Using Constraint Programming for Lattice Protein Folding

Rolf Backofen
Institut für Informatik
Ludwig-Maximilians-Universität München
Oettingenstraße 67, D-80538 München
Email: `backofen@informatik.uni-muenchen.de`

Abstract

We present a global search technique for finding the global minimal conformation of a sequence in Dill's HP-lattice model[5,6]. The HP-lattice model is a simplified model of proteins, that has become a major tool for investigating general properties of protein folding.

The search technique uses constraint programming for efficiently pruning the search tree. We state the problem of structure prediction in the HP-lattice model and describe our implementation using the Oz-system[7].

1 Introduction

The protein folding problem is one of the major unsolved problems in computational biology. For this reason, simplified models have been introduced, which have become a major tool for investigating general properties of protein folding.

An important class of simplified models are the so-called lattice models. The simplifications 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 positions in a regular lattice. Thus, every conformation of a lattice protein is a self-avoiding walk in $\mathbf{Z}^2$ or $\mathbf{Z}^3$ (depending on whether one considers two-dimensional or three-dimensional lattice models). A discussion of lattice proteins can be found in Dill et al.[2].

The most predominant representative of lattice models is the HP-model, which was introduced by Lau and Dill[5,6]. In this model, the 20 letter alphabet of amino acids (and the corresponding manifoldness of forces between them) 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

	H	P
H	-1	0
P	0	0

which 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, but occupy neighboring positions in the conformation (i.e., the distance vector between their positions in the conformation is a unit vector). A conformation with *minimal energy* (in the following called *optimal conformation*) is just a conformation with the maximal number of contacts between H-monomers. Figure 1 shows a conformation for the sequence PHPPHHPH in the 2-dimensional lattice whose energy is -2.

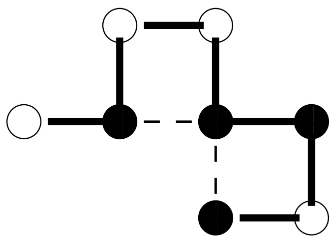

Figure 1: Sample conformation for PHPPHHPH. The white beads represent P, the black ones H monomers. The two contacts are indicated via dashed lines.

2 Previous Work

In the literature, there were several algorithms proposed for the HP-model. E.g., there are heuristic approaches such as the hydrophobic zipper [3], the genetic algorithm by Unger and Moult [9] and the chain growth algorithm by Bornberg-Bauer [1]. Another example is an approximation algorithm as described in Hart and Istrail [4], which in linear time produces a conformation whose energy is know to be at least $\frac{3}{8}$ of the optimal energy (which, on the other hand, is not an approximation ratio that can be used in an application). And there is an exact algorithm, namely the CHCC of Yue and Dill [10], which finds all optimal conformations.

Our algorithm is motivated by the CHCC-algorithm [10]. The main idea of CHCC is that the surface area of the hydrophobic core is more easily to estimate (given partial information about the final conformation) than the number of HH-contacts, and that the core surface area and the number of contacts are related one-to-one. Using this observation, in a first step, CHCC enumerates all possible shapes of the region containing all H-monomers of the given sequence (i.e., all core shapes). This enumeration is done in a way such that core shapes with a smaller surface area are enumerated before core shapes with a larger surface area. For every core shape, CHCC enumerates all positions of the monomers that fit into the given core shape. CHCC uses some conditions (or constraints) to reduce the size of the search tree. The motivation behind CHCC was to provide an algorithm which encounters *all* optimal conformations of a sequence.

We had a different motivation for our algorithm. The first motivation was to provide a fast algorithm to find the minimal energy for a given sequence (see the results section for some sample run times). In this case, it is better not to enumerate the explicit core shapes before enumerating the conformations. The reason is that different core shapes share many sub conformation, which means that part of an optimal conformation is enumerated several times in the CHCC algorithm. This makes no difference if one wants to enumerate all optimal conformations, but causes an overhead if one searches for only one optimal conformation. The second, and equally important motivation, was to provide an algorithm, which allows to investigate different search strategies. Here, again enumerating the core shapes directly is a disadvantage.

3 Constraint Programming

Constraint Programming is a relatively new programming technique, which allows to combine a declarative definition of a problem (like in PROLOG) with a precise definition of the operational behavior of the program. It may not be mistaken for constraint *logic* programming, although constraint programming has some of its roots in constraint logic programming. Constraint logic programming is usually a sequential form of programming with a fixed builtin search strategy. (Concurrent) Constraint Programming is an inherently concurrent programming paradigm, since all constraints are handled in parallel, which causes a mutual reinforcements of the constraints. Furthermore, the search strategy is not fixed, and in the case of the Oz-System[8,7] we are using, the search strategy can easily be programmed. Note that the word *constraint* is often used without using the programming paradigm underlying constraint programming. Thus, although CHCC means *Constrained Hydrophobic Core Construction*, it does not use the techniques of constraint programming.

4 Description of the algorithm

4.1 Surface and Energy

A sequence is an element in $\{H, P\}^*$. With s_i we denote the $i-th$ element of s. A conformation c of a sequence s is a function

$$c : [1..|s|] \to \mathbf{Z}^d$$

(where $d = 2$ or $d = 3$ depending on whether we consider 2-dimensional or 3-dimensional lattice) such that

1. $\forall 1 \leq i < |s| : ||c(i) - c(i+1)|| = 1$ (where $||\cdot||$ is the euclidic norm on $\mathbf{Z}^d$)
2. and $\forall i \neq j : c(i) \neq c(j)$.

The first condition is imposed by the lattice constraint and implies that the distance vector between to successive elements must be unit-vectors (or negative unit-vectors) in every admissible conformation. The second condition is the constraint that the conformation must be self-avoiding.

Given a conformation c of a sequence s, the number of contacts $Contact_s(c)$ in c is defined as the number of pairs (i,j) with $i+1 < j$ such that

$$s_i = H \wedge s_j = H \wedge ||c(i) - c(j)|| = 1$$

(in other words, the number of pairs of H-monomers that have distance 1 in the conformation c, but are not successive in the sequence s). The energy of c is just $-Contact_s(c)$. The *surface* $Surf_s(c)$ is defined as the number of neighbor positions of all $c(i)$ with s_i is a H-monomer that are not occupied by other H-monomers.

Now Yue and Dill [10] made the observation that there is a simple linear equation relating surface and energy. This equation uses the fact that every monomer has $2 \cdot d$ neighbors in the $\mathbf{Z}^d$, each of which is in any conformation either filled with either a H-monomer, a P-monomer, or left free. Let n_H^s be the number of H-monomers (resp. P-monomers) in s, and then we have for every conformation c that

$$2 \cdot d \cdot n_H^s \quad = \quad 2 \cdot [Contact_s(c) + HHBonds(s)] + Surf_s(c) \tag{1}$$

where $HHBonds(s)$ is the number of bonds between H-monomers (i.e., the number of H-monomers whose successor in s is also a H-monomer). Since $HHBonds(s)$ is constant for all conformation c of s, this implies that minimizing the surface is the same as maximizing the number of contacts.

4.2 Overall Search Structure

Our algorithm is a combination of a Branch-and-Bound search together with a constrain-and-generate principle, as it is usual for constraint optimization. We minimize the variable `Surface`, which has a finite domain associated.

Our algorithm works as follows. Initially, all constraint are set up. Then, in a generate step, a variable *var* is selected whose value is not yet determined, and a corresponding value *val* out of the associated range is chosen. Then the variable is set to this value in the first branch (i.e., the constraint *var*=:*val* is added). In the second branch, which is visited after the first branch is completed, the constraint *var*$\neq$:*val* is added. In both cases, after the generate step, the constraint programming system evaluates the effected variables according to the constraints, which results in an association of smaller value ranges to some (or many) variables and thus prune the search tree by removing inconsistent conformations.

At every step of the search, we call the set of finite domains associated to the variables and the set of determined variables and their associated value the *configuration* at that step. The configuration reflects a set of possible conformation that are compatible with this configuration. The finite domain associated with `Surface` reflects the range of surfaces of the conformations compatible with the configuration. Thus, the lower bound of `Surface` is the lower bound used for pruning the search tree in the branch-and-bound search.

The generate-and-constraint steps are iterated until all variables are determined (which implies, that a valid conformation is found). If we have found a valid conformation c, then the constraints will guarantee that `Surface` is determined and the associated value is $Surf_s(c)$. Then the additional constraint

$$\texttt{Surface} \quad < \quad Surf_s(c) \tag{2}$$

is added, and the search is continued in order to find the next conformation, which must have a smaller surface due to (2). This implies that the algorithm finally finds a conformation with minimal surface (and henceforth with maximal number of contacts by Equation (1).

4.3 The Variables

In the first step, the frame of the optimal conformation is calculated using the approach of [10]. Given a conformation, the *frame* of the conformation is the minimal rectangular box that contains all H-monomers of the sequence. A frame is uniquely determined by its dimension in x, y and z-direction. Yue and Dill [10] provided a method to calculate a lower bound on the surface when all H-monomers are packed within a specific frame. Thus, there are usually a few frames to be searched through to find the optimal conformation, since often bigger frames have a higher lower bound for the surface than an optimal sequence found in a smaller frame. For all examples in [10], there is even only one frame that has to be searched through. The assumption of a frame that contains all H-monomers is an efficient way of excluding many non-optimal conformations. Note that also some of the P-monomers must be included within this frame, namely those P-monomers whose left and right neighbor in chain is a H-monomer. The reason is just that one cannot include the surrounding H-monomers into the core without also including the middle P-monomer. These P-monomers are called *P-singlets* (see [10] for a more detailed description of P-singlets).

Our constraint problem consists of different variables, which have all a finite domain associated. Given a specific sequence s, the variables of our constraint problem are listed in Figure 2.

`Frx, Fry, Frz`	dimension of the frame
$\mathtt{X}_i$, $\mathtt{Y}_i$, $\mathtt{Z}_i$	x-,y-, and z-coordinate of the i^{th} monomer (where $1 \leq i \leq \text{length}(s)$)
$\mathtt{E}_j\mathtt{.seh}$, $\mathtt{E}_j\mathtt{.soh}$	number of even and odd H-monomers of the j^{th} x-plane (or x-layer) in the frame, respectively (where $1 \leq j \leq \mathtt{Frx}$);
$\mathtt{E}_j\mathtt{.sep}$, $\mathtt{E}_j\mathtt{.sop}$	number of even and odd P-singlets of the j^{th} x-layer in the frame, respectively
$\mathtt{P}_k\mathtt{.ctp}$	type of the k^{th} position of the frame (where $1 \leq k \leq \mathtt{Frx} \cdot \mathtt{Fry} \cdot \mathtt{Frz}$); the core type $\mathtt{P}_k\mathtt{.ctp}$ of the k^{th} position is either 1, if it is occupied by a H-monomer, and 0 otherwise
$\mathtt{O}_i^k$	for every position k of the frame and every monomer i; $\mathtt{O}_i^k$ has boolean value (i.e., 0 or 1), and is 1 iff monomer i occupies the k^{th} position of the frame.
$\mathtt{Surf}_k^l$	surface contribution between neighbour positions k and l under the condition, that k is occupied by a H-monomer. Thus, k must be within the frame, and l can be within the frame or outside the frame with distance 1 from the frame boundaries
`Surface`	complete surface of the conformation

Figure 2: The variables and their description

Initially, all the variables are undetermined, i.e., there is no fixed value but a range of values associated to each variable (e.g., the range for the variables $\mathtt{X}_i$, $\mathtt{Y}_i$, and $\mathtt{Z}_i$ is $1 \ldots (2 \cdot \text{length}(s))$). A *solution* of this constraint problem is an assignment of values to all variables, which respects all constraints we will list below. Although the $\mathtt{X}_i$, $\mathtt{Y}_i$, and $\mathtt{Z}_i$ are sufficient to describe a conformation, we need the other variables for efficiently pruning the search tree.

4.4 Constraints and Search Strategy

The basic constraints, which describe basic properties of self-avoiding walks, are the following. The self-avoidingness is just

$$(\mathtt{X}_i, \mathtt{Y}_i, \mathtt{Z}_i) \neq (\mathtt{X}_j, \mathtt{Y}_j, \mathtt{Z}_j) \qquad \text{for } i \neq j.\text{[a]}$$

Next, we have to express that the distance between two successive monomers is 1. For this purpose, we introduce for every monomer i with $1 \leq i < \text{length}(s)$ three variables $\mathtt{Xdiff}_i$, $\mathtt{Ydiff}_i$ and $\mathtt{Zdiff}_i$, which all have the value range $0 \ldots 1$ associated. Then we can express the unit-vector distance constraint by

$$\begin{aligned} \mathtt{Xdiff}_i &=: \mathtt{Abs}(\mathtt{X}_i - \mathtt{X}_{i+1}) \\ \mathtt{Ydiff}_i &=: \mathtt{Abs}(\mathtt{Y}_i - \mathtt{Y}_{i+1}) \\ \mathtt{Zdiff}_i &=: \mathtt{Abs}(\mathtt{Z}_i - \mathtt{Z}_{i+1}) \\ \mathtt{Sum}[\mathtt{Xdiff}_i, \mathtt{Ydiff}_i, \mathtt{Zdiff}_i] &=: 1 \end{aligned} \tag{3}$$

Both **Abs** and **Sum** are builtin functions of the constraint programming language Oz. The important property of these builtin functions is that they allow constraint propagation on undetermined variables (i.e., variables where only a range of values is associated).

The other constraints are as follows. Clearly, we must have

$$\begin{aligned} \sum_{j=1}^{\mathtt{Frx}} \mathtt{E}_j\mathtt{.soh} &= \text{number of odd H-monomers in s} \\ \sum_{j=1}^{\mathtt{Frx}} \mathtt{E}_j\mathtt{.seh} &= \text{number of even H-monomers in s} \\ \sum_{j=1}^{\mathtt{Frx}} \mathtt{E}_j\mathtt{.sop} &= \text{number of odd P-singlets in s} \\ \sum_{j=1}^{\mathtt{Frx}} \mathtt{E}_j\mathtt{.sep} &= \text{number of even P-singlets in s} \end{aligned} \tag{4}$$

[a] This cannot be directly encoded in Oz, but we reduce this constraints to difference constraints on integers

Furthermore, we have for every layer j that

$$\mathtt{E}_j\,.\,\mathtt{soh} + \mathtt{E}_j\,.\,\mathtt{seh} + \mathtt{E}_j\,.\,\mathtt{sop} + \mathtt{E}_j\,.\,\mathtt{sep} \quad \leq \quad \mathtt{Fry} \cdot \mathtt{Frz} \tag{5}$$

We write $\mathtt{Elem}_j^i = 1$ if the i^{th} monomer is an element of the j^{th} layer in x-direction. Then $\mathtt{Elem}_j^i$ can be defined by

$$\mathtt{Elem}_j^i = 1 \quad \Leftrightarrow \quad \mathtt{X}_i = j + \text{x-coordinate of starting point of frame.}$$

Furthermore, we can state that whenever two monomers i and $i+3$ are in the same layer, then $i+1$ and $i+2$ must also be in one layer due to the condition that we must fold into a lattice conformation. I.e., for every $1 \leq j \leq \mathtt{Frx}$ we have

$$\mathtt{Elem}_j^i = 1 \wedge \mathtt{Elem}_j^{i+3} = 1 \quad \Rightarrow \quad \mathtt{X}_{i+1} = \mathtt{X}_{i+2}$$

Furthermore, there is a special treatment of P-singlets, which may not be buried into the core (forming a caveat) in order to achieve an optimal conformation (Yue and Dill [10] have a similar strategy for avoiding caveats; they have calculated that this strategy is in almost all cases correct for sequences with length < 70). Thus we can state for every P-singlet i that

$$\begin{aligned} \mathtt{Elem}_j^i = 1 \wedge \mathtt{Elem}_j^{i+1} = 0 \quad &\Rightarrow \quad \mathtt{Elem}_j^{i-1} = 1 \\ \mathtt{Elem}_j^i = 1 \wedge \mathtt{Elem}_j^{i-1} = 0 \quad &\Rightarrow \quad \mathtt{Elem}_j^{i+1} = 1. \end{aligned}$$

At some stage of the search we have to assign monomers to frame positions. A monomer i is assigned the position k by setting $\mathtt{O}_i^k$ to 1 in one branch (which has just the effect that y_i and z_i is set to the y- and z-coordinate of the position k), and 0 in the other. Self-avoidingness is achieved by

$$\mathtt{Sum}[\mathtt{O}_1^k, \ldots \mathtt{O}_{\mathrm{length}(s)}^k] \quad =<: \quad 1$$

But there are additional constraints which restricts the core type and the monomers that can be placed at some position. If at some stage we know that no monomers can be placed at some position k, then we know that the core type must be 0. This is implemented by the constraint

$$(\mathtt{Sum}[\mathtt{O}_{i_1}^k, \ldots, \mathtt{O}_{i_n}^k] =: \mathtt{0}) \quad \Longleftrightarrow \quad (\mathtt{P}_k\,.\,\mathtt{ctp} =: 0),$$

where $i_1, \ldots, i_n$ are all H-monomers in s. This kind of constraint is called a *reified constraint*, and can directly be stated this way in the language Oz. There are other constraints which relates the core types of different positions, but we do not state them here for simplicity reasons.

Finally, we have constraints relating core types of positions and surface contributions:

$$\texttt{Surface} \;=:\; \sum_{k,l} \texttt{Surf}_k^l$$

If l is a position outside the frame (i.e., if its x,y or z-coordinate is outside the frame), then

$$(\texttt{Surf}_k^l =: \texttt{1}) \iff (\texttt{P}_k\texttt{.ctp} =: \texttt{1})$$

Otherwise we have $(\texttt{Surf}_k^l =: \texttt{1}) \iff (\texttt{P}_k\texttt{.ctp} =: \texttt{1}) \wedge (\texttt{P}_l\texttt{.ctp} =: \texttt{0})$

Search Strategy Our search strategy is as follows. We select the variables according to the following order (from left to right)

$$\begin{array}{ccccccc} \texttt{Frx} & & \texttt{E}_j\texttt{.seh} & & & & \texttt{X}_i \\ \texttt{Fry} & < & \texttt{E}_j\texttt{.soh} & < & \texttt{O}_i^k & < & \texttt{Y}_i \\ \texttt{Frz} & & \texttt{E}_j\texttt{.sep} & & & & \texttt{Z}_i \\ & & \texttt{E}_j\texttt{.sop} & & & & \end{array}$$

That means, in the first round, the frame dimensions are select for the search. After the frame dimensions have been chosen, the number of even and odd H-monomers and P-singlets are assigned to the different layers in x-direction. Constraints that prune the search tree at that point are clearly (4) and (5). But there are additional constraints which allows us to give a more precise lower bound on the surface and thus prune the search tree more efficiently. The basis for these constraints are two observations. The first one, made in [10], is that the surface of conformation in some specific layer depends only on the minimal rectangular box that can be drawn around the H-monomers in this layer. The second, to our knowledge first stated in [4], is the observation that in regular lattice models, contacts can only be formed between monomers whose sequence numbers have different parity (i.e., monomers whose sequence index is even can have only contacts with those whose index is odd, and vice versa). Using the second observation we were able to calculate the minimal rectangular box that can contain the $\texttt{E}_j\texttt{.soh}$ odd and $\texttt{E}_j\texttt{.seh}$ even H-monomers in layer j (see 3). From this we get the minimal surface contribution in y- and z-direction. The minimal surface in x-direction (for every layer j) is just

$$|\texttt{E}_j\texttt{.soh} - \texttt{E}_{j+1}\texttt{.seh}| + |\texttt{E}_j\texttt{.seh} - \texttt{E}_{j+1}\texttt{.soh}|.$$

Both together yields a lower bound for the surface and is used to prune the search tree.

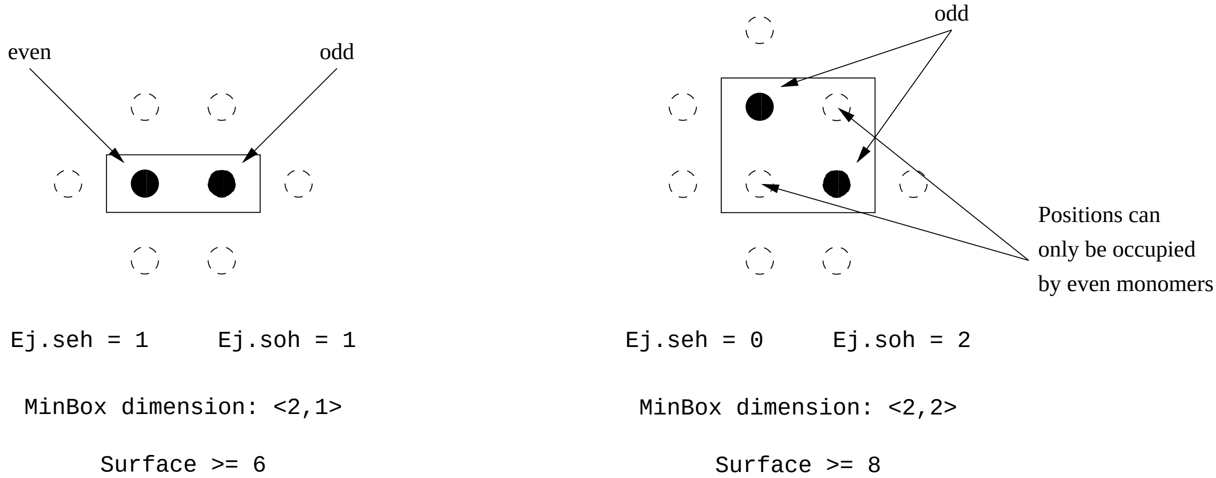

Figure 3: Minimal enclosing box for different assignments of $\mathtt{E}_j\mathtt{.soh}$ and $\mathtt{E}_j\mathtt{.seh}$.

In the following step, we assign H-monomers and P-singlets to the different layers according to $\mathtt{E}_j\mathtt{.seh}$, $\mathtt{E}_j\mathtt{.soh}$, $\mathtt{E}_j\mathtt{.sep}$ and $\mathtt{E}_j\mathtt{.sop}$. If all H-monomers and P-singlets are assigned to layers, we search for the positions of these monomers within the frame. The final step consists of assigning x-, y- and z-values to all monomers which are neither an H-monomer nor a P-singlet.

5 Results

We have tested the program on all sequences presented in [10]. For all we found an optimal conformation. In table 1, we have listed the test sequences together with the found optimal conformation, the sequence length and the optimal surface.
In table 2, we have listed

1. the number of steps to find a first conformation,
2. the number of steps to find the optimal conformation, if the first was not optimal (which was only the case for the sequence L2),
3. the total number of steps required to prove that the found conformation is an optimal conformation,

[b] Yue and Dill [10] have stated that the optimal surface is 16, but this is a typo since the conformation they have shown for this sequence has a surface of 32.

	Sequence and Sample Conformation	Length	Optimal Surface
L1	HPPPPHHHHPPHPHPHHHPHPPHHPPH	27	40
	RFDBLLFRFUBULBDFLUBLDRDDFU		
L2	HPPPHHHHPHPHHPPPHPHHPHPPPHP	27	38
	RFDLLBUURFDLLBBRURDDFDBLUB		
L3	HPHHPPHHPPHHHHPPPHPPPHHHPPH	27	38
	RFLDLUBBUFFFDFURBUBBDFRFDL		
L4	HHPHHPHHPHHHHHHPPHHHHHPPHHHHHHH	31	52
	RRFDBLDRFLLBUFLURFDDRFUBBUFRDD		
L5	PHPPHPPHPPHPPHPPHPPHPPHPPHPPHPPHPPHP	36	32[b]
	RFDBDRUFUBRBLULDLDRDRURBLDLULURBRFR		

Table 1: Test sequences. Below every sequence, we list an optimal conformation. Every conformation is represented as a sequence of bond direction (R=right,L=left,F=forward,B=backward,U=up and D=down).

4. and the total runtime on a Pentium 180 Pro.

Seq.	1^{st} Conf.		2^{nd} Conf.		total # Steps	Runtime
	# Steps	Surface	# Steps	Surface		
L1	914	40 (opt.)	—	—	921	3.85 sec
L2	1322	40	1345	38 (opt.)	5372	1 min 35 sec
L3	1396	38 (opt.)	—	—	1404	4.09 sec
L4	35	52 (opt.)	—	—	38	0.68 sec
L5	1081	32 (opt.)	—	—	1081	4.32 sec

Table 2: Search time and Number of Search Steps for the sample sequences.

It is hard to relate these results with the results given in [10]. Due to the fact that we do not explicitly enumerate the form of the core, our algorithm is better suited for finding one best conformation instead of all best conformations. Not enumerating the core form has the effect, that every sub conformation is only enumerated once. On the other hand, Yue and Dill's algorithm is especially designed for finding all conformations, which clearly results in longer runtime. Using this approach, enumerating the core form is a good strategy since it allows to introduce additional constraints. The negative side effect of enumerating sub conformations several times has no implication if one is interested in all optimal conformation. The runtime (on a Sun4) for all optimal conformations in [10] are 1 h 38 min for L1, 1 h 14 min for L2, 5 h 19 min for

L3, 5 h 19 min for L4 and 20 min for L5, respectively. There is a newer, more efficient version of this algorithm reported in [11], but there are no explicit runtime given for these or others sequences.

Acknowledgment

I would like to thank Prof. Karplus for helpful discussions on the topic of lattice models, and for motivating me to apply constraint programming techniques to lattice protein folding. I would like to thank Erich Bornberg-Bauer, who initiated this research, too. I would like to thank him also for explaining me the biological background, and for many discussion and hints.

1. Erich Bornberg-Bauer. Chain growth algorithms for hp-type lattice proteins. In *Proceedings of the first Annual International Conference on Computational Molecular Biology (RECOMB)*, pages 47 – 55. ACM Press, 1997.
2. 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.
3. Ken A. Dill, Klaus M. Fiebig, and Hue Sun Chan. Cooperativity in protein-folding kinetics. *Proc. Natl. Acad. Sci., USA*, 90:1942 – 1946, 1993.
4. William E. Hart and Sorin C. Istrail. Fast protein folding in the hydrophobid-hydrophilic model within three-eighths of optimal. *Journal of Computational Biology*, 3(1):53 – 96, 1996.
5. 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.
6. Kit Fun Lau and Ken A. Dill. Theory for protein mutability and biogenesis. *Proc. Natl. Acad. Sci., USA*, 87:638 – 642, 1990.
7. 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.
8. Gert Smolka. Problem solving with constraints and programming. *ACM Computing Surveys*, 28(4es), December 1996. Electronic Section.
9. R. Unger and J. Moult. Genetic algorithms for protein folding simulations. *Journal of Molecular Biology*, 231:75–81, 1993.
10. Kaizhi Yue and Ken A. Dill. Sequence-structure relationships in proteins and copolymers. *Pysical Review E*, 48(3):2267–2278, September 1993.
11. Kaizhi Yue and Ken A. Dill. Forces of tertiary structural organization in globular proteins. *Proc. Natl. Acad. Sci.*, 92:146 – 150, 1995.