Fast, Constraint-based Threading of HP-Sequences to Hydrophobic Cores

Rolf Backofen and Sebastian Will*

Institut für Informatik, LMU München
Oettingenstraße 67, D-80538 München
{backofen,wills}@informatik.uni-muenchen.de

Abstract. Lattice protein models are used for hierarchical approaches to protein structure prediction, as well as for investigating principles of protein folding. So far, one has the problem that there exists no lattice that can model real protein conformations with good quality *and* for which an efficient method to find native conformations is known.
We present the first method for the FCC-HP-Model [3] that is capable of finding native conformations for real-sized HP-sequences. It has been shown [23] that the FCC lattice can model real protein conformations with coordinate root mean square deviation below 2 Å.
Our method uses a constraint-based approach. It works by first calculating maximally compact sets of points (hydrophobic cores), and then threading the given HP-sequence to the hydrophobic cores such that the core is occupied by H-monomers.

1 Introduction

The protein structure prediction is one of the most important unsolved problems of computational biology. It can be specified as follows: Given a protein by its sequence of amino acids (more generally monomers), what is its native structure? NP-completeness of the problem has been proven for many different models (including lattice and off-lattice models) [10, 12]. These results strongly suggest that the protein folding problem is NP-hard in general. Therefore, it is unlikely that a general, efficient algorithm for solving this problem can be given. Actually, the situation is even worse, since the general principles how natural proteins fold into a native structure are unknown. This is cumbersome since rational design is commonly viewed to be of paramount importance e.g. for drug design, where one faces the difficulty to design proteins that have a unique and stable native structure.

To tackle structure prediction and related problems, simplified models have been introduced. They are used in hierarchical approaches for protein folding (e.g., [29], see also the meeting review of CASP3 [18], where several groups have successfully used lattice models). Furthermore, they have became a major tool for investigating general properties of protein folding.

* Supported by the PhD programme "Graduiertenkolleg Logik in der Informatik" (GKLI) of the "Deutsche Forschungsgemeinschaft" (DFG).

Most important are the so-called lattice models. The simplifications commonly used in this class of models are: 1) monomers (or residues) are represented using a unified size 2) bond length is unified 3) the positions of the monomers are restricted to lattice positions and 4) a simplified energy function. Native conformations are those having minimal energy.

In the literature, many different lattice models (i.e., lattices and energy functions) have been used. Examples of how such models can be used for predicting the native structure or for investigating principles of protein folding are given in [28, 1, 15, 27, 17, 2, 20, 29]. Of course, the question arises which lattice and energy function have to be preferred. There are two (somewhat conflicting) aspects that have to be evaluated when choosing a model: 1) the accuracy of the lattice in approximating real protein conformations and the ability of the energy function to discriminate native from non-native conformations, and 2) the availability and quality of search algorithm for finding minimal (or nearly minimal) energy conformations.
While the first aspect is well-investigated in the literature (e.g., [23, 13]), the second aspect is underrepresented. By and large, there are mainly two different heuristic search approaches used in the literature: 1) Ad hoc restriction of the search space to compact or quasi-compact conformations (a good example is [28], where the search space is restricted to conformations forming an $n \times n \times n$-cube). The main drawback here is that the restriction to compact conformation is not motivated biologically for a complete amino acid sequence (as done in these approaches), but only for the hydrophobic amino acids. In consequence, the restriction either has to be relaxed and then leads to an inefficient algorithm or is chosen too strong and thus may exclude optimal conformations. 2) Stochastic sampling like Monte Carlo methods with simulated annealing, genetic algorithms etc. Here, the degree of (sub)optimality for the best conformations and the quality of the sampling cannot be determined by state of the art methods.[1]

In this paper, we follow the proposal by [3] to use a lattice model with a simple energy function, namely the HP (hydrophobic-polar) model (which has been introduced in [19] using cubic lattice), but on a better suited lattice, namely the face-centered cubic lattice (*FCC*). In the FCC, every point has 12 neighbors (instead of 6 as in the cubic lattice). The resulting model is called the *FCC-HP-model*. In the HP-model, the 20 letter alphabet of amino acids is reduced to a two letter alphabet, namely H and P. H represents *hydrophobic* amino acids, whereas P represent *polar* or hydrophilic amino acids. The energy function for the HP-model is given by the matrix as shown in Figure 1(a). It simply states that the energy contribution of a contact between two monomers is -1 if both are H-monomers, and 0 otherwise. Two monomers form a *contact* in some specific conformation if they are not connected via a bond, and occupied positions are nearest neighbors. A conformation with *minimal energy* (called

[1] Despite that there are mathematical treatments of Monte Carlo methods with simulated annealing, the partition function of the ensemble (which is needed for a precise statement) is in general unknown.

(a)

	H	P
H	-1	0
P	0	0

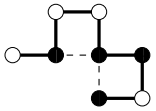
(b) 

Fig. 1. Energy matrix and sample conformation for the HP-model

native conformation) is just a conformation with the maximal number of contacts between H-monomers. Just recently, the structure prediction problem has been shown to be NP-complete even for the HP-model [10, 12].

A sample conformation for the sequence PHPPHHPH in the two-dimensional cubic lattice with energy -2 is shown in Figure 1(b). The white beads represent P, the black ones H monomers. The two contacts are indicated via dashed lines.

There are two reasons for using the FCC-HP-Model:
1) The FCC can model real protein conformations with good quality (see [23], where it was shown that FCC can model protein conformations with coordinate root mean square deviation below 2 Å)
2) The HP-model models the important aspect of hydrophobicity. Essentially, it is a polymer chain representation (on a lattice) with one stabilizing interaction each time two hydrophobic residues have unit distance. This enforces compactification while polar residues and solvent is not explicitly regarded. The idea of the model is the assumption that the hydrophobic effect determines the overall configuration of a protein (for a definition of the HP-model, see [19, 13]).

Once a search algorithm for minimal energy conformations is established for this FCC-HP-model, one can employ it as a filter step in a hierarchical approach. This way, one can improve the energy function to achieve better biological relevance and go on to resemble amino acid positions more accurately.

Related Work and Contribution In this paper, we describe a successful application of constraint-programming for finding native conformations in the FCC-HP-model. In this respect, the situation as given in the literature was not very promising. Although the FCC-HP-model is known to be an important lattice model, no exact algorithm was known for finding native conformations in any model different from the cubic lattice. Even for the cubic lattice, there are only three exact algorithms known [30, 4, 7], which are able to enumerate minimal (or nearly minimal) energy conformations, all for the cubic lattice. However, the ability of this lattice to approximate real protein conformations is poor. For example, [3] pointed out especially the parity problem in the cubic lattice. This drawback of the cubic lattice is that every two monomers with chain positions of the same parity cannot form a contact.

So far, beside heuristic approaches (e.g., the hydrophobic zipper [14], the genetic algorithm by Unger and Moult [26], the chain growth algorithm by Bornberg-Bauer [11], or [8], which is a method applicable for any regular lattice), there is only one approximation algorithm [3] for the FCC. It finds conformations whose number of contacts is guaranteed to be 60% of the number of contacts of the native conformation (which is far from being useful, since, even if the algorithm yields far better results, the information on the quality of the outcome is

still too fuzzy). The situation was even worse, since the main ingredient needed for an exact method, namely bounds on the number of HH-contacts given some partial information about the conformation, was missing. This changed with [5, 6], where such a bound is introduced and applied to the problem of finding maximally compact hydrophobic cores. Given a conformation of an HP-sequence, the *hydrophobic core* of this sequence is the set of all points occupied by H-monomers. A hydrophobic core of n-points is *maximally compact* if there is no packing of n-points in the FCC which has more contacts. In this paper, we show how we can efficiently thread a given HP-sequence to a maximally compact hydrophobic core[2]. We have implemented our method in the constraint language Oz [25] with extensions in C++.

2 Preliminaries

Given vectors $\boldsymbol{v_1}, \ldots, \boldsymbol{v_n}$, the lattice generated by $\boldsymbol{v_1}, \ldots, \boldsymbol{v_n}$ is the minimal set of points L such that $\forall \boldsymbol{u}, \boldsymbol{v} \in L$, both $\boldsymbol{u}+\boldsymbol{v} \in L$ and $\boldsymbol{u}-\boldsymbol{v} \in L$. The *face-centered cubic lattice* (FCC) is defined as the lattice

$$D_3 = \{\begin{pmatrix} x \\ y \\ z \end{pmatrix} \mid \begin{pmatrix} x \\ y \\ z \end{pmatrix} \in \mathbb{Z}^3 \text{ and } x+y+z \text{ is even}\}.$$

We use $\uplus$ to denote disjoint union. The set N_{D_3} of *minimal vectors* connecting so-called *neighbors* in D_3 is given by

$$N_{D_3} = \left\{\begin{pmatrix} 0 \\ \pm 1 \\ \pm 1 \end{pmatrix}\right\} \uplus \left\{\begin{pmatrix} \pm 1 \\ 0 \\ \pm 1 \end{pmatrix}\right\} \uplus \left\{\begin{pmatrix} \pm 1 \\ \pm 1 \\ 0 \end{pmatrix}\right\}.$$

Thus, every point in the FCC has 12 neighbors. A *hydrophobic core* is is a function $f : D_3 \to \{0,1\}$, where $f^{-1}(1) \neq \emptyset$. The purpose of a hydrophobic core is to characterize the set of positions occupied by H-monomers. We will identify a hydrophobic core f with the set of all *points occupied by* f, i.e. $\{\boldsymbol{p} \mid f(\boldsymbol{p}) = 1\}$. Hence, for hydrophobic cores f_1, f_2 we will use standard set notation for size $|f_1|$, union $f_1 \cup f_2$, disjoint union $f_1 \uplus f_2$, and intersection $f_1 \cap f_2$. Given a hydrophobic core f, we define the *number of contacts of* f by $\mathrm{con}(f) := \frac{1}{2}\,|\{(\boldsymbol{p}, \boldsymbol{p'}) \mid f(\boldsymbol{p}) \wedge f(\boldsymbol{p'}) \wedge (\boldsymbol{p} - \boldsymbol{p'}) \in N_{D_3}\}|$. A hydrophobic core f is *maximally compact* if $\mathrm{con}(f) = \max\{\mathrm{con}(f') \mid |f| = |f'|\}$.

An *HP-sequence* is an element in $\{H, P\}^*$. With s_i we denote the ith element of a sequence s. A *conformation* c of a sequence s is a function $c : \{1, \ldots, |s|\} \to D_3$ such that 1) $\forall 1 \leq i < |s| : c(i) - c(i+1) \in N_{D_3}$, and 2) $\forall i \neq j : c(i) \neq c(j)$. The hydrophobic core associated with a conformation c is defined as the set of positions occupied by an H-monomer in c. The *number of contacts* $\mathrm{con}(c)$ of a conformation c is defined to be $\mathrm{con}(f)$, where f is the hydrophobic core associated with c. A conformation c is called *native for* s if it has maximal number of contacts.

A *finite CSP (constraint satisfaction problem)* $P = (\mathcal{X}, \mathcal{D}, \mathcal{C})$ is defined by

[2] Of course, the methods described in this paper can also be applied to hydrophobic cores that are not maximally compact.

– a set of *variables* $\mathcal{X}$,
– a set of finite domains $\mathcal{D}$, where the domain of $x \in \mathcal{X}$ is $\mathrm{dom}(x) \in \mathcal{D}$.
– a set of constraints $\mathcal{C}$ between the variables.

A constraint C on the tuple $X(C) = (x_1, \ldots, x_n)$ of variables is interpreted as a subset $\mathrm{T}(C)$ of the Cartesian product $\mathrm{dom}(x_1) \times \cdots \times \mathrm{dom}(x_n)$ which specifies allowed combinations of values for the variables. A constraint C, where $X(C) = (x_1, \ldots, x_n)$, is called *n-ary*.

$a \in \mathrm{dom}(x)$ is *consistent* with a constraint C, if either $x \notin X(C)$ or x is the $i-th$ variable of C and $\exists \tau \in \mathrm{T}(C) : a = \tau_i$. A constraint C is *(hyper-)arc consistent* iff for all $x_i \in X(C)$, $\mathrm{dom}(x_i) \neq \emptyset$ and for all $a \in \mathrm{dom}(x_i)$ holds a is consistent with C.

2.1 Enumerating Hydrophobic Cores

We are interested in maximally compact hydrophobic cores, since a conformation with a maximally compact hydrophobic core is already native.[3] We recall the main principles for calculating maximally compact hydrophobic cores as described in [5, 6].

To determine maximally compact hydrophobic cores, we partition a hydrophobic core f into cores $f_1, \ldots, f_k$ of the layers $x = 1, \ldots, x = k$. For searching a maximal hydrophobic core f, we do a branch-and-bound search on k and $f_1 \ldots f_k$. Of course, the problem is to give good bounds that allow us to cut off many k and $f_1 \ldots f_k$ that will not maximize $\mathrm{con}(f_1 \uplus \ldots \uplus f_k)$. For this purpose, we distinguish between contacts in a single layer ($= \mathrm{con}(f_i)$ for $1 \leq i \leq k$), and interlayer contacts $\mathrm{IC}_{f_i}^{f_{i+1}}$ for $1 \leq i < k$ between two successive layers. Interlayer contacts are pairs $(\boldsymbol{p}, \boldsymbol{p'})$ such that $\boldsymbol{p}$ and $\boldsymbol{p'}$ are neighbors, $\boldsymbol{p} \in f_i$ and $\boldsymbol{p'} \in f_{i+1}$. The hard part is to bound the number of contacts between two successive layers, since a simple but tight bound for the number of (intra)layer contacts can be taken from the literature [30].

For defining the bound on the number of contacts between two successive layers, we introduce the notion of an i-point, where $i = 1, 2, 3, 4$. Any point in $x = c+1$ can have at most 4 neighbors in the plane $x = c$. Let f be a hydrophobic core of the plane $x = c$. Call a point $\boldsymbol{p}$ in plane $x = c + 1$ an *i-point for f* if it has i neighbors in plane $x = c$ that are contained in f (where $i \leq 4$). Of course, if one occupies an i-point in plane $x = c+1$, then this point generates i contacts between layer $x = c$ and $x = c + 1$. In the following, we will restrict ourself to the case where $c = 1$ for simplicity. Of course, the calculation is independent of the choice of c.

Consider as an example the two hydrophobic cores f_1 of plane $x = 1$ and f_2 of plane $x = 2$ as shown in Figure ??. f_1 contains 5 points, and f_2 contains 3 points. Since f_2 contains one 4-point, one 3-point and one 2-point of f_1, there are 9 contacts between these two layers. It is easy to see that we generate the most contacts between layers $x = 1$ and $x = 2$ by first occupying the 4-points,

[3] Of course, there can be the rare case that there is a native conformation whose hydrophobic core is *not* maximally compact.

then the 3 points and so on until we reach the number of points to be occupied in layer $x = 2$.[4]

For this reason, we are interested in calculating the maximal number of i-points (for $i = 1, 2, 3, 4$), given only the number of colored points n in layer $x = 1$. But this would overestimate the number of possible contacts, since we would maximize the number of 4-, 3-, 2- and 1- points independently from each other. We have found a dependency between these numbers, which requires to fix the side length (a, b) of the minimal rectangle around all colored points in layer $x = 1$ (called the *frame*). In our example, the frame is $(3, 2)$.

Denote with $max_i(n, a, b)$ the maximal number of i-points in layer $x = 2$ for any hydrophobic core of layer $x = 1$ with n points and frame (a, b). Then we have found that

$$max_4(n, a, b) = n + 1 - a - b \qquad max_2(n, a, b) = 2a + 2b - 2\ell - 4$$
$$max_3(n, a, b) = \ell \qquad max_1(n, a, b) = \ell + 4.$$

The remaining part is to find $\ell = max_3(n, a, b)$, which is described in detail in [5, 6]. This calculation involves several special cases to treat layers that are not sufficiently filled with H-monomers. Using these $max_i(n, a, b)$, we can define a bound

$$B^{n_{i+1}}_{n_i,a_i,b_i} \geq \max \left\{ \mathrm{IC}^{f_{i+1}}_{f_i} \middle| \begin{array}{c} \#f_i = n_i,\ f_i \text{ has frame } (a_i, b_i), \\ \text{and } \#f_{i+1} = n_{i+1}, \end{array} \right\}$$

where $1 \leq i \leq k-1$ and $\#X$ denotes the cardinality of a set X. This bound can be calculated in polynomial time using dynamic programming [5, 6].

This bound is used in searching for a maximally compact core for n H-monomers as follows. Instead of directly enumerating k and all possible cores $f_1 \uplus \ldots \uplus f_k$, we search through all possible sequences $((n_1, a_1, b_1) \ldots (n_k, a_k, b_k))$ of parameters with the property that $n = \sum_i n_i$. By using the $B^{n_{i+1}}_{n_i,a_i,b_i}$, only a few layer sequences have to be considered further. For these optimal layer sequences, we search for all admissible cores $f_1 \uplus \ldots \uplus f_k$ using again a constraint-based approach. Our implementation is able to find maximally hydrophobic cores for n upto 100 within seconds.

3 Threading an HP-sequence to a Hydrophobic Core

3.1 Problem Description and Modeling

Since we are able to determine maximally hydrophobic cores, it remains to thread an HP-sequence to such an optimal core in order to get HP-optimally folded structures for the sequence. We tackle the problem by a constraint based approach.

For this reason, let a hydrophobic core be given as a set of lattice points $\mathbf{C}$. The sequence is given as a word s in $\{\mathrm{H}, \mathrm{P}\}^*$. For correct input, the size of $\mathbf{C}$ equals the number of H's occurrences in the sequence.

[4] Note that this strategy might not result necessarily in the coloring with the maximal number of contacts, since we might loose contacts within the layer $x = 2$.

The protein structure is modeled by a set of variables $x_1, \ldots, x_{|s|}$, whose finite domains are sets of lattice points, resp. more generally nodes of a graph, where a graph G is a tuple (V, E) of the finite set of nodes V and the set of edges $E \subseteq V \times V$. The problem is now to find a solution, i.e. an assignment of the monomers to nodes, satisfying the following constraints

1. the nodes x_i, where $s_i = \mathrm{H}$ and $1 \leq i \leq |s|$, are elements of $\mathbf{C}$.
2. all the x_i, where $1 \leq i \leq |s|$, are different
3. the nodes $x_1, \ldots, x_{|s|}$ form a path

Note that for correct input, the first constraint implies that P monomers are not in the core. However, due to the finite chain length we can determine finite domains for the P-representing variables. The second constraint tells that a protein structure has to be self avoiding. Finally, the last constraint tells that chain bonds between monomers are to be preserved in a protein structure, such that the monomer positions form a path through the lattice.

Some attention has to be paid to the fact that many constraint systems do only support integer finite domain variables, whereas in our formulation domains are lattice nodes. Since depending on the input only a finite set of nodes can be assigned in solutions, we straightforwardly solve this by assigning unique integers to these nodes.

3.2 Path Constraints

The treatment of the first constraint of the preceeding section involves the computation of domains and the assignment of domains to the variables. Both of the remaining constraints can be handled globally. The global treatment of the so called all-different constraint is well described in [24]. Thus, we will focus on the treatment of the path constraint. We will further discuss how one gets further propagation by combining the two constraints.

For generality, we discuss the constraints on arbitrary finite graphs. Clearly, we can use the results for the FCC lattice afterwards. There, the set of graph nodes is a subset of the lattice nodes and the edges are all pairs of graph nodes in minimal lattice distance.

In the following, we fix a graph $G = (V, E)$. A *path of length* n is a word $p = p_1 \ldots p_n$ of length n of alphabet V, such that

$$\forall 1 \leq i \leq n-1 : (p_i, p_{i+1}) \in E.$$

Denote the set of paths of length n by $\mathrm{paths}(n)$. Note that intentionally paths are allowed to contain cycles due to the definition.

We define a path constraint to state that the nodes assigned to the argument variables form a path.

Definition 1 (Path Constraint). *Let $x_1, \ldots, x_n$ be variables. We call a path $p \in \mathrm{paths}(n)$* consistent for $x_1, \ldots, x_n$, *iff $\forall 1 \leq i \leq n : p_i \in \mathrm{dom}(x_i)$ holds.*

The path constraint $C = \mathrm{Path}(x_1, \ldots, x_n)$ *is defined by the tuples*

$$\mathrm{T}(C) = \{p \in \mathrm{paths}(n) | p \textit{ is consistent for } x_1, \ldots, x_n\}.$$

Hyper-arc consistency of this path constraint is a local property in the following sense. By a general result of Freuder [16], arc consistency amounts to global consistency in a tree-structured network of binary constraints. The next lemma is an instance of this result.

Lemma 1. *Let $x_1, \ldots, x_n$ be variables.* $\mathrm{Path}(x_1, \ldots, x_n)$ *is hyper-arc consistent, iff for* $1 \leq i \leq n-1$ *all constraints* $\mathrm{Path}(x_i, x_{i+1})$ *are arc consistent.*

Due to this lemma, the hyper-arc consistency of the n-ary path constraint is reduced to the arc consistency of the set of all 2-ary path constraints.

3.3 Combining path and all-different constraint

The combination of the path constraint with the all-different constraint yields a new constraint which allows only self avoiding paths. Formally, let $x_1, \ldots, x_n$ be variables, define the *all-different constraint* $C = \mathrm{AllDiff}(x_1, \ldots, x_n)$ by

$$\mathrm{T}(C) = \left\{ (\tau_1, \ldots, \tau_n) \in \mathrm{dom}(x_1) \times \cdots \times \mathrm{dom}(x_n) \middle| \forall 1 \leq i < j \leq n : \tau_i \neq \tau_j \right\}.$$

We define the *self avoiding path constraint* $\mathrm{SAPath}(x_1, \ldots, x_n)$ by

$$\mathrm{T}(\mathrm{SAPath}(x_1, \ldots, x_n)) = \mathrm{T}(\mathrm{AllDiff}(x_1, \ldots, x_n)) \cap \mathrm{T}(\mathrm{Path}(x_1, \ldots, x_n)).$$

Unfortunately, we are not aware of any efficient arc consistency algorithm for this combined constraint in the literature. Furthermore, it is unlikely that there exists one. It is well known that many problems involving self-avoiding walks (we use the term path here), especially counting of such walks, are intrinsically hard and there are no efficient algorithms to solve them [21].

On the other hand, the treatment of self avoiding paths promises much better propagation in practice. Therefore, we propose a relaxation of the intractable self-avoiding path arc consistency in the following. An efficiently tractable relaxation one may think of first, is to constrain the paths to be non-reversing. Non-reversing paths are paths which do not turn back immediately, hence their class lies between general paths and self-avoiding paths. Here, we choose a more general approach and define the following sets of paths.

Definition 2. *Let $1 \leq k \leq n$. A* k-avoiding path $p = p_1 \ldots p_n$ of length n *is a path $p \in \mathrm{paths}(n)$, where for all $1 \leq i \leq n-k+1$, the $p_i \ldots p_{i+k-1}$ are all different. We define that for $k > n$, k-avoiding is equivalent to n-avoiding. Denote the set of k-avoiding paths of length n by* $\mathrm{paths}[k](n)$.

Note that obviously, general paths (resp. self-avoiding paths) of length n are special cases of k-avoiding paths namely 1-avoiding paths (resp. n-avoiding paths) of length n. For graphs with symmetric and non-reflexive edges, the property non-reversing is equivalent to 3-avoiding. Obviously by definition, $\mathrm{paths}[k'](n) \subseteq \mathrm{paths}[k](n)$ holds for all $1 \leq k \leq k' \leq n$.

Let $x_1, \ldots, x_n$ be variables. Define the *set of k-avoiding paths consistent with* $x_1, \ldots, x_n$ as $\mathrm{cpaths}[k](x_1, \ldots, x_n)$. We define corresponding constraints, which

constrain their variables to form k-avoiding paths. Define the *k-avoiding path constraint* $\mathrm{Path}[k](x_1,\ldots,x_n)$ by

$$\mathrm{T}(\mathrm{Path}[k](x_1,\ldots,x_n)) = \mathrm{cpaths}[k](x_1,\ldots,x_n).$$

Analogously to the general path constraint the k-avoiding path constraints possess locality, i.e. we get arc consistency of an n-ary k-avoiding path constraint by the arc consistency of the k-ary k-avoiding path constraints on every length k subsequences of variables.

Since the k-ary constraints have to be computed independently by searching for self-avoiding paths, the reduction to local arc consistency leads to unnecessary inefficiency. To avoid this, we propose a global algorithm in the following. This will be rewarded by even stronger propagation possibilities.

The key to our algorithm is the counting of paths. For arc consistency, we need to know, whenever there is no path left, where a i-th monomer is positioned on a node v. It is a good starting point to count the number of all (consistent) k-avoiding paths.

Denote the cardinality of a set X by $\#X$. For computing the number of paths $\#\,\mathrm{cpaths}[k](x_1,\ldots,x_n)$, we will first define the *set of k-avoiding paths consistent with $x = x_1,\ldots,x_n$ with suffix (path) $q = q_1\ldots q_m$* for $n \geq m$ as

$$\mathrm{scpaths}[k](x)[q] = \left\{\, p \in \mathrm{cpaths}[k](x) \,\middle|\, \forall 1 \leq i \leq m : p_{n-m+i} = q_i \,\right\}$$

To resemble an efficient implementation more closely, we define $\mathrm{sp}[k+1](x)[q]$ analogous to $\mathrm{scpaths}[k](x)[q]$ with the only difference that $\mathrm{sp}[k+1](x)[q]$ is only defined when q is consistent with $x_{n-k+1},\ldots,x_n$. Note that for all practical purposes, we will consider only $\mathrm{scpaths}[k](x)[q]$ where $|q| = k-1$. The idea is that one has to remember a suffix (or later a prefix) of length $k-1$ in order to check k-avoiding.

Lemma 2. *Let $x = x_1,\ldots,x_n$ be variables, $0 < k \leq n$.*

The number of paths $\#\,\mathrm{cpaths}[k+1](x)$ is equal to the sum

$$\sum_{q \in \mathrm{paths}(k)} \#\,\mathrm{scpaths}[k+1](x)[q].$$

For $q = q_1\ldots q_k \in \mathrm{paths}(k)$, the following number of paths can be computed recursively.

$$\#\,\mathrm{scpaths}[k+1](x)[q] = \begin{cases} \#\,\mathrm{sp}[k+1](x)[q] & q \in \mathrm{cpaths}[k](x_{n-k+1},\ldots,x_n) \\ 0 & \textit{otherwise,} \end{cases}$$

where for $q \in \mathrm{cpaths}[k](x_{n-k+1},\ldots,x_n)$,

$$\#\,\mathrm{sp}[k+1](x)[q] = \begin{cases} 1 & n = k \\ \displaystyle\sum_{\substack{(q_0,q_1)\in E,\\ q_0 \notin \{q_1,\ldots,q_k\},\\ q_0 \in \mathrm{dom}(x_{n-k})}} \#\,\mathrm{sp}[k+1](x_1,\ldots,x_{n-1})[q_0\ldots q_{k-1}] & n > k. \end{cases}$$

Clearly, the numbers of paths with suffixes can be computed efficiently by a dynamic programming algorithm furnished by the recursive definition.

This algorithm to compute the numbers of k-avoiding paths of maximal length n, where $2 \leq k \leq n$, has a polynomial complexity in n and the number of nodes $|V|$.

Note that the lemma handles only the case of k-avoiding paths, where $k \geq 2$. The reason is that for the path property itself we have to remember a history of minimal length 1. Hence, the number of 1-avoiding paths can not be computed more efficiently than the number of 2-avoiding paths. Obviously the lemma could be slightly modified (by dropping the condition $q_0 \notin \{q_1, \ldots, q_k\}$ in the sum of the recursion step) to compute the number of 1-avoiding, i.e. general paths.

Analogously to paths with suffixes, we can treat paths with prefixes. Hence, define the *set of k-avoiding paths consistent with $x = x_1, \ldots, x_n$ with prefix $q = q_1 \ldots q_m$* as

$$\mathrm{pcpaths}[k][q](x) = \left\{ p \in \mathrm{cpaths}[k](x) \middle| \forall 1 \leq i \leq \min(m,n) : p_i = q_i \right\}.$$

It is easy to see (by symmetry), that the paths with prefixes can be treated analogously to paths with suffixes.

We can now express the number of k-avoiding paths consistent with $x = x_1, \ldots, x_n$, where the i-th monomer occupies the position v, in terms of suffix and prefix path numbers.

For preparation, define the set of these paths as $\mathrm{cpaths}[k](x|i \mapsto v)$. In the case of usual paths, the number of walks that map x_i to position v is the number of prefixes of length i that end in v times the number of suffixes of length $n-i$ starting in v. For k-avoiding paths, this does not suffice, since the composition of a k-avoiding prefix and suffix will not generate a k-avoiding path in general. To guarantee this, the prefix and suffix has to overlap at least by $k-1$ positions. Note that the i can be located arbitrarily in this overlapping region. These considerations are summarized by the next lemma.

Lemma 3. *Let $x = x_1, \ldots, x_n$ be variables, $1 \leq i \leq n$, and $v \in V$. Let j be such that $1 \leq k+1 \leq n$, $1 \leq j \leq i \leq j+k-1 \leq n$.*

$$\#\,\mathrm{cpaths}[k+1](x|i \mapsto v) = \sum_{\substack{q \in \mathrm{paths}[k](k) \\ q_{i-j+1} = v}} \begin{pmatrix} \#\,\mathrm{scpaths}[k+1](x_1, \ldots, x_{j+k-1})[q] \cdot \\ \#\,\mathrm{pcpaths}[k+1][q](x_j, \ldots, x_n) \end{pmatrix}.$$

Based on the computation of these numbers we develop an arc consistency algorithm for the k-avoiding path constraints.

Theorem 1. *Let $x = x_1, \ldots, x_n$ be variables with non-empty domains. The constraint $C = \mathrm{Path}[k](x)$ is arc consistent, iff for every $1 \leq i \leq n$ and $v \in V$, where $\#\,\mathrm{cpaths}[k](x|i \mapsto v) = 0$, it holds that $v \notin \mathrm{dom}(x_i)$.*

Proof. Let x and C be defined as in the theorem.

First, let C be arc consistent. Let $1 \leq i \leq n$ and $v \in V$, such that the set $\mathrm{cpaths}[k](x|i \mapsto v)$ is empty. Then, there is no path $p \in \mathrm{paths}(k)x$, where $p_i = v$. Hence there is no such path in $\mathrm{T}(C)$. We get $v \notin \mathrm{dom}(x_i)$, due to the arc consistency of C.

Second, let C be not arc consistent. We show that there is a $1 \leq i \leq n$ and $v \in V$, such that $v \in \mathrm{dom}(x_i)$ and $\#\,\mathrm{cpaths}[k](x|i \mapsto v) = 0$. The arc consistency of C has to be violated by at least one pair $1 \leq i \leq n$ and $v \in V$, where $v \in \mathrm{dom}(x_i)$. Choose such i and v. Since consequently there is no path in $\mathrm{T}(C)$, where $p_i = k$, there is no such path in $\mathrm{cpaths}[k](x)$. This implies $\mathrm{cpaths}[k](x|i \mapsto v) = \emptyset$.

Assume that the variables in a set X are constrained as all different. If we can derive, that in every solution one of the variables in $Y \subseteq X$ is assigned to a node v, we may introduce the basic constraints $v \notin \mathrm{dom}(x)$ for all $x \in X - Y$. The following theorem tells how to derive this.

Theorem 2. *Let $x = x_1, \ldots, x_n$ be variables, $1 \leq k \leq n$, and $\tau \in \mathrm{T}(\mathrm{Path}[k](x))$. Further, $S \subseteq \{1, \ldots, n\}$, such that $\max S - \min S \leq k$, and $v \in V$.*

Then, $\sum_{j \in S} \#\,\mathrm{cpaths}[k](x|j \mapsto v) = \#\,\mathrm{cpaths}[k](x)$ implies that $\tau_j = v$ for exactly one $j \in S$.

Proof. Let n, x, k, τ, S, and v be defined as in the theorem.

Let $j \in S$ and $p \in \mathrm{cpaths}[k](y|j \mapsto v)$. Since $\max S - \min S \leq k$, we know that $p_{j'} = v$ if and only if $j = j'$ for all $j' \in S$. Hence, the sets $\mathrm{cpaths}[k](y|j \mapsto v)$ are disjoint for $j \in S$. Thus, $\sum_{j \in S} \#\,\mathrm{cpaths}[k](y|j \mapsto v) = \#\,\mathrm{cpaths}[k](y)$ implies $\biguplus_{j \in S} \mathrm{cpaths}[k](y|j \mapsto v) = \mathrm{cpaths}[k](y)$, i.e., for every path $p \in \mathrm{cpaths}[k](y)$, $p_j = v$ for exactly one $j \in S$.

Finally, since $\tau_r \ldots \tau_{r+m-1} \in \mathrm{cpaths}[k](y)$, we get $\tau_j = v$ for exactly one $j \in S$.

In the following, we discuss in more detail how to avoid unnecessary large values for k, since the consistency and propagation algorithms are due to our recursion equations still exponential in k.

For $s, t \in V$, define a *path from s to t* as a path $p = p_1 \ldots p_n$, where $p_1 = s$ and $p_n = t$. Further, define a distance on nodes by

$$\mathrm{dist}(s,t) = \min \left\{ n > 0 \middle| p \in \mathrm{paths}(n), s = p_1, p_n = t \right\}.$$

Since V is finite, the defined distance can be computed by Dijkstra's shortest path algorithm. Note that $\mathrm{dist}(s,t)$ is neither a metric nor total.

Depending on the distance of first and last nodes of a path, k-avoidingness might be already guaranteed by k'-avoidingness for $k' < k$. This is stated by the next theorem.

Theorem 3. *Let $s, t \in V$, such that $d = \mathrm{dist}(s,t)$ is defined. Let $n > 0$, $1 \leq k', k \leq n$, such that $d + k' - n = n - k$. For every path $p \in \mathrm{paths}[k'](n)$ from s to t, it holds $p \in \mathrm{paths}[k](n)$.*

Proof. Fix $s, t \in V$, such that $d = \mathrm{dist}(s,t)$ is defined. Let $1 \leq k' \leq k \leq n$, where $d + k' - n = n - k$. Let $p \in \mathrm{paths}[k'](n)$ be a path from s to t. Assume $p \notin \mathrm{paths}[k](n)$. Then exists $1 \leq i \leq j \leq n$, where $j - i > k$ and $p_i = p_j$. Then, $p_1 \ldots p_i p_j + 1 \ldots p_n$ is a path of length $n - (j - i)$ from s to t. Now, by the minimality of d, $n - (j - i) \geq d$ holds. This implies $n - k > d$. By assumption $k = 2n - d - k'$. Hence, $n - (2n - d - k') > d$ and thus $k' - n > 0$ in contradiction to $k' \leq n$.

In a constraint search, the theorem allows to replace k-avoiding path constraints by more efficiently computed, but semantically equivalent k'-avoiding path constraints, whenever the conditions of the theorem are derived. Inversely, if we derive that k'-avoiding paths are in fact k-avoiding this allows stronger propagation due to theorem 2.

3.4 A Propagator for the Path Constraint

Based on the considerations of the previous subsections we sketch an implementation of the k-avoiding path constraint propagator.

Let $x = x_1, \ldots, x_n$ be finite domain variables. The general strategy of the propagator for $\mathrm{Path}[k](x)$ is as follows

1. For all $q \in \mathrm{paths}[k](k)$ and $k \leq i \leq n$, compute $\#\,\mathrm{scpaths}[k](x_1, \ldots, x_i)[q]$ and $\#\,\mathrm{pcpaths}[k][q](x_{n-i+1}, \ldots, x_n)$.
2. Compute from this the numbers $\#\,\mathrm{cpaths}[k](x|i \mapsto v)$ for all $1 \leq i \leq n$ and $v \in V$. Whenever such a value is 0, remove v from the domain of x_i.
3. If at least one domain of the $x_1, \ldots, x_n$ changes repeat from step 1.

Even since we have presented efficient algorithms to compute the above numbers and thus get arc consistency of the path constraint, there are some remaining problems. Most demanding are incremental computation and the saving of copying time.

At the first invocation, the computation of the path numbers can be done by dynamic programming algorithms. If domains are narrowed, the previously computed path numbers can be updated. For this aim, there exists an efficient update algorithm, which works destructively on the data structures. However, the incremental computation comes at the price of copying the data structures, whenever the tree branches.

Since for our purpose, the k-avoiding path propagator always works in presence of an all-different constraints, the k-avoiding path propagator should be able to handle further propagation due to the combination with this constraint. The justification to do this is given by Theorem 2. We use that for the arc consistency of a k-avoiding path constraint, the numbers $\#\,\mathrm{cpaths}[k'](x|i \mapsto v)$ are already computed for all $k' \leq k$. For tractability one has to restrict the subsets S, e.g. to all subsets of successive numbers up to size k.

Finally, one can simplify a k-avoiding path propagator by a more efficient k'-avoiding one, in situation described by Theorem 3, while preserving semantical equivalence.

3.5 Results

Exact structure prediction in the HP-model on the cubic lattice was previously possible up to chain lengths of 88 [30]. Yue and Dill report to find a native conformation for those chains in times ranging from minutes to hours. Our own algorithm for exact structure prediction on the cubic lattice regularly folds proteins with a length of $30 - 40$ monomers [4, 7]. Note that structure prediction in the cubic lattice is not necessarily easier for inexact, heuristic methods. For example, in [9] a heuristic stochastic approach is reported to fail on all but one of the investigated 48-mers.

We implemented two threading algorithms. For the first algorithm, we implemented a propagator to handle general paths by reduction to binary path constraint propagators. For the second algorithm, an experimental, non-optimized version of a propagator for 3-avoiding pathes is implemented. The propagators are implemented as extension to Mozart (Oz 3) [25]. Mozart provides a convenient interface for extension by C++-constraint-propagators [22].

For benchmarking of the two threading algorithms, the following experiment was performed. Random HP-sequences were threaded to cores of sizes n =25, 50, and 75. Therefore, for each core 50 sequences were randomly generated with n H-monomers and $0.8 \cdot n$ P-monomers, which is a rather high ratio of P-monomers to H-monomers and is chosen to challenge the algorithm. Additionally, we threaded 50 random sequences of length 160 to a core of size 100. We also managed to thread some random sequences of length 180 to this core. For each sequence, the threading is performed by both algorithms.

Both algorithms thread the very majority of the test sequences successfully. The results show that the combination of the path constraint with the all-different constraint yields significantly better propagation even for the strong relaxation of only 3-avoiding paths. Both algorithms successfully threaded all of the 50 sequences to the core of size 25 (which means a sequence length of 45). For longer sequences, the second algorithm succeeds for significantly more sequences than the first one. Furthermore, it often finds a solution in less search nodes (up to a factor of 303). The results are summarized in Table 1.

core size	seq. length	fails alg. 1	fails alg. 2	avg. nodes alg. 1	avg. nodes alg. 2
25	45	0%	0%	36	36
50	90	12%	2%	970	103
75	135	20%	8%	586	513
100	160	60%	50%	1468	598

Table 1. Threading of random sequences to cores of four different sizes. The table shows size of the core, the length of the sequences, the percentage of sequences which could not be threaded successfully within the given time limit by the two algorithms, and the average number of nodes in successfull runs by both algorithms. We choose a time limit of 5 minutes for the first algorithm. The second algorithm is given a longer time limit of 15 minutes, since the path propagator is experimental and non-optimized.

References

1. V. I. Abkevich, A. M. Gutin, and E. I. Shakhnovich. Impact of local and nonlocal interactions on thermodynamics and kinetics of protein folding. *Journal of Molecular Biology*, 252:460–471, 1995.
2. V.I. Abkevich, A.M. Gutin, and E.I. Shakhnovich. Computer simulations of prebiotic evolution. In Russ B. Altman, A. Keith Dunker, Lawrence Hunter, and Teri E. Klein, editors, *PSB'97*, pages 27–38, 1997.
3. Richa Agarwala, Serafim Batzoglou, Vlado Dancik, Scott E. Decatur, Martin Farach, Sridhar Hannenhalli, S. Muthukrishnan, and Steven Skiena. Local rules for protein folding on a triangular lattice and generalized hydrophobicity in the HP-model. *Journal of Computational Biology*, 4(2):275–296, 1997.
4. Rolf Backofen. Constraint techniques for solving the protein structure prediction problem. In Michael Maher and Jean-Francois Puget, editors, *Proceedings of* 4^{th} *International Conference on Principle and Practice of Constraint Programming (CP'98)*, volume 1520 of *Lecture Notes in Computer Science*, pages 72–86. Springer Verlag, 1998.
5. Rolf Backofen. An upper bound for number of contacts in the HP-model on the face-centered-cubic lattice (FCC). In Raffaele Giancarlo and David Sankoff, editors, *Proc. of the 11th Annual Symposium on Combinatorial Pattern Matching (CPM2000)*, volume 1848 of *Lecture Notes in Computer Science*, pages 277–292, Berlin, 2000. Springer–Verlag.
6. Rolf Backofen and Sebastian Will. Optimally compact finite sphere packings — hydrophobic cores in the FCC. In Amihood Amir and Gad Landau, editors, *Proc. of the 12th Annual Symposium on Combinatorial Pattern Matching (CPM2001)*, volume 2089 of *Lecture Notes in Computer Science*, pages 257–271, Berlin, 2001. Springer–Verlag.
7. Rolf Backofen, Sebastian Will, and Erich Bornberg-Bauer. Application of constraint programming techniques for structure prediction of lattice proteins with extended alphabets. *J. Bioinformatics*, 15(3):234–242, 1999.
8. Rolf Backofen, Sebastian Will, and Peter Clote. Algorithmic approach to quantifying the hydrophobic force contribution in protein folding. In Russ B. Altman, A. Keith Dunker, Lawrence Hunter, and Teri E. Klein, editors, *Pacific Symposium on Biocomputing (PSB 2000)*, volume 5, pages 92–103, 2000.
9. U Bastolla, H Frauenkron, E Gerstner, P Grassberger, and W Nadler. Testing a new monte carlo algorithm for protein folding. *Proteins*, 32(1):52–66, 1998.
10. B. Berger and T. Leighton. Protein folding in the hydrophobic-hydrophilic (HP) modell is NP-complete. In *Proc. of the Second Annual International Conferences on Compututational Molecular Biology (RECOMB98)*, pages 30–39, New York, 1998.
11. Erich Bornberg-Bauer. Chain growth algorithms for HP-type lattice proteins. In *Proc. of the* 1^{st} *Annual International Conference on Computational Molecular Biology (RECOMB)*, pages 47 – 55. ACM Press, 1997.
12. P. Crescenzi, D. Goldman, C. Papadimitriou, A. Piccolboni, and M. Yannakakis. On the complexity of protein folding. In *Proc. of STOC*, pages 597–603, 1998. Short version in *Proc. of RECOMB'98*, pages 61–62.
13. K.A. Dill, S. Bromberg, K. Yue, K.M. Fiebig, D.P. Yee, P.D. Thomas, and H.S. Chan. Principles of protein folding – a perspective of simple exact models. *Protein Science*, 4:561–602, 1995.

14. Ken A. Dill, Klaus M. Fiebig, and Hue Sun Chan. Cooperativity in protein-folding kinetics. *Proc. Natl. Acad. Sci. USA*, 90:1942 – 1946, 1993.
15. Aaron R. Dinner, Andreaj Šali, and Martin Karplus. The folding mechanism of larger model proteins: Role of native structure. *Proc. Natl. Acad. Sci. USA*, 93:8356–8361, 1996.
16. Eugene C. Freuder. A sufficient condition for backtrack-free search. *Journal of the Association for Computing Machinery*, 29(1):24–32, 1982.
17. S. Govindarajan and R. A. Goldstein. The foldability landscape of model proteins. *Biopolymers*, 42(4):427–438, 1997.
18. Patrice Koehl and Michael Levitt. A brighter future for protein structure prediction. *Nature Structural Biology*, 6:108–111, 1999.
19. Kit Fun Lau and Ken A. Dill. A lattice statistical mechanics model of the conformational and sequence spaces of proteins. *Macromolecules*, 22:3986 – 3997, 1989.
20. Hao Li, Robert Helling, Chao Tnag, and Ned Wingreen. Emergence of preferred structures in a simple model of protein folding. *Science*, 273:666–669, 1996.
21. Neil Madras and Gordon Slade. *The Self-Avoiding Walk*. Probability and Its Applications. Springer, 1996.
22. Tobias Müller and Jörg 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.
23. Britt H. Park and Michael Levitt. The complexity and accuracy of discrete state models of protein structure. *Journal of Molecular Biology*, 249:493–507, 1995.
24. J.-C. Regin. A filtering algorithm for constraints of difference in CSPs. In *Proc. 12th Conf. American Assoc. Artificial Intelligence*, volume 1, pages 362–367. Amer. Assoc. Artificial Intelligence, 1994.
25. Gert Smolka. The Oz programming model. In Jan van Leeuwen, editor, *Computer Science Today*, Lecture Notes in Computer Science, vol. 1000, pages 324–343. Springer-Verlag, Berlin, 1995.
26. R. Unger and J. Moult. Genetic algorithms for protein folding simulations. *Journal of Molecular Biology*, 231:75–81, 1993.
27. Ron Unger and John Moult. Local interactions dominate folding in a simple protein model. *Journal of Molecular Biology*, 259:988–994, 1996.
28. A. Šali, E. Shakhnovich, and M. Karplus. Kinetics of protein folding. *Journal of Molecular Biology*, 235:1614–1636, 1994.
29. Yu Xia, Enoch S. Huang, Michael Levitt, and Ram Samudrala. Ab initio construction of protein tertiary structures using a hierarchical approach. *Journal of Molecular Biology*, 300:171 – 185, 2000.
30. Kaizhi Yue and Ken A. Dill. Forces of tertiary structural organization in globular proteins. *Proc. Natl. Acad. Sci. USA*, 92:146 – 150, 1995.