invariant under rigid transformations. The relative position description, thus, is suitable to be associated to a specific energy contribution provided by the represented class.

5 Absolute and Relative Positions

In this section, we discuss a formalism to describe the SSEs and their placement in the lattice. We specify an object (later denoted by *template*) by a list of FCC points. The object is placed in the space by transforming these points by means of rotating and translating. This transformation defines the absolute position of the placed object (later denoted by *instance*). We also introduce the relative position of two SSEs, which is the transformation required to map the absolute position of the first element into the one of the second. Fig. 1b) provides an illustration of the concepts that are introduced in this section.

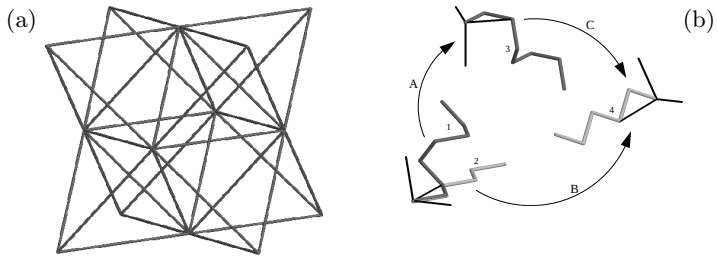

Fig. 1. a) Unit Cell of the face-centered cubic lattice (FCC). There is one point in each corner of a cube and one point in the center of each face. We show the connections by unit vectors. b) Idea of absolute and relative positions. The figure shows instances a helix template of length 8 and type 0 and a sheet template of length 6. If the instances 1 and 2 are in absolute position id, then A (resp. B) is the absolute position of the instance 3 (resp. 4). $C = \rho(A, B)$ is the relative position between the instances (from A to B). We show local coordinate systems for each instance to illustrate the transformation.

5.1 Templates

A *template* is a function $T : [0\,..\,\ell - 1] \to \mathsf{FCC}$, where $\ell \in \mathbb{N}$ is its *length*. We introduce two classes of templates, namely α-helices and β-sheets. In these cases, a template is the geometric description of a helix (sheet) starting from $(0, 0, 0)$.

Definition 4 (Templates for SSEs). *Let a function* $h : \mathbb{N} \to \mathsf{FCC}$ *be given by* $h(4i+0) = (0,0,0) + i\,\boldsymbol{d}_h$, $h(4i+1) = (1,0,1) + i\,\boldsymbol{d}_h$, $h(4i+2) = (2,0,0) + i\,\boldsymbol{d}_h$, *and* $h(4i+3) = (2,1,-1) + i\,\boldsymbol{d}_h$ *for* $i \in \mathbb{N}$, *where* $\boldsymbol{d}_h = (2,2,0)$. *Then, the* helix template of length n and type τ, *where* $n \in \mathbb{N}$ *and* $\tau \in \{0,1\}$, *is the function* $\mathrm{helix}_n^\tau : [0\,..\,n-1] \to \mathsf{FCC},\ \mathrm{helix}_n^\tau(i) = h(i-\tau) \quad (i \in [0\,..\,n-1])$.

Furthermore, let a function $\mathrm{sheet}_n : [0\,..\,n-1] \to \mathsf{FCC}$, *be given by* $s(2i+0) = (0,0,0) + i\,\boldsymbol{d}_s$ *and* $s(2i+1) = (1,1,0) + i\,\boldsymbol{d}_s$ *for* $i \in \mathbb{N}$, *where* $\boldsymbol{d}_s = (2,0,0)$.

Note that the helix templates of type 0 (resp. 1) describe helices, where the first three points form an angle of 90° (resp. 120°).

5.2 Position of an Instance

Definition 5 (Absolute Position, Instance). *An* absolute position *is an isometric mapping. An* instance I with absolute position $\langle M, \boldsymbol{t}\rangle$ of a template T *is a pair* $I = \langle M, \boldsymbol{t}\rangle \lozenge T$. *The* instance function of I, *denoted by* I^{fun}, *is* $\langle M, \boldsymbol{t}\rangle \circ T$.

For $i \in \mathbb{Z}$, *we use the short notations* $I(i)$ *for* $I^{fun}(i)$ *and* $\mathrm{dom}(I)$ *for* $\mathrm{dom}(I^{fun})$. *The* image of an instance I *is* $\mathrm{img}(I) = \{I(i) | i \in \mathrm{dom}(I)\}$.

The instance function of the instance $\langle \mathrm{id}, (0,0,0)\rangle \lozenge T$ is $\langle \mathrm{id}, (0,0,0)\rangle \circ T = T$.

Definition 6 (Relative Position). *For instances* I_i *with absolute positions* A_i $(i = 1, 2)$, *the* relative position $\rho(A_1, A_2)$ from A_1 to A_2 *is the isometric mapping, where* $\rho(A_1, A_2) = A_1^{-1} \circ A_2$. *Then,* $\rho(A_1, A_2)$ *is also called the* relative position of the instances I_1 and I_2.

The relative position between two instances $I_i = A_i \Diamond T_i$ $(i = 1, 2)$ can be used to obtain the second instance from the first instance as $I_2 = A_1 \circ \rho(A_1, A_2) \Diamond T_2$, since by definition $A_2 = A_1 \circ \rho(A_1, A_2)$.

The following proposition claims that the relative position of two instances is invariant under global transformations.

Proposition 1. *Given the instances $I_i = A_i \Diamond T_i, I'_i = A'_i \Diamond T'_i$ (i=1,2), the relative positions $R = \rho(A_1, A_2), R' = \rho(A'_1, A'_2)$ and the isometric mappings C_i, such that $A'_i = C_i \circ A_i$, then $R = R'$ if and only if $C_1 = C_2$.*

Note that by Proposition 1, $\rho(M \circ A_1, M \circ A_2) = \rho(A_1, A_2)$ for isometric mappings M, A_1, and A_2. Especially, for every relative position $\rho(A_1, A_2)$, there is an identical relative position of the form $\rho(\mathrm{id}, B) = B$ for some isometric mapping B, namely $B = A_1^{-1} \circ A_2$.

6 Energy contribution of instance pairs

In the first phase of the algorithm, we pre-compute the energy contribution of a pair of instances in every relevant placement. Therefore, we enumerate the set of relative positions between the instances of two templates, where the two instances interact sterically. Only for those relative positions, the energy contribution is not equal to zero.

Definition 7 (Interaction). *Two instances I_1 and I_2* interact*, if and only if there exist $i_1 \in \mathrm{dom}(I_1), i_2 \in \mathrm{dom}(I_2)$, such that $\|I_2(i_2) - I_1(i_1)\|_2 \leq 2$. We define the* interaction set of templates T_1 and T_2 *as* $\mathrm{InteractionSet}(T_1, T_2) =$

$$\{R = \rho(\mathrm{id}, R) \mid I_1 = \mathrm{id} \Diamond T_1, I_2 = R \Diamond T_2, I_1 \text{ and } I_2 \text{ interact}\}.$$

Note that in the definition, we have in mind to fix the first instance (here to id) and move the second instance to every interacting position. Nevertheless, due to Proposition 1, the interaction set contains all relative positions between arbitrary instances that are interacting. Note that the interaction set is finite, since the instances are finite.

If we define neighVecs as a tuple of the 19 vectors $\boldsymbol{p} \in \mathsf{FCC}$ with $\|\boldsymbol{p}\|_2 \leq 2$ in arbitrarily fixed order, then for a template T, we define an *extended template* T^{ext} by $T^{\mathrm{ext}}(19 \cdot i + j) = T(i) + \mathrm{neighVecs}_j$, for $i \in \mathrm{dom}(T)$ and $0 \leq j < 19$.

We say that two instances *intersect* if and only if their images have a non-empty intersection.

Proposition 2. *Two instances $I_1 = A_1 \Diamond T_1$ and $I_2 = A_2 \Diamond T_2$ interact if and only if $I_1^{ext} = A_1 \Diamond T_1^{ext}$ and I_2 intersect.*

Definition 8 (Energy Contribution). *Two instances of templates T_i with sequence s_i $(i = 1, 2)$ with relative position R give an energy contribution of*

$$\mathrm{E}^{\mathrm{T}}(s_1, s_2, T_1, T_2, R) = \sum_{0 \leq j < |s_1|, 0 \leq k < |s_2|} \mathrm{E}(s_{1j}, s_{2k}, \mathrm{id} \Diamond T_1(j + b_1), R \Diamond T_2(k + b_2)), \quad (1)$$

where $b_1 = \min(\mathrm{dom}(T_1))$ and $b_2 = \min(\mathrm{dom}(T_2))$.

Recall that for the templates T_1 and T_2, there are only finitely many relative positions R, such that $\mathrm{id} \lozenge T_1$ and $R \lozenge T_2$ interact, i.e. $\mathrm{InteractionSet}(T_1, T_2)$ is finite. By merging the two definitions of interaction and the energy contribution of a pair of amino-acids, if $\mathrm{E}^{\mathrm{T}}(s_1, s_2, T_1, T_2, R)$ is different from 0, then $\mathrm{id} \lozenge T_1$ and $R \lozenge T_2$ interact. Note that from the interaction of $\mathrm{id} \lozenge T_1$ and $R \lozenge T_2$, we can not conclude that $\mathrm{E}^{\mathrm{T}}(s_1, s_2, T_1, T_2, R) \neq 0$, since depending on the table of pairwise potentials certain interaction patterns of the two instances could sum up to 0.

Due to this, in order to completely give the energy for every relative position R it suffices to consider all $R \in \mathrm{InteractionSet}(T_1, T_2)$. Now, the problem discussed in this subsection reduces to generate $\mathrm{InteractionSet}(T_1, T_2)$. Instead of enumerating the relative positions, where instances of T_1 and T_2 interact, we equivalently enumerate the relative positions, where instances of T_1^{ext} and T_2 intersect. Due to the following proposition, we can completely enumerate the set $\mathrm{InteractionSet}(T_1, T_2)$.

Proposition 3. *For templates T_1 and T_2,*

$$\mathrm{InteractionSet}(T_1, T_2) = \biguplus_{M \text{ rot. matrix}} \{\langle M, \boldsymbol{t}\rangle | \exists j_1, j_2 : \mathrm{id} \lozenge T_1^{ext}(j_1) = \langle M, \boldsymbol{t}\rangle \lozenge T_2(j_2)\}.$$

Proof. Obviously, one can partition any interaction set by the rotation matrices of the relative positions. Then, the proposition is a consequence of Proposition 2.

For the inclusion $\subseteq$, let $R = \langle M, \boldsymbol{t}\rangle$ in $\mathrm{InteractionSet}(T_1, T_2)$. Then, the instances $\mathrm{id} \lozenge T_1$ and $R \lozenge T_2$ interact, i.e. the instances $\mathrm{id} \lozenge T_1^{\mathrm{ext}}$ and $R \lozenge T_2$ intersect.

For the inclusion $\supseteq$, for any $R = \langle M, \boldsymbol{t}\rangle$ in one of the subsets $\{\langle M, \boldsymbol{t}\rangle | \exists j_1, j_2 : \mathrm{id} \lozenge T_1^{\mathrm{ext}}(j_1) = \langle M, \boldsymbol{t}\rangle \lozenge T_2(j_2)\}$, the instances $\mathrm{id} \lozenge T_1^{\mathrm{ext}}$ and $R \lozenge T_2$ intersect. Thus, $\mathrm{id} \lozenge T_1$ and $R \lozenge T_2$ interact. □

The proposition suggests the following algorithm. For every rotation matrix M, for every pair of indices $j_1 \in \mathrm{dom}(T_1)$ and $j_2 \in \mathrm{dom}(T_2)$, collect the unique translation vectors $\boldsymbol{t}$, where $\mathrm{id} \lozenge T_1^{\mathrm{ext}}(j_1) = \langle M, \boldsymbol{t}\rangle \lozenge T_2(j_2)$.

Note that there is indeed a unique $\boldsymbol{t}$ for every M, j_1 and j_2, which is calculated in constant time. Also note that whereas this algorithm can enumerate a vector $\boldsymbol{t}$ more than once, the algorithm still calculates only a limited number of $24 \times |\mathrm{img}(T_1^{\mathrm{ext}})| \times |\mathrm{img}(T_2)|$ many vectors $\boldsymbol{t}$.

7 Constraint Model

Recall, that we discuss the problem of finding the folding ω for a given protein sequence s and a given secondary structure $\mathrm{sse} = (t_i, b_i, e_i)_{0 \leq i < |\,\mathrm{sse}\,|}$, that satisfies sse and has minimal energy $\mathrm{E}^{\mathrm{C}}(s, \omega)$. We already suggested a heuristic for the minimization, which consists of two separate phases and uses the outcome of the already described computation of energy contributions of instances.

In a first branch-and-bound search, we place the secondary-structure elements (SSEs) while optimizing the energy contribution E^{ss}, i.e. we search for a folding ω that satisfies sse and has minimal energy $\mathrm{E}^{\mathrm{ss}}(s, \mathrm{sse}, \omega)$ and a finite

total energy $\mathrm{E}^{\mathrm{C}}(s,\omega)$. For finite energy $\mathrm{E}^{\mathrm{C}}(s,\omega)$, there must not be clashes in the tertiary-structure. When we place only the SSEs, we can not check the non-clashing of the remaining amino-acids efficiently by consistency methods. Hence, we search for one consistent placing of the remaining amino-acids each time we find a new placement of the SSEs. Then, a second branch-and-bound search computes a placement of SSEs and remaining amino-acids that optimizes the total energy $\mathrm{E}^{\mathrm{C}}(s,\omega)$ and places the SSEs nearly optimally w.r.t. E^{ss}.

All the constraints and the enumeration strategy are common for the two phases. Due to this, we are able to give only a common description as well as to implement only a single solver in our implementation language *Oz 3.0* [15].

The constraint model as well as our enumeration strategy divides into two parts. The first part deals with the placement of the SSEs, whereas the second part handles the placement of the remaining amino-acids. The focus of our work is on the first part and only this part shall be described in more detail.

In the first part, we use the pre-computed energy contributions of instance pairs. Therefore, we start with relating our placement problem to the notion of templates and instances.

We define templates for every SSE in sse. For each SSE k, we introduce an absolute position A_k and a type $\tau_k \in \{0,1,2\}$, where $0 \leq k < |\,\mathrm{sse}\,|$. Such a type τ_k combines the distinction between α-helix and β-sheet with the types of helix-templates (cf. Def 4). For a helix, this type gives just the type 0 or 1 of the corresponding helix-template and for sheets this type is always 2. For $0 \leq k < |\,\mathrm{sse}\,|$, where $t_k = \boldsymbol{\alpha}$, we define $T_k^{\tau} = \mathrm{helix}^{\tau}_{e_k - b_k + 1}$ $(\tau = 0, 1)$ and where $t_k = \boldsymbol{\beta}$, we define $T_k^2 = \mathrm{sheet}_{e_k - b_k + 1}$. Let s_k denote the sub-sequence of the k-th SSE. We define I_k^{τ} as $A_k \Diamond T_k^{\tau}$ for absolute positions A_k. Now, the tertiary-structure ω is related to the instances, by $\omega(i) = A_k \Diamond T_k^{\tau_k}(i - b_k)$ for $i \in [b_k \,..\, e_k]$.

Then, for given template definitions, the positions of amino-acids in SSEs are completely specified by $(A_k)_{0 \leq k < |\,\mathrm{sse}\,|}$ and $(\tau_k)_{0 \leq k < |\,\mathrm{sse}\,|}$. Due to this, we can equivalently investigate the problem in terms of instances. The relation between the instances determines the energy term E^{ss}. In the same time the relations are constrained by the properties of a folding.

We choose to enumerate the relative positions of the elements instead their absolute positions.[4] Besides breaking of symmetries[5], there are several advantages in enumerating relative positions instead absolute positions. Most notably, there is a direct correspondence between the relative position and the energy contribution of a pair of SSEs. This immediate relation is used to dynamically guide the search, i.e. we enumerate highly contributing relations between SSEs first. Furthermore, enumerating relative positions is more general than enumerating absolute positions. Imagine, that we enumerate absolute positions. Then, after w.l.o.g. fixing the absolute position of the first instance to $\langle \mathrm{id}, \mathbf{0} \rangle$, enumerating the absolute positions of the remaining elements is equivalent to enumerating

[4] Notably, the energy contribution E^{ss} is determined only from the relative positions.

[5] Note that by using relative positions instead absolute positions, we break all geometrical symmetries in the problem for free. In general, breaking of symmetries in constraint modeling is a non-trivial task and a broadly discussed topic (e.g. see [2]).

their relative positions to the first element. In contrast, our more flexible enumeration strategy can dynamically decide to enumerate relations earlier that contribute stronger to the total energy than any relation to the first element.

As described in the previous section, we are able to calculate the energy contribution of two instances in every relative position and in particular enumerate the finite list of relative positions, where this energy contribution differs from zero. For every pair of SSEs in sse, we generate such a list of relative positions and corresponding energy contributions. Since for α-helices there are two types of helix-templates, we also include the information on the type in the list for each pair of SSEs. Here, it is convenient to partition the list into two tables: NTab_{ij} and CTab_{ij}. In the former table, we include every relative position that produces a finite, non-zero energy (i.e. there is an interaction but no clash for this relative position), and in the latter we collect all relative positions, where the elements i and j clash (infinite energy). In preparation of our enumeration strategy, the tables NTab_{ij} are ordered by increasing energy-values. For being uniform, we generate for every pair of secondary-structure elements i and j, the tables NTab_{ij} and CTab_{ij}, which consist of all records $(\sigma_i, \sigma_j, R, E)$, where σ_i (resp. σ_j) is a possible value for the type τ_i (resp. τ_j) and $R \in \mathrm{InteractionSet}(T_i^{\sigma_i}, T_j^{\sigma_j})$. Then, E is the corresponding energy contribution $\mathrm{E}^{\mathrm{T}}(s_i, s_j, T_i^{\sigma_i}, T_j^{\sigma_j}, R)$. [6]

7.1 Variables and Constraints

First, for $0 \leq i < j < |\,\mathrm{sse}\,|$ we introduce finite-domain variables $\mathtt{X}_{ij}$, where the domain of the variable $\mathtt{X}_{ij}$ is $[0\,..\,|\,\mathrm{NTab}_{ij}\,|-1] \uplus \{\Delta\}$. The value of $\mathtt{X}_{ij}$ is either an index in the table NTab_{ij} or Δ. In the first case, the relation (i.e. relative position R_{ij} *and* helix-types) between the SSEs i and j is specified by the $\mathtt{X}_{ij}$-th entry in the table NTab_{ij}. The case $\mathtt{X}_{ij} = \Delta$ represents those relative positions that provide no energy contribution, but cause a distance of elements i and j that still allows to connect the elements in a folding.

Since the variables $\mathtt{X}_{ij}$ completely specify the relative positions only for $\mathtt{X}_{ij} \neq \Delta$, we introduce an explicit representation of relative positions for ensuring consistency. The relative position between the elements i and j is given by variables $\mathtt{R}_{ij}^{M}$, which encodes for the transformation matrix of R_{ij}, and $\mathtt{R}_{ij}^{x}$, $\mathtt{R}_{ij}^{y}$, and $\mathtt{R}_{ij}^{z}$, which represent the coordinates of the translation vector of R_{ij}. The domain of $\mathtt{R}_{ij}^{M}$ is finite, since there are only 24 rotational matrices. Also the variables $\mathtt{R}_{ij}^{x}$, $\mathtt{R}_{ij}^{y}$, and $\mathtt{R}_{ij}^{z}$ have finite-domains, since the number of translations is limited due to the bond-constraint connecting the amino-acids between elements i and j. Since the tuple $(\mathtt{R}_{ij}^{M}, \mathtt{R}_{ij}^{x}, \mathtt{R}_{ij}^{y}, \mathtt{R}_{ij}^{z})$ represents one relative position, we address this tuple as the variable $\mathtt{R}_{ij}$. For $0 \leq i < |\,\mathrm{sse}\,|$, we add variables $\mathtt{Type}_i \in \{0, 1, 2\}$, which correspond to the types τ_i of the SSE*s*. If $t_i = \boldsymbol{\alpha}$, the value of $\mathtt{Type}_i$ gives the type of the helix 0 or 1. For $t_i = \boldsymbol{\beta}$, we set $\mathtt{Type}_i = 2$. The variables $\mathtt{R}_{ij}$, $\mathtt{Type}_i$ and $\mathtt{Type}_j$ are related to the variable $\mathtt{X}_{ij}$ via

[6] Note that in these tables we represent the rotational matrices of the relative positions by unique indices in the interval $[0\,..\,23]$. This representation is also used to model domains of matrices with integer finite domain variables.

the table NTab_{ij}. This relation is handled by a native propagator[7], which does only cheap propagation, if sufficiently many variables are ground. Furthermore, it checks the validity of the encoded isometric mapping (w.r.t. clashes and the bond-constraint). Also the relation between the relative position variables $\mathtt{R}_{ij}$ that corresponds to the transitivity $R_{ij} \circ R_{jk} = R_{ik}$ is only propagated in such simple cases.

Finally, the energy contribution E^{ss} is computed in a variable $\mathtt{Energy}^{\mathtt{ss}}$ as the sum of variables $\mathtt{Energy}_{ij}$. These variables denote the energy that is contributed by the elements i and j in the relation that is specified by $\mathtt{X}_{ij}$.

The essential propagation in our approach is done by a propagator for the transitivity constraint $R_{ij} \circ R_{jk} = R_{ik}$ on the variables $\mathtt{X}_{ij}$. This propagator has to relate the indices in the domains of the variables $\mathtt{X}_{ij}$ to the corresponding relations. Moreover, it has to handle Δ-values in the domains correctly. For this reason, this propagator distinguishes several cases. *Case 1)* At least two of the variables are determined to Δ. In this case no further propagation can be applied. *Case 2)* The domains of all three variables contain Δ. Then, at this stage we can not derive any information. *Case 3)* The domains of exactly two variables contain Δ. Here, the domain without Δ is used to prune the other two domains. W.l.o.g. let us assume that $\mathtt{X}_{jk}$ and $\mathtt{X}_{ik}$ contain Δ. We now describe how to prune the domain of $\mathtt{X}_{jk}$. For every isometric mapping B indexed by $\mathtt{X}_{jk}$ we check if there exists at least one isometric mapping A indexed by $\mathtt{X}_{ij}$ such that $A \circ B$ is indexed by a value of $\mathtt{X}_{ik}$. If there is no support, the index is removed from the domain of $\mathtt{X}_{jk}$. The analogous procedure is applied to prune $\mathtt{X}_{ik}$. *Case 4)* The domain of at most one variable contains Δ. W.l.o.g. assume that $\mathtt{X}_{ik}$ contains Δ. Then, we collect the composition of every pair from the domains of $\mathtt{X}_{ij}$ and $\mathtt{X}_{jk}$ and intersect the result with the domain of $\mathtt{X}_{ik}$. Moreover, we prune the domains of $\mathtt{X}_{ij}$ and $\mathtt{X}_{jk}$ using the technique of the previous item. Note that during this propagation, we compose only isometric mappings from the tables with consistent types in the same table rows.

To give some implementation details, note that in Oz, every index variable has a range that begins from 1. Moreover, since Oz does not support negative values in finite domains, we shift the domains of variables $\mathtt{R}^x_{ij}, \mathtt{R}^y_{ij}, \mathtt{R}^z_{ij}$ and in all energy variables accordingly. For convenience, we represent the Δ value differently for each i, j with the integer value $|\,\mathrm{NTab}_{ij}\,|$. Due to the complex propagation of the transitivity constraint for the variables $\mathtt{X}_{ij}$, we preferred to implement the propagator in C++, recalling it inside the Mozart code.

7.2 Search Strategy

For placing the SSEs, we enumerate the variables $\mathtt{X}_{ij}$ in a dynamic enumeration order, which combines a first-fail strategy with a preference for variables $\mathtt{X}_{ij}$ that allow a strong contribution to the energy E^{ss}. Here, we find the best contribution that is *allowed by the variable* $\mathtt{X}_{ij}$, if we look up in the table NTab_{ij} using the lowest possible value of $\mathtt{X}_{ij}$ as index. Recall that the tables NTab_{ij} are ordered

[7] In Oz, a native propagator is a propagator implemented in C++. For the purpose of writing special propagators, Oz is extensible via its C++-interface [11].

by increasing energy, in particular for this purpose. As value we will always select the minimal value in the domain, which again corresponds to the optimal energy contribution of the elements i and j.

We will only enumerate the variables $\mathtt{X}_{ij}$ in order to find a good placement of SSEs. Note that since we can assign Δ to variables $\mathtt{X}_{ij}$, not every assignment of these variables completely determines a placement. However, we will only consider placements that are completely determined after the enumeration of the variables $\mathtt{X}_{ij}$. This is justified as a heuristic, since we only search for energetically good placements and such placements will have many strong interactions between the elements. In this situation, there are usually many variables $\mathtt{X}_{ij} \neq \Delta$, which then determine the remaining relative positions.

In practice, we apply also a filtering of the domains of the variables $\mathtt{X}_{ij}$ by their energy. Since the special element Δ also represents the filtered elements, the constraint problem is not changed, except that the energy is calculated less accurately. Filtering these domains has two conflicting impacts. Whereas strong filtering on the domain sizes results in faster enumeration, larger domains achieve a stronger propagation (due to the transitivity constraints). For that reason, we apply two filters. First, we filter the tables NTab to contain only entries with energies in a range of $x\%$ of the optimal energy. The filtered tables constitute the domains of the variables $\mathtt{X}_i j$. The second filtering affects only the enumeration, where we enumerate only values within a range of $y\%$ ($y < x$) of the optimum. With careful filtering we are able to reduce search times, while preserving a good quality of the results. The strategy is justified, since good placements have usually sufficiently many high scoring pairwise energy contributions.

After the SSEs are placed, we compute absolute positions of their amino-acids by fixing the absolute position of the first element (thereby breaking the symmetries). Then, the positions of the remaining amino-acids are enumerated using a first-fail strategy combined with a preference for good energy contribution. Constraints ensure non-clashing and calculate the remaining energy term $\mathrm{E}^{\mathrm{nn}} + \mathrm{E}^{\mathrm{sn}}$ in a variable $\mathtt{Energy}^{\mathtt{nss}}$.

8 Results

We implemented the described constraint-model in the programming language Oz 3.0 [15] using its most recent implementation Mozart 1.3.0. We extended the language by special constraint propagators written in C++. Also the computation of the energy contributions of instance-pairs is implemented in C++. The implementations are available via `http://www.bio.inf.uni-jena.de`.

For evaluating our result, we ran the prediction for proteins with known structure from the PDB [5]. All predictions are performed on a Pentium 4 at 2.4GHz. Figure 2 shows our prediction for the protein domain "Maternal effect protein Staufen" with PDB-code 1STU in comparison to the known tertiary structure. The protein consists of 68 residues, forming two α-helices and three β-sheets. The computation of the energy contribution of instances was performed in 14 seconds. In the first search, we found a good placement of the SSEs in 3.5

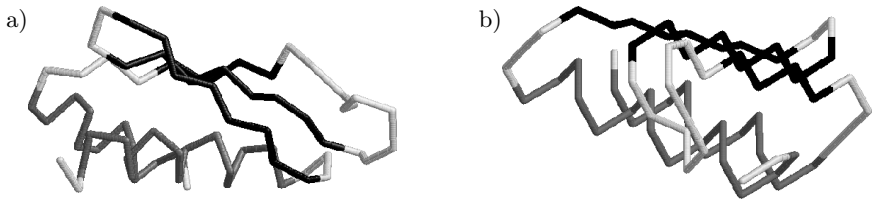

Fig. 2. "Maternal effect protein Staufen". a) backbone of the structure in the PDB (1STU, model 1) b) prediction of our algorithm

minutes, which we could not improve in 7 minutes of search. For the optimal placement of the secondary structure from the first search, we performed a second search for an energetically good tertiary structure placing the remaining amino-acids. The shown solution was found after 9m and could not be improved in 20m. Note that we find structures with only slightly lower energy after 62s of search. We applied the filtering described in Sub-section 7.2 using 30% and 20%.

Furthermore, we compared our approach to the one of [10]. We predicted structures for three proteins from the PDB, which were folded there also. We list PDB-code, number of residues, numbers of SSEs, and run-times for each protein. For our approach, we list the run-times of the three phases separately.

	length	number of SSEs		run-times of phases			run-times of [10]
		α-helices	β-sheets	1	2	3	
1VII	36	3	0	1.6s	3.1s	32s	6m56s
1E0M	37	0	3	0.3s	17s	1m42s	9m45s
2GP8	40	2	0	3.1s	60ms	1.8s	9m0s

Currently, we do not always find good solutions by applying our strategy. Possible reasons and improvements are discussed in the next section.

9 Conclusion and Future Work

We present a novel application of constraint programming to the protein structure prediction problem. The proposed approach combines the use of secondary structure annotation with a strategy that bases tertiary-structure predictions on energetically good placements of SSEs. As we demonstrate using examples from nature, this approach improves in effectivity and application range over recent constraint-based structure prediction algorithms.

However, our main goal in this work is to investigate a basic pattern of a constraint-based protein structure prediction algorithm that is based on secondary-structure information. We plan build on this work in order to improve both, efficiency of the structure prediction and accuracy in modeling proteins.

Regarding the efficiency of structure prediction, please note again, that this work focuses on the placement of SSEs. In consequence, the placement of the remaining amino-acids leaves room for further optimization. Nevertheless, the current strategy turned out to be effective in the discussed application range. A more sophisticated strategy becomes in particular important when investigating improvements in terms of accuracy.

For the aspect of accuracy, we consider it promising to investigate in particular four improvements: refinement of solutions by stochastic optimization, using more complex energy functions (e.g., Lennard-Jones potential), modeling the tertiary-structure off-lattice, and modeling sidechains of amino-acids.

Acknowledgments

The authors thank Federico Fogolari for several useful discussions. A. Dal Palù and A. Dovier are partially supported by MIUR Project *Verifica di sistemi reattivi basata su vincoli (COVER)* and by FSE project *Misura D4*.

References

1. R. Backofen. The protein structure prediction problem: A constraint optimization approach using a new lower bound. *Constraints*, 6(2–3):223–255, 2001.
2. R. Backofen and S. Will. Excluding symmetries in constraint-based search. *Constraints*, 7(3):333–349, 2002.
3. Z. Bagci, R. L. Jernigan, and I. Bahar. Residue coordination in proteins conforms to the closest packing of spheres. *Polymer*, 43:451–459, 2002.
4. Z. Bagci, R. L. Jernigan, and I. Bahar. Residue packing in proteins: Uniform distribution on a coarse-grained scale. *J Chem Phys*, 116:2269–2276, 2002.
5. H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. The protein data bank. *Nucleic Acids Research*, 28:235–242, 2000. `http://www.rcsb.org/pdb/`.
6. 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.
7. R. Bonneau and D. Baker. Ab initio protein structure prediction: progress and prospects. *Annu. Rev. Biophys. Biomol. Struct.*, 30:173–89, 2001.
8. B. Cipra. Packing challenge mastered at last. *Science*, 281:1267, 1998.
9. P. Crescenzi, D. Goldman, C. Papadimitrou, A. Piccolboni, and M. Yannakakis. On the complexity of protein folding. In *Proc. of STOC*, pages 597–603, 1998.
10. A. Dal Palù, A. Dovier, and F. Fogolari. Protein folding in $CLP(\mathcal{FD})$ with empirical contact energies. In *Recent Advances in Constraints*, volume 3010 of *Lecture Notes in Computer Science*, pages 250–265. Springer-Verlag, Berlin, 2004.
11. T. Müller and J. Würtz. Interfacing propagators with a concurrent constraint language. In *JICSLP96 Post-conference workshop and Compulog Net Meeting on Parallelism and Implementation Technology for (Constraint) Logic Programming Languages*, pages 195–206, 1996.
12. B. H. Park and M. Levitt. The complexity and accuracy of discrete state models of protein structure. *Journal of Molecular Biology*, 249(2):493–507, 1995.
13. B. Rost and C. Sander. Prediction of protein secondary structure at better than 70% accuracy. *Journal of Molecular Biology*, 232(2):584–99, 1993.
14. J. Skolnick and A. Kolinski. Computational studies of protein folding. *Computing in Science and Engineering*, 3(5):40–50, 2001.
15. G. Smolka. The Oz programming model. In *Computer Science Today: Recent Trends and Developments*. Springer-Verlag, Berlin, 1995.
16. S. Will. Constraint-based hydrophobic core construction for protein structure prediction in the face-centered-cubic lattice. In *Proceedings of the Pacific Symposium on Biocomputing 2002 (PSB 2002)*.