On Lattice Protein Structure Prediction Revisited

Ivan Dotu Manuel Cebrián Pascal Van Hentenryck Peter Clote

Abstract—Protein structure prediction is regarded as a highly challenging problem both for the biology and for the computational communities. Many approaches have been developed in the recent years, moving to increasingly complex lattice models or even off-lattice models. This paper presents a Large Neighborhood Search (LNS) to find the native state for the Hydrophobic-Polar (HP) model on the Face Centered Cubic (FCC) lattice or, in other words, a self-avoiding walk on the FCC lattice having a maximum number of H-H contacts. The algorithm starts with a tabu-search algorithm, whose solution is then improved by a combination of constraint programming and LNS. This hybrid algorithm improves earlier approaches in the literature over several well-known instances and demonstrates the potential of constraint-programming approaches for *ab initio* methods.

Index Terms—Protein Structure, Constraint Programming, Local Search.

1 Introduction

In 1973, Nobel laureat C.B. Anfinsen [?] denatured the 124 residue protein, bovine ribonuclease A, by the addition of urea. Upon removal of the denaturant, the ribonuclease, an enzyme, was determined to be fully functional, thus attesting the successful reformation of functional 3-dimensional structure. Since no chaperone molecules were present, Anfinsen's experiment was interpreted to mean that the native state of some roteins[1] is its minimum free energy conformation, and hence that protein structure determination is a computational problem which can in principle be solved by applying a combinatorial search strategy to an appropriate energy model.

Protein structure prediction is historically one of the oldest, most important, yet stubbornly recalcitrant problems of bioinformatics. Solution of this problem would have an enormous impact on medicine and the pharmaceutical industry, since successful tertiary structure prediction, given only the amino acid sequence information, would allow the computational screening of potential drug targets. In particular, there is currently much research on computational docking of drugs (small chemical ligands) to a protein surface (such as a G-coupled protein receptor, the most common drug target).[2] Indeed, it has been stated that: "Prediction of protein structure *in silico* has thus been the 'holy grail' of computational biologists for many years" [?]. Despite the quantity of work on this problem over the past 30 years, and despite the variety of methods developed for structure prediction, no truly accurate *ab initio* methods exist to predict the 3-dimensional structure from amino acid sequence. Indeed, Helles (2008) [?] benchmarked the accuracy of 18 *ab initio* methods, whose average normalized root mean square deviation ranged from 11.17 Å to 3.48 Å, while Dalton and Jackson (2007) [?] similarly benchmarked five well-known homology modeling programs and three common sequence-structure alignment programs. In contrast, computational drug screening requires atomic scale accuracy, since the size of a single water molecule is about 1.4 Å.

In this paper, we describe a combination of constraint programming and Large Neighborhood Search (LNS) to determine close-to-optimal conformations for the Lau-Dill HP-model on the face-centered cubic lattice. Before describing our contribution, we first present an overview of computational methods for protein structure prediction. In general, methods are classified as *homology (comparative) modeling*, *threading*, *lattice model*, and *ab initio*. Protein structure prediction is an immense field that cannot be adequately surveyed in this introduction. Numerous books (e.g., [?]) and excellent reviews, (e.g., [?]) are available. Nevertheless, to situate the contribution of our work within the broader scope of protein structure prediction, we briefly describe each of the methods – homology, threading, *ab initio* – before focusing on lattice models.

In homology modeling, the amino acid sequence of a novel protein P is aligned against sequences of proteins Q, whose tertiary structure is available in the Protein Data Bank (PDB) [?]. Regions of P aligned to regions of Q are assumed to have the same fold, while non-aligned regions are modeled by interconnecting loops. Examples of comparative modeling software are SWISS-MODEL, developed by M. Peitsch, T. Schwede et al., and recently described in [?], as well as MODELER developed by the Šali Lab [?]. Comparative modeling relies on the assumption that evolutionarily related (homologous) proteins retain high sequence identity and adopt

- *Ivan Dotu Pascal Van Hentenryck are with the Department of Computer Science, Brown University, Box 1910, Providence, RI 02912.*
- *Manuel Cebrián is at the Human Dynamics Group, The Media Laboratory, Massachusetts Institute of Technology, Cambridge MA 02139.*
- *Peter Clote is with the Biology Department, Boston College, Chestnut Hill, MA 02467, and is Digiteo Chair, Laboratoire d'Informatique (LIX), Ecole Polytechnique and Laboratoire de Recherche en Informatique (LRI), Université Paris-Sud XI.*

1. Although most computational structure prediction methods equate the native state with the minimum free energy structure, the situation is more complicated, since some proteins fold cotranslationally, as in the P22 tailspike protein [?], while other proteins may have two or more low energy metastable structures, as in the prion protein [?].

2. Starting from the X-ray structure of HIV-1 protease with peptidomimetic inhibitors, Lam et al. [?] used computer-aided drug design tools and first principles in order to engineer a novel cyclic HIV-protease inhibitor.

the same fold.

Threading [?], [?], though known to be NP-complete [?], is a promising *de novo* protein structure approach, which relies on *threading* portions $a_i, \ldots, a_j$ of the amino acid sequence $a_1, \ldots, a_n$ onto a *fragment library*, which latter consists of frequently adopted partial folds. Pseudo-energy (aka knowledge-based potential) is computed from the frequency of occurrence of certain folds with certain types of amino acid sequence. Impressive results have been obtained with the Skolnick Lab program I-TASSER [?] with web server [?], which yielded the best-ranked structure predictions in the blind test CASP-7 (Critical Assessment of Techniques for Protein Structure Prediction) in 2006. Success of threading hinges on two things: *energetics*, i.e., that the PDB is relatively saturated and contains occurrences of almost all protein folds, and *search strategy*, i.e., usually Monte-Carlo or some type of branch-and-bound algorithm. According to a study of Zhang and Skolnick [?], the PDB is currently sufficiently saturated to permit adequate threading approaches, albeit with insufficient accuracy for the requirements of computational drug design.[3]

Despite advances in comparative modeling and threading, there is an interest in *ab initio* protein structure prediction, since this is the only method that attempts to understand protein folding from basic principles, i.e., by applying a search strategy with (generally) a physics-based energy function. Moreover, only *ab initio* methods can be applied for proteins having no homology with proteins of known structure. In molecular dynamics (MD), protein structure is predicted by iteratively solving Newton's equations for all pairs of atoms (possibly including solvent) using mean force potentials, that generally include pairwise (non-contact) terms for Lennard-Jones, electrostatic, hydrogen bonding, etc. Well-known MD software CHARMM [?] and Amber [?], as well as variant Molsoft ICM [?], the latter employing internal coordinates (dihedral angle space) and local optimization, are used to simulate protein docking, protein-ligand interactions, etc. since molecular dynamics generally is too slow to allow *ab initio* folding of any but the smallest proteins. Other *ab initio* methods include the Baker Lab program Rosetta [?], benchmarked in [?] with comparable accuracy as the Skolnick Lab program I-TASSER [?]. Search strategies of *ab initio* methods include molecular dynamics simulation, Metropolis Monte-Carlo (Rosetta [?]), Monte-Carlo with replica exchange (I-TASSER [?]), branch-and-bound (ASTROFOLD [?]), integer linear programming (ASTROFOLD [?]), Monte-Carlo with simulated annealing, evolutionary algorithms, and genetic algorithms.

2 Problem Formalization

A *lattice* is a discrete integer approximation to a vector space, formally defined to be the set of *integral* linear combinations of a finite set of vectors in $\mathbb{Z}^n$; i.e.,

$$L = \left\{ \sum_{i=1}^{k} a_i \vec{v}_i : a_i \in \mathbb{Z} \right\} \tag{1}$$

3. According to [?], using the TASSER algorithm, "in 408 cases the best of the top five full-length models has a RMSD $<$ 6.5 Ångstroms."

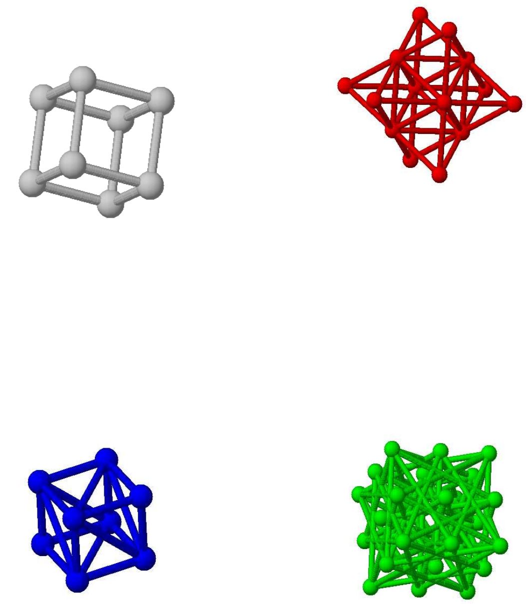


Fig. 1: Lattices used in protein structure modeling. *(a)* Points (x, y, z) in cubic lattice, satisfying $0 \leq x, y, z \leq 1$. *(b)* Points (x, y, z) in FCC lattice, satisfying $0 \leq x, y, z \leq 2$. *(c)* Points (x, y, z) in tetrahedral lattice, satisfying $0 \leq x, y, z \leq 1$. *(d)* Points (x, y, z) in 210 (knight's move) lattice, satisfying $0 \leq x, y, z \leq 2$.

where $\vec{v}_1, \ldots, \vec{v}_k \in \mathbb{Z}^n$. If k is the minimum value for which (**??**) holds, then $\vec{v}_1, \ldots, \vec{v}_k$ form a *basis*, and k is said to be the *dimension* (also called *coordination* or *contact* number) of L. Two lattice points $p, q \in L$ are said to be *in contact* if $q = p + \vec{v}_i$ for some vector $\vec{v}_i$ in the basis of L. Historically, many different lattices have been considered, some of which are depicted in Figure **??**: *(i)* the 2-dimensional square lattice with $n = 2$, coordination number 4, and basis vectors $(0, 1), (1, 0)$; *(ii)* the 2-dimensional triangular (aka hexagonal) lattice with $n = 2$, coordination number 6, and basis vectors $(1, 0)$, $(-1, 0)$,$(0, 1)$, $(0, -1)$, $(1, 1)$, $(-1, -1)$; *(iii)* the 3-dimensional cubic lattice with $n = 3$, coordination number 6, and basis vectors $(1, 0, 0)$, $(-1, 0, 0)$, $(0, 1, 0)$, $(0, -1, 0)$, $(0, 0, 1)$, $(0, 0, -1)$; *(iv)* the 3-dimensional face-centered cubic (FCC) lattice with $n = 3$, coordination number 12, and basis vectors $(1, 1, 0)$, $(-1, -1, 0)$, $(-1, 1, 0)$, $(1, -1, 0)$, $(0, 1, 1)$, $(0, 1, -1)$, $(1, 0, 1)$, $(1, 0, -1)$, $(0, -1, 1)$, $(-1, 0, 1)$, $(0, -1, -1)$, $(-1, 0, -1)$; *(v)* the tetrahedral lattice with $n = 3$, coordination number 12, and basis vectors $(1, 0, 0)$, $(-1, 0, 0)$, $(0, 1, 0)$, $(0, -1, 0)$, $(0, 0, 1)$, $(0, 0, -1)$, $(1, -1, 0)$, $(-1, 1, 0)$, $(1, 0, -1)$, $(-1, 0, 1)$, $(0, 1, -1)$, $(0, -1, 1)$; *(vi)* the 210 lattice [?] with $n = 3$, coordination number 24, and whose basis vectors $(-1, -2, 0)$, $(-1, 2, 0)$, $(1, -2, 0)$, $(1, 2, 0)$, $(-1, 0, -2)$, $(-1, 0, 2)$, $(1, 0, -2)$, $(1, 0, 2)$, $(-2, -1, 0)$, $(2, -1, 0)$, $(-2, 1, 0)$, $(2, 1, 0)$, $(0, -1, -2)$, $(0, -1, 2)$, $(0, 1, -2)$, $(0, 1, 2)$, $(-2, 0, -1)$, $(2, 0, -1)$, $(-2, 0, 1)$, $(2, 0, 1)$, $(0, -2, -1)$, $(0, 2, -1)$, $(0, -2, 1)$, $(0, 2, 1)$ resemble the move of a knight in chess. For more details on properties of these and other lattices, see the book by Conway and Sloane [?]. In this paper, we consider the face-centered cubic (FCC) lattice which is generated by the following 12 basis vectors (identified with compass directions [?]):

$$\begin{array}{lll} N:(1,1,0) & S:(-1,-1,0) & W:(-1,1,0) \\ E:(1,-1,0) & NW_+:(0,1,1) & NW_-:(0,1,-1) \\ NE_+:(1,0,1) & NE_-:(1,0,\_1) & SE_+:(0,-1,1) \\ SW_+:(-1,0,1) & SE_-:(0,-1,-1) & SW_-:(-1,0,-1). \end{array}$$

It follows that the FCC lattice consists of all integer points (x, y, z), such that $(x + y + z) \bmod 2 = 0$, and that lattice points $p = (x, y, z)$ and $q = (x', y', z')$ are in *contact*, denoted by $co(p, q)$, if $(x - x') + (y - y') + (z - z') \bmod 2 \equiv 0$, $|x - x'| \leq 1$, $|y - y'| \leq 1$, and $|z - z'| \leq 1$. We will sometimes state that lattice points p, q are at *unit distance*, when we formally mean that they are in contact. Since the distance between two successive alpha-carbon atoms is on average 3.8Å with a standard deviation of 0.04Å, a reasonable coarse-grain approach is to model an n-residue protein by a self-avoiding walk $p_1, \ldots, p_n$ on a lattice.

Many groups have employed the cubic model, despite the well-known parity problem, i.e., that if $p_1, \ldots, p_n$ is a self-avoiding walk on the cubic lattice, then p_i and p_j cannot be in contact for any two indices i, j of the same parity (both odd or both even). Covell and Jernigan [?] have shown that the FCC lattice, proven to admit the tightest packing of spheres [?], is the most appropriate 3-dimensional lattice for fitting protein C_α-atoms as a self-avoiding walk, and that *root mean square deviation* (`rms`) values are smaller for the FCC lattice than for the cubic, body-centered cubic and tetrahedral lattices. Here `rms` between two C_α-traces $(p_1, p_2, \ldots, p_n)$ and $(q_1, q_2, \ldots, q_n)$, where $p_i, q_i \in \mathbf{R}^3$, is given by $\sqrt{\frac{\sum_{i=1}^n (p_i - q_i)^2}{n}}$.

In 1972, Lau and Dill [?] proposed the *hydrophobic-hydrophilic* (HP) model, which provides a coarse approximation to the most important force responsible for the hydrophobic collapse which has been experimentally observed in protein folding. Amino acids are classified into either hydrophobic (e.g. Ala, Gly, Ile, Leu, Met, Phe, Pro, Trp, Val) or hydrophilic (e.g. Arg, Asn, Asp, Cys, Glu, Gln, His, Lys, Ser, Thr, Tyr) residues.[4] In the HP-model, there is an energy of -1 contributed by any two non-consecutive hydrophobic residues that are *in contact* on the lattice. For this reason, the HP-model is said to have a contact potential, depicted in the left panel of Figure **??**, where 'H' designates hydrophobic, while 'P' designates polar (i.e., hydrophilic). To account for electrostatic forces involving negatively charged residues (Asp, Glu) and positively charged residues (Arg, His, Lys), the HP-model has been extended to the HPNX-model, with hydrophobic (H), positively charged (P), negatively charged (N) and neutral hydrophilic (X) terms. The right panel of Figure **??** depicts the HPNX-potential used in [?].

Though Lau and Dill [?] originally considered only the 2-dimensional square lattice, their model allowed the formulation of the following simply stated combinatorial problem. For a given lattice and an arbitary HP-sequence, determine a self-avoiding walk on the lattice having minimum energy, i.e., a minimum energy lattice conformation. This problem was shown to be NP-complete for the 2-dimensional square lattice

4. This classification follows Rasmol [?], although other classifications are possible; indeed, proline is often considered hydrophilic.

	H	P
H	-1	0
P	0	0

	H	P	N	X
H	-4	0	0	0
P	0	+1	-1	0
N	0	-1	+1	0
X	0	0	0	0

Fig. 2: Energy for HP- and HPNX-model.

by Crescenzi et al. [?] and for the 3-dimensional cubic lattice by Berger and Leighton [?]. Recent work of Khodabakhshi et al. [?] considers the inverse folding problem for a variant of the HP-model[5] on the 3D hexagonal prism lattice.

While most work on lattice models concerns the HP-model, or closely related HPNX-model, other energy potentials for lattice models have been considered. Solvent *accessible surface area* (ASA), first introduced by Lee and Richards [?], has given rise to several measures, whose optimization can be carried out within a lattice model framework [?], for example *contact order* [?]. In [?], [?], Šali et al. considered a 27-mer with normally distributed hydrophobic contact potential having mean of -2 and standard deviation of -1, hence likely to fold into the $3 \times 3 \times 3$ compact cube. Šali et al. measured the average time required to reach the native state, formally the *mean first passage time* (MFPT), for a 27-mer to reach the minimum energy conformation on the compact cube, using a Monte Carlo simulation of protein folding on the cubic lattice. The authors claimed to have solved the Levinthal paradox by showing that thermodynamics suffices to drive a protein to rapidly find its native state.

3 RELATED WORK

3.1 Approaches to the HP Model

We first survey some search strategies for the HP-model. In [?], [?], Yue and Dill describe a *constrained hydrophobic core construction* (CHCC), which they apply in a branch-and-bound exhaustive search to construct self-avoiding walks in the cubic lattice having the maximum possible number of H-H contacts (native state). Subsequently, in [?], Yue and Dill applied "constraint-based exhaustive search" in their *Geocore* branch-and-bound method involving discretized dihedral angles, in order to determine the minimum energy conformation(s) of several small proteins including crambin. Since the number of conformations for an n-mer on the 3-dimensional cubic lattice is estimated to be approximately 4.5^n [?], it has been argued [?] that the HP model is a reasonable model in which to investigate the Levinthal paradox. In [?], Unger and Moult described a genetic algorithm for the HP-model on the 2-dimensional square lattice, where pointwise mutation corresponds to a conformation pivot move. This approach was extended in Backofen, Will, and Clote [?] to a genetic algorithm on the FCC lattice, in order to quantify hydrophobicity in protein folding. As previously mentioned,

5. In [?], Khodabakhshi et al. consider the HPC-model, where residues are classified as non-cysteine hydrophobic (H), hydrophilic (P), or cysteine (C). Since disulfide bonds are covalent bonds between sulfur atoms of cysteine residues, the HPC model allows one to model proteins containing disulfide bonds. The `DiANNA` web server [?], [?] predicts the disulfide bond topology of an input amino acid sequence, by using a novel architecture neural network.

Khodabakhshi et al. [?] developed an algorithm for inverse folding for the HPC-model on the 3D hexagonal prism lattice.

In [?], [?], [?], [?], Backofen and Will implemented the CHCC approach of Yue and Dill [?] using a modern constraint programming language (Oz), where symmetries were excluded, and both the cubic and FCC lattice were considered. In this fashion, Backofen and Will were able to provide an exact solution for small HP- and HPNX-sequences beyond the reach of earlier exhaustive methods. In [?], [?], Will precomputed *hydrophobic cores*, maximally compact face-centered cubic self-avoiding walks of (only) hydrophobic residues. By threading an HP-sequence onto hydrophobic cores, the optimum conformation can always be found, given sufficient (possibly exponential) computation time, provided the hydrophobic core has been precomputed.

Dal Palu et al. [?] use secondary structure and disulfide bonds formulated as constraints using constraint logic programming over finite domains to compute a predicted structure on the face-centered cubic lattice. They describe tests ranging from the 12 residue fragment (PDB code 1LE0) with RMSD of 2.8 Å achieved in 1.3 seconds, to the 63 residue protein (PDB code 1YPA) with RMSD of 17.1 Å in 10 hours. Further optimization was performed after the alpha-carbon trace was replaced by an all-atom model (presumably using well-known Holm-Sander method [?]), thus achieving an all-atom prediction of the 63 residue protein (PDB code 1YPA) with RMSD of 9.2 Å within 116.9 hours computation time. This study suggests that protein structure prediction might best proceed in a hierarchical fashion, first taking into account secondary structure on a coarse-grain lattice model and subsequently performing all-atom refinement.

3.2 Beyond the HP Model

The HP-model can be viewed as a coarse approximation of more complex *contact potentials*. In [?], Miyazawa and Jernigan introduced two kinds of contact potential matrices, i.e., 20×20 matrices that determine a residue-dependent energy potential to be applied in the case that two residues are in contact (either on the lattice, or within a fixed threshold such as 7 Å from each other). Pokarowski et al. [?] analyzed 29 contact matrices and showed that in essence all known contact potentials are one of the two types that Miyazawa and Jernigan [?] had previously introduced. Type 1 contact potential is given by the formula $e(i, j) = h(i) + h(j)$, where $1 \leq i \leq 20$ ranges over the 20 amino acids and h is a residue-type dependent factor that is highly correlated with frequency of occurrence of a given amino acid type in a non-redundant collection of proteins. Type 2 contact potential is given by the formula $e(i, j) = c_0 - h(i)h(j) + q(i)q(j)$, where c_0 is a constant, h is highly correlated with the Kyte-Doolittle hydrophobicity scale [?], and a residue-type dependent factor q is highly correlated with isoelectric points pI. The "knight's move" 210 lattice was used by Skolnick and Kolinski [?] to fold the 99-residue beta protein, apoplastocyanin, to within 2 Å of its crystal structure with PDB accession code 2PCY.

4 Local Search Approaches

This section introduces 2 different local search techniques that have been developed in order to compute FCC self-avoiding walks containing a near optimal number of unit distance H-H contacts. These techniques have been implemented as independent tools but also as part of a hybrid technique introduced in the following sections.

4.1 Tabu Search

4.1.1 The Model

This section presents our model for protein structure prediction. The model associates a decision variable v_i with every amino acid's position on the lattice. In other words, given a sequence of amino acids S such that $\|S\| = n$, the variable v_i takes its value in $\mathbb{Z}^3$ and represents the x, y, and z coordinates of the ith amino acid of S in the lattice. These variables must satisfy the following constraints:

- Self-Avoiding constraint: For all $i \neq j$: $v_i \neq v_j$.
- FCC Lattice Constraints: The sum of the coordinates of each point must be even.
- Adjacency: Two consecutive elements i and $i + 1$ must be neighbors in the lattice, i.e. in *contact* or at unit distance (as mentioned before, on the FCC, this means at Euclidean distance $\sqrt{2}$).

These are all hard constraints. They will hold initially and be preserved across local moves. In the following, we use σ to denote a complete assignment of the variables v_i that satisfies all the constraints.

4.1.2 The Fitness Function

The HP-model for protein structure prediction features an energy function which is rather poor in guiding the search towards high-quality solutions. Indeed, the number of H-H contacts only increases (decreases) when the algorithm positions (separates) two H amino acids at (from) unit distance; any other does not change the energy. As a result, a local-search algorithm based on such an objective will mostly perform a random walk.

To address this issue, our algorithm introduces a fitness function to guide the algorithm effectively. Define *distance* between two amino acids as $d(i, j)^2 = (x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2$, i.e., the square of the Euclidean distance between the ith and the jth amino acids in the current conformation of a sequence S of length n. Now consider the deviation from the unit distance (to the power of 2) to be $dv(i, j) = d(i, j)^2 - 2$. Our fitness function (or *cost*) is:

$$f(\sigma) = \sum_{i,j:i+1<j}^{n} (dv(i, j))^k \times (s_i = H, s_j = H)$$

where the sum is over i, j such that $i + 1 < j$ and $k \geq 1$ is a parameter of the algorithm. In particular, larger values of k give more weight to unit distances. Observe that these values are only defined when i and j correspond to H-type amino acids. The fitness function f is thus a measure of the deviation from the unit distance for every pair of (non consecutive) H-type amino acids. Therefore, in order to maximize the number of HH contacts, we need to minimize f.

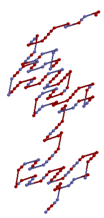


Fig. 3: Initial structure for the instance S4, an H-P sequence taken from [**?**, p. 130].

One may view f as a guide towards a compact structure where H-amino acids are close together, thus yielding several HH contacts. It is clear that, in order to achieve unit distance between H-type amino acids, they need to be close to each other. The impact of this fitness function will be better understood in the Experimental Results section. Note that $f(\sigma^*) = 0$ means that all pairs of H-type amino acids are at unit distance in σ^*.

4.1.3 The Self-avoiding Constraint

One of the constraints requires that all amino acid positions on the lattice be different. Representing this constraint explicitly is very costly and slows down the search considerably. Instead, the algorithm maintains the constraint implicitly. Each time a local move is performed on v_i, the algorithm only checks those amino acids v_j ($j \neq i$) whose norm is equal to $||v_i||$, since $v_i = v_j \Rightarrow ||v_i|| = |v_j||$. The constraint check is performed in $O(1)$ expected time, since the number of amino acids with the same norm is very low, even in the latest stages of the search process when the molecule is densely packed.

4.1.4 The Neighborhood

In this work, we allowed only one-monomer moves, in which only a single monomer changes position between two successive conformations. Our benchmarks suggest that a one-monomer move set suffices for good results on the FCC lattice, although since this is not the case for the CC lattice, for which Šali et al. [**?**] considered *crankshaft* (2-monomer) moves as well. If $p_1, \ldots, p_n$ denote current positions of monomers $1, \ldots, n$, then define the neighborhood $N(i)$ of the ith monomer as the set P of points p such that $d(p, p_i)^2 = 2\ ,\ p \in P$. A neighborhood move of the ith monomer of a tentative solution σ is defined to be a point in

$$S(\sigma, i) = \{p \in \mathbb{Z}^3 \mid p \in N(i-1) \wedge p \in N(i+1)\}$$

The neighborhood of σ can then be defined as

$$\mathcal{N}(\sigma) = \{(i,p) \ \mid \ 0 < i < n \ \wedge \ p \in S(\sigma, i)\}.$$

```
1.  PSPLS(S)
2.      forall i ∈ S
3.          tabu[i] ← {};
4.      σ ← initial configuration;
5.      σ* ← σ;
6.      l ← 0;
7.      s ← 0;
8.      while l ≤ maxIt   do
9.          select (i,p) ∈ 𝒩(σ)
                minimizing f(σ[v_i ← p]);
10.         τ ← RANDOM([4,n/2]);  tabu tenure time
11.         tabu[i] ←
                tabu[i] ∪ {move(i,p,σ)};
12.         σ ← σ[v_1 ← p];
13.         if f(σ) < f(σ*) then
14.             σ* ← σ;
15.             s ← 0;
16.         else if s > maxStable then
17.             σ ←random configuration;
18.             s ← 0;
19.             forall i ∈ S do
20.                 tabu[i] = {};
21.         else
22.             s++;
23.         l++;
```

Fig. 4: The Local Search Algorithm.

4.1.5 A Randomized Initialization

The initial solution has a significant impact on the quality and the speed of the local search algorithm. Given our one-monomer neighborhood and our fitness function, it is important to generate a feasible and compact initial solution with some HH contacts. The initialization iterates the following steps while there are amino acids to place.

1) Repeat a random number of times
   a) Repeat Forward for a random number of steps.
   b) Move Left.
   c) Repeat Backward for a random number of steps.
2) Move Up.
3) Switch moves with their opposites (e.g., Forward becomes Backward and Left becomes Right).

An initial configuration for the "S4" instance is depicted in Figure **??**.

4.1.6 The Tabu-Search Algorithm

We are now ready to present the basic local search algorithm. The algorithm, depicted in Figure **??**, a tabu search with a restarting component. Lines 2-7 perform the initializations. In particular, the tabu list is initialized in lines 2-3, the initial solution is generated in line 4, while lines 6 and 7 initialize the iteration counter k, and the stability counter s. The initial configuration σ is obtained in the manner explained above. The best solution found so far σ^* is initialized to σ.

The tabu list is distributed across the amino acids and maintains a set of moves. A move is formally defined as

$$move(i, p, \sigma) = p - \sigma(v_{i-1})$$

where $\sigma(v_{i-1})$ denotes the position of amino acid $i-1$ in assignment σ and p is the new position for amino acid i.

Note that the subtraction of v_{i-1} from p yields one of the basic vectors previously defined (N,S,W,E, ...). The tabu tenure is randomly selected between 4 and half the length of the sequence.

The core of the algorithm is given in lines 8-23, where local moves are iterated for a number of iterations. The local move is selected in line 9. Here, we use $\sigma[v_i \leftarrow p]$ to denote the solution obtained by changing the value of v_i to p in σ. The key idea is to select the best move in the neighborhood which is not tabu (meaning it has been previously performed) or which improves the best solution. The tabu list is updated in line 11, and the new tentative solution is computed in line 12. Lines 13-15 update the best solution, while lines 16-20 specify the restarting component.

The restarting component simply reinitializes the search from a random configuration whenever the best solution found so far has not been improved upon for *maxStable* iterations. Note that the stability counter s is incremented in line 22 and reset to zero in line 15 (when a new best solution is found) and in line 18 (when the search is restarted).

4.2 Tabu Search Over Two Neighborhoods

The second technique shares all the model related aspects of the previous one but it differentiates between two types of neighborhood:

1) The Neighborhood of the H (Hydrophobic) amino acids.
2) The Neighborhood of the P (Polar) amino acids.

The neighborhood of H amino acids is explored in the same fashion as the unique neighborhood was explored in the previous tabu search algorithm explained above. On the other hand, the P neighborhood is explored randomly, i.e., a random move within this neighborhood is selected at each iteration. Let us name $PSPL(H, numIter)$ as the effect of the previously explained tabu algorithm on neighborhood H for $numIter$ iterations and similarly $PSPL(H, numIter)$ to that of the neighborhood P. This new "meta" tabu search can be detailed as follows:

1) Do $PSPL(H, \inf)$ until fitness does not improve
2) Do $PSPL(P, r)$
3) Do $PSPL(H, p)$
4) Go to step 1

The main idea is to restrict tabu search to a neighborhood where all the amino acids are Hydrophobic until the fitness function cannot be improved any further. At this point we randomly move Polar amino acids (which do not have any impact on the fitness) r times as a diversification mechanism. After that, we explore the H neighborhood again but we allow the fitness function to oscillate for the first p iterations. The values of r and p are usually dependent on the length of the sequence.

4.3 A new initialization

We have also introduced a new method to find an initial solution. It simply consists of performing complete search using the Constraint Programming model detailed in the following section along with a greedy heuristic to evaluate ordering (choose first the value that maximizes the number of HH contacts).

We perform a limited optimization and return the best configuration found after a certain number of failures.

5 Large Neighborhood Search

Structure prediction is a highly complex combinatorial optimization problem. As a result, constraint programming search may spend considerable time in suboptimal regions of the search space. To remedy this limitation, our algorithm uses the idea of large neighborhood search (LNS) [**?**] which focuses on reoptimizing subparts of a solution.

Here we describe 3 different Large Neighborhood Search (LNS) approaches that utilize the previously described Local Search techniques and a constraint programming (CP) model that is detailed below.

5.1 The CP Model

The CP model receives as input a sequence of binary values H_i $(0 \leq i < n)$ denoting whether amino acid i is hydrophobic $(H_i = 1)$. Its output associates each amino acid i with a point (x_i, y_i, z_i) in the FCC lattice. Recall that the FCC lattice is the closure of 12 vectors $V = \{v_0, \ldots, v_{11}\}$ defined as follows:

$$
\begin{array}{lll}
v_0 = \{1, 1, 0\} & v_1 = \{-1, -1, 0\} & v_2 = \{-1, 1, 0\} \\
v_3 = \{1, -1, 0\} & v_4 = \{1, 0, 1\} & v_5 = \{-1, 0, -1\} \\
v_6 = \{-1, 0, 1\} & v_7 = \{1, 0, -1\} & v_8 = \{0, 1, 1\} \\
v_9 = \{0, -1, -1\} & v_{10} = \{0, -1, 1\} & v_{11} = \{0, 1, -1\}.
\end{array}
$$

5.2 Decision Variables

Although the output maps each amino acid i into a FCC lattice point, the model uses move vectors as decision variables. These vectors (m_i^x, m_i^y, m_i^z) specify how to move from point $i-1$ to point i in the self-avoiding walk. The use of *move variables* greatly simplifies the modeling: Self-avoidance is maintained through the lattice points, but move vectors along with a lexicographical variable ordering allow us to implicitly check chain connection and drastically reduces the search space.

5.3 The Domain Constraints

Each move variable (m_i^x, m_i^y, m_i^z) has a finite domain consisting of the FCC lattice vectors $\{v_0, \ldots, v_{11}\}$, that is

$$(m_i^x, m_i^y, m_i^z) \ \in \ \{v_0, \ldots, v_{11}\}.$$

Each coordinate x_i, y_i, and z_i in the 3D point (x_i, y_i, z_i) associated with amino acid i has a finite domain $0..2n$.

5.4 The Lattice Constraints

The lattice constraints link the *move variables* and the points in the FCC lattice. They are specified as follows:

$$\forall\, 0 < i < n : \ x_i = x_{i-1} + m_i^x \ \&\ y_i = y_{i-1} + m_i^y \ \&\ z_i = z_{i-1} + m_i^z.$$

The model also uses the redundant constraints $(x_i + y_i + z_i) \bmod 2 = 0$ which are implied by the FCC lattice. In addition, the initial point is fixed.

5.5 The Self-Avoiding Walk Constraints

To express that all amino acids are assigned different points in the FCC lattice, the model uses a constraint

$$abs(\sum_{k\in i..j} m_k^x) + abs(\sum_{k\in i..j} m_k^y) + abs(\sum_{k\in i..j} m_k^z) \neq 0$$

for each pair (i, j) of amino acids, ensuring the moves from the position of amino acid i do not place j at the same position as i. Indeed, the two points (x_i, y_i, z_i) and (x_j, y_j, z_j) are at the same position if each of the sums in the above expression is zero.

5.6 The Objective Function

The objective function maximizes the number of contacts between hydrophobic amino acids $\sum_{i,j|i+1<j}(d_{ij} = 2) \times H_i \times H_j$ where d_{ij} denotes the square of Euclidean distance between amino acids i and j. Since the minimal distance in the FCC lattice is $\sqrt{2}$, the condition $d_{ij} = 2$ holds when there exists a contact between amino acids i and j.

5.7 The Search Procedure

The search procedure assigns positions to the amino acids in sequence by selecting moves in their domains. The only heuristic choice thus concerns which moves to select. In the course of this research, a number of move selection heuristics were evaluated. Besides the traditional lexicographic and random value selections, the heuristics included

1) **Minimizing the distance to the origin:** Choosing the move minimizing the distance of the corresponding amino acid to the origin.
2) **Minimizing the distance to the centroid:** Choosing the move minimizing the distance of the corresponding amino acid to the centroid.
3) **Maximizing density:** Choosing the move maximizing the density of the structure.
4) **Maximizing hydrophobic density:** Choosing the move that maximizing the density of the structure consisting only of the hydrophobic amino acids.

The centroid of the conformation is defined as $(\frac{1}{n}\sum_{i=0}^{n-1} x_i, \frac{1}{n}\sum_{i=0}^{n-1} y_i, \frac{1}{n}\sum_{i=0}^{n-1} z_i)$. Most of the dedicated heuristics bring significant improvements in performance, although those minimizing the distance to the origin and the centroid seem to be most effective. Our implementation randomly selects one of the four heuristics.

5.8 Strengthening the Model During Search

We now describe a number of tightenings of the model which are applied during search. Their main benefit is to strengthen the bound on the objective function.

5.8.1 Linking FCC Moves and Distance Constraints

In the model described so far, the distance between two amino acids ignores the fact that the points are placed on the FCC lattice. The model may be improved by deriving the fact that two amino acids are necessarily placed at a distance greater than $\sqrt{2}$ and thus cannot be in contact. Such derived information directly improves the bound on the objective function.

However computing the possible distances between two amino acids is quite complex in general. As a result, our constraint-programming algorithm only generates relevant distances each time a new amino acid is positioned. More precisely, assuming that amino acid i has just been positioned on the FCC lattice, the algorithm determines which unassigned amino acids cannot be in contact with already placed amino acids (only for H-type amino acids). The key idea is to compute the shortest path sp_{ij} in the FCC lattice between amino acid i and an already placed amino acid j: It then follows that unassigned amino acids $i+1, \ldots, i+sp_{ij}-2$ cannot be in contact with j. Formally, after placing amino acid i, the model is augmented with the constraints

$$\forall 0 \leq j \leq i-2, i+1 \leq l \leq i+sp_{ij}-2 : d_{jl} > 2$$

which ensures that amino acids j and l cannot be in contact.

5.8.2 Bounding the Number of Contacts

The expression of the objective function also does not take into account how the amino acids are placed in the FCC lattice. As a result, it typically gives weak bounds on the objective value. This section shows how to bound the objective value at a search node more effectively.

The key idea to bound the objective value is to compute the maximum number of contacts for each unassigned amino acid independently, thus ignoring their interactions through the self-avoiding walk. Consider a node of search tree where the sequence can be partitioned into the concatenation $A :: U$, where A (assigned) is the subsequence of already positioned amino acids in which i is the last assigned, and where U (unassigned) contains the remaining unassigned amino acids (also, we only consider $a \in A | H_a == 1$ and $k \in U | H_k == 1$). The objective function can then be bounded by

$$obj \leq contact(A) +$$
$$\sum_{k\in U} \min(maxContact(k), bcontact(k, A) + fcontact(k, U))$$

where $contact(A)$ denotes the number of contacts in subsequence A, $bcontact(k, A)$ bounds the number of contacts of an amino acid $k \in U$ with those amino acids in A, and $fcontact(k, U)$ bounds the number of contacts of k with those amino acids in U occurring later in the sequence. The contacts of each amino acid $k \in U$, $maxContact(k)$, are bounded by 10, since a point in the FCC lattice has 12 neighbors and there cannot be any contact between two successive amino acid in the sequence. However, if $k == n-1$, i.e., if k is the last amino acid of the sequence then $maxContact(k) == 11$, since that k has no successor amino acid.

To bound the contact of amino acid k with A, the idea is to consider the neighbors of each amino acid $a \in A$ and to find the one maximizing the contacts with k, i.e.,

$$\begin{array}{lcl} bcontact(k, A) & = & \max_{a\in A} bcontact(k, a, A) \\ bcontact(k, a, A) & = & \#\{j \in A \mid j \in N(a) \wedge j \in R(k, a)\}. \end{array}$$

where $N(a)$ denotes the neighbors of amino acid a and $R(k, a)$ denotes the amino acid in A reachable from k, i.e.,

1. **LNS_PSP**(σ)
2. $limit \leftarrow limit_0$
3. $fraction \leftarrow fraction_0$
4. **for** m iterations **do**
5. **uniform select** $i \in 1..n-1$
6. $size \leftarrow n \cdot fraction$
7. $j \leftarrow i + size$
8. $\langle \sigma^*, explored \rangle$ = **CPSolve**($\sigma, i..j, limit$)
9. **if** $\sigma^* \neq \perp$ **then**
10. $\sigma \leftarrow \sigma^*$
11. $limit \leftarrow limit_0$
12. $fraction \leftarrow fraction_0$
13. **else if** *explored* **then**
14. $fraction \leftarrow fraction + \Delta fraction$
15. **else**
16. $limit \leftarrow limit + \Delta limit$
17. **return** σ

Fig. 5: LNS for Protein Structure Prediction ($limit_0$=500 failures, $fraction_0 = \frac{3}{100}$, $\Delta fraction = \frac{1}{1000}$ and $\Delta limit$=100 failures).

$R(k, A) = \{a \in A \mid sp_{ai} \leq (k-i)+1\}$. Recall that i is the last amino acid assigned. Finally, to bound the number of contacts of k with those amino acids occurring later in the sequence, we use

$$fcontact(k, U) = \sum_{l \in U : l \geq k+2} H_l$$

to count the number of hydrophobic amino acids occurring later in U that can be in contact with k. This bound can be computed in time $O(n^2)$ and is quite tight when the number of amino acids in U is reasonably small.

5.9 Sequence Reoptimized LNS

Given a feasible walk σ, the idea is to solve the structure prediction problem for a subsequence of the original sequence, assuming that the remaining amino acids are positioned like in σ. More precisely, given an interval $i..j$, an LNS optimization step consists of solving the original model with the additional constraints

$$\forall\, k : 0 \leq k < i : x_k = \sigma(x_k) \;\wedge\; y_k = \sigma(y_k) \;\wedge\; z_k = \sigma(z_k)$$

$$\forall\, k : j < k < n : x_k = \sigma(x_k) \;\wedge\; y_k = \sigma(y_k) \;\wedge\; z_k = \sigma(z_k)$$

where $\sigma(x)$ denotes the value of variable x in solution σ.

The complete LNS algorithm is depicted in Figure **??**. It receives as input a high-quality solution produced by the tabu-search algorithm described in [**?**] and uses a subroutine *CPSolve*($\sigma, i..j, l$) which solves augmented models using constraint programming and terminates after at most $limit$ failures had occurred or when the entire search space has been explored. It returns a pair $\langle \sigma^*, explored \rangle$, where σ^* is either a new best solution or $\perp$ if no such solution was found, and *explored* is a boolean which is true when the entire search space has been explored for the augmented model. Lines 2–3 initialize two parameters: the limit on the number of failures and the fraction of the subsequence to (re)-position on the FCC lattice. Line 8 is the call to the constraint-programming solver. After this call there are three possibilities. First, that the search is successful: then the best solution is updated and the parameters are re-initialized (lines 9–12). Second, the search space has been explored entirely with no improvement; the fraction of the sequence to re-position is increased at a certain rate $\Delta fraction$ (lines 13–14). Finally, *CPSolve* reached $limit$ without an improvement: the number of failures is increased in $\Delta limit$ to give it more time to succeed in the next trial (lines 15–16).

5.10 Multiple Sequence Reoptimized LNS

This is a slight modification of the previously defined LNS algorithm. It re-optimizes a solution iteratively but instead of fixing several amino acids and solving a subsequence, it solves several subsequences at the same time. In general, given a set of intervals I defined by $Im_i...Im_j$, for a given interval $Im \in I$, an LNS optimization step consists of solving the original model with the additional constraints

$$\forall\, k \notin I : x_k = \sigma(x_k) \;\wedge\; y_k = \sigma(y_k) \;\wedge\; z_k = \sigma(z_k)$$

where $k \in I$ means than $Im_i > k > Im_j$ for some interval $Im \in I$. The algorithm starts with one interval and it increases the number of intervals in successive runs.

5.11 3D Structure Reoptimized LNS

This second modification concerns fixing all the amino acids except for a set of 3D positions instead of subsequences. Given a set of 3D boxes B around an amino acid position from a feasible walk σ the LNS algorithm solves the original model adding the constraints

$$\forall\, k :\notin B : x_k = \sigma(x_k) \;\wedge\; y_k = \sigma(y_k) \;\wedge\; z_k = \sigma(z_k)$$

where $k \in B$ means than point $(\sigma(x_k), \sigma(y_k), \sigma(z_k)) \in Bm$ for some box $Bm \in B$. The algorithm starts with one box and it increases the number of boxes in successive runs.

6 Experimental Results

All sequences used in the benchmarking studies as well as additional scripts, program outputs, etc. can be found at http://bioinformatics.bc.edu/clotelab/FCCproteinStructure/. Results from Tables **??** and **??** were obtained by a COMET [**?**], [**?**] implementation of the LS and LNS algorithms, run on a single core of a 60 Intel based, dual-core, dual processor, Dell Poweredge 1855 blade server located in the Computer Science Department of Brown University. Each blade of the server had 8 Gygabytes of memory and 300 Gygabytes local disk. Results from Table **??** were performed on a cluster of Dell Power Edge 1950 4-core Intel E5430 processors with 2.66 GHz and 16 Gb of RAM, located in the Biology Department of Boston College. For tests at both Brown University and Boston College, PBS/Torque was used to batch the runs; however, no algorithmic parallelism, and each run used only one core.

LS based algorithms were run with a limit of 10,000 iterations. The LN-2N algorithm uses the Randomized Initialization initialization described in subsection 4.1 of this paper.

Hybrid Large Neighborhood Search (LNS-) based algorithms were run (after 10,000 iterations of Local Search) for 10 minutes on the "Harvard Instances" and for 30 minutes on the rest. Exact computation times for LS and LNS depended on sequence length and are given in Table **??**; however, total computation time per sequence was at most approximately 35 minutes.

6.1 The Harvard Instances

Reference [**?**] contains a comparison of several methods to fold 10 different proteins, called the "Harvard instances", on the cubic lattice. The cubic lattice has been heavily studied as pointed out in the introduction, but the FCC lattice has been shown to admit the tightest packing of spheres [**?**], indicating that it allows for more complex 3D structures. The first results for these instances on the FCC lattice were presented in [**?**], [**?**] and confirmed that the FCC lattice allows for structures with much lower energy than the cubic lattice.

Tables **??** and **??** depict the results of the LS and our Hybrid LNS algorithms. Note that the values shown in the table correspond to the number of HH contacts. The LNS step improves all solutions in less than 30 minutes. Since no complete search algorithms have been applied to these instances on the FCC lattice, the energy of the optimal structure is not known. However, given the consistency in the energies of all the sequences (which all have 48 amino acids and 24 hydrophobic amino acids), it is likely the case that these results are near-optimal.

6.2 Other Instances

The S and R instances, taken from [**?**], The only FCC foldings available in the literature are the S instances S1-S4, taken from page 130 of Will's dissertation [**?**], and the R instances R1-R3, taken from Table 1 of [**?**]. S. Will kindly supplied us with five additional instances F90 and three additional instances F180. Tables **??** and **??** compare the number of H-H contacts computed by each variant of algorithm presented in this paper, for the Harvard instances and the S,R,F90 and F180 instances. Table **??** indicates the number of residues, or sequence length (Len), the number of H residues (numH), as well as the maximum and average number of contacts found. Run time was fixed at a maximum of approximately 35 minutes, where Table **??** indicates the number of seconds used by the initial local search (LS) strategy, followed by the number of minutes used by (the variant) of large neighborhood search. Our results demonstrate that LNS significantly improves on the local search algorithm, with improvements ranging from 1.7% to 13%. The largest improvements occur on the R instances, which is tue to the lower quality of local search for these instances. Results for the S instances are within 4.5% of the optimal solution, while our algorithm is within 16.3% of the optimal solutions on the R instances.

Table **??** presents the nonconvergence rate for Will's hydrophobic core threading algorithm, `HPstruct`, for randomizations of the F90 and F180 instances with a run time bound of 30 minutes. In order to produce additional test instances that are similar to those provided us by S. Will, we we doubled the length of the F90 instances by concatenating a copy of the same sequence. Subsequently, for each of the 5 F90, 5 F90-doubled and 3 F180 sequences, we generated 10 randomized sequences having the *same* diresidues, by running our implementation of the Altschul-Erikson diresidue shuffle algorithm [**?**] using our code described in [**?**]. (Randomization code and all sequences are available at http://bioinformatics.bc.edu/clotelab/FCCproteinStructure/.) For each instance class, Table **??** indicates the number of sequences tested, their length and the failure rate of `HPstruct` with 30 minute time bound.[6] Within the 30 minute time bound, we had approximate solutions using LS+LNS; due to space restrictions, these results are available at our web site. Finally, Figure **??** depicts a 3D view of the best configuration found for S2 for some of the algorithms presented as well as the native state.

It is also important to stress how the optimal solutions were obtained in [**?**]. Will's algorithm solves a substantially different problem than we consider, namely, the problem of threading a HP-sequence onto hydrophobic cores from a collection of (off-line) precomputed H cores. Unlike the Yue-Dill CHCC method [**?**], [**?**], which computes H cores on the fly, the faster program, `HPstruct`, requires precomputed complete set of hydrophobic cores for each given number of H residues. While the Yue-Dill CHCC approach is slower than `HPstruct`, it can always (in principle) determine an optimal structure, provided computation time is unbounded (decades or eons). In contrast, Will's `HPstruct` can fail due to the unavailability of a precomputed optimal H core.

There is a fundamental conceptual difference between the algorithm(s), LS+LNS, presented in this paper and the hydrophobic-core constraint-programming methods of Yue and Dill [**?**], [**?**] and Will and Backofen [**?**], [**?**]. This difference can best be described using the concepts of *Monte-Carlo* and *Las Vegas* probabilistic algorithms from theoretical computer science [**?**]. Monte-Carlo algorithms always converge, but have a (small) probability of error in the solution proposed; in contrast, Las Vegas algorithms always return the correct solution, but have a (small) probability of not converging. By analogy, our approach (LS+LNS) is akin to a Monte-Carlo method, in that an approximate solution is always returned. In contrast, hydrophobic-core constraint programming is akin to a Las Vegas method, in that any solution returned is an exact (optimal) solution; however, in many cases, the hydrophobic-core method fails to return any answer. Table 6.1 on page 129 of [**?**, p. 129] states that the threading algorithm only solves 50% of the instances with an H core of size 100 within 15 minutes. This is corroborated by Table **??**, where it is shown that the failure rate of `HPstruct` can be almost 80% for length 180 sequences, and 100% for length 360 sequences,

6. As well, we doubled the length of the F180 sequences, resulting in total length of 360, by concatenating a copy of the same sequence. Subsequently, for each of the 3 F180-doubled sequences, we generated 10 randomized sequences using our implementation of the Altschul-Erikson algorithm. Although we obtained approximation solutions using local search (LS) alone, large neighborhood search (LNS) required too much memory when run on length 360 sequences using the Boston College cluster. However, in all these cases, Will's program `HPstruct` was unable to compute the optimal conformation using the H core for 200 hydrophobic residues, kindly sent to us by S. Will.

Seq.	Native E.	Len	numH	LS	LS-G	LS-2N	LS-2N-G	Time
H1	69	48	24	65 (57.50)	51 (47.17)	**68** (64.70)	**68** (64.61)	75 s
H2	69	48	24	64 (56.59)	55 (46.79)	**69** (64.32)	68 (62.51)	75 s
H3	72	48	24	66 (56.69)	58 (54.38)	**68** (62.08)	67 (62.51)	75 s
H4	71	48	24	65 (58.08)	56 (49.26)	67 (63.15)	**68** (63.10)	75 s
H5	70	48	24	64 (57.01)	57 (42.95)	67 (63.38)	**68** (63.79)	75 s
H6	70	48	24	63 (56.52)	40 (34.35)	**69** (63.38)	68 (64.91)	75 s
H7	70	48	24	63 (58.15)	49 (41.10)	**68** (63.36)	67 (63.75)	75 s
H8	69	48	24	63 (55.31)	54 (50.27)	**67** (62.20)	66 (62.56)	75 s
H9	71	48	24	67 (58.91)	54 (46.77)	**69** (64.90)	**69** (64.40)	75 s
H10	68	48	24	64 (57.47)	45 (30.03)	**67** (63.96)	**67** (63.61)	75 s
S1	357	135	100	296 (271.03)	276 (270.99)	343 (320.55)	**345** (323.81)	300 s
S2	360	151	100	304 (268.43)	250 (244.23)	**339** (318.30)	**339** (316.60)	300 s
S3	367	162	100	293 (259.55)	234 (228.71)	332 (310.02)	**337** (306.03)	300 s
S4	370	164	100	294 (263.73)	226 (222.99)	**337** (307.77)	329 (300.92)	300 s
R1	384	200	100	287 (240.85)	212 (205.58)	**292** (254.69)	291 (264.53)	500 s
R2	383	200	100	290 (239.12)	209 (205.60)	294 (262.74)	**296** (267.75)	500 s
R3	385	200	100	260 (230.57)	228 (212.12)	**305** (260.70)	299 (267.05)	500 s
F90_1	168	91	50	143 (125.75)	104 (102.97)	**154** (142.25)	153 (142.77)	180 s
F90_2	168	91	50	142 (123.68)	117 (112.05)	156 (141.45)	**157** (141.89)	180 s
F90_3	167	91	50	138 (121.80)	110 (101.7)	157 (143.79)	**159** (145.24)	180 s
F90_4	168	91	50	144 (124.35)	94 (92.74)	**162** (144.17)	158 (139.26)	180 s
F90_5	167	91	50	138 (121.59)	110 (107.65)	**157** (143.32)	154 (145.00)	180 s
F180_1	?	180	100	244 (204.28)	201 (188.06)	261 (232.30)	**265** (240.88)	300 s
F180_2	?	180	100	240 (222.40)	228 (211.07)	**279** (255.24)	278 (254.11)	300 s
F180_3	378	180	100	256 (227.69)	195 (191.91)	**292** (262.86)	287 (261.55)	300 s

TABLE 1: Largest number of contacts found for each Local Search (LS) based algorithm showing the average contacts over 100 runs in parenthesis. Sequence length and number of hydrophobic residues dictated the time used for local search: *(i)* Harvard instances: 75 sec. LS; *(ii)* S: 300 sec. LS; *(iii)* R: 500 sec. LS; *(iv)* F90: 180 sec. LS; *(v)* F180: 300 sec. Boldface font indicates the largest value found. Native E. (i.e. native energy) is the optimal number of contacts, LS is Tabu Search (i.e. local search) with randomized initialization, LS-G is Tabu Search with the new initialization, LS-2N is Two Neighborhoods Tabu Search with randomized initialization, LS-2N-G is Two Neighborhoods Tabu Search with the new initialization. The Harvard instances H1-H10 are taken from [**?**], the S instances S1-S4 from page 130 of Will's dissertation [**?**], the R instances R1-R3 from Table 1 of [**?**], and the F90 and F180 instances were provided to us by S. Will, who kindly provided us with precomputed hydrophobic cores and executable code for his hydrophobic core threading software, `HPstruct`, to permit testing. In all cases, native energy is computed using `HPstruct`, described in [**?**], [**?**], although the web server could only converge for the Harvard instances. For instances F180_1, F180_2, Will's program `HPstruct` did not converge within 35 minutes.

allowing for 30 minutes of computation. (In case of 100% failure, the cause is due to the unavailability of H cores for 200 H residues.) On small problem instances, as illustrated in Tables **??**,**??**, Will's method is in almost all cases the method of choice, providing a rapid computation of the exact solution. In contrast, for larger instances, as illustrated in Table **??**, there are serious problems of convergence of the threading algorithm – either the algorithm did not converge within 30 minutes or there were no precomputed hydrophobic cores necessary for the initialization of the threading algorithm.

From Tables **??**,**??**, it can be seen how the local search achieves initial solutions which are then quickly improved by the LNS. Running LNS for a longer time improves, in general, the average number of contacts obtained, with a gradual limit on improvement. Figure **??** depicts the improvement of solutions of LN-2D plus LNS-3D algorithm over time. The algorithm exhibits a steep ascent, followed by a more moderate increase, and then another steep ascent. Note that Will's `HPstruct` program relies heavily on the energy model and its use of precomputed hydrophobic cores, thus perhaps making it difficult to generalize to other energy models. In contrast, our algorithm could be modified to handle dihedral angles and a different energy model. Indeed, preliminary results indicate that our method can be applied to problems such as RNA structure prediction with certain modifications.

7 Conclusions and Future Work

This paper presented variants of a LS+LNS algorithm for finding high-quality self avoiding walks for the Hydrophobic-Polar (HP) energy model on the Face Centered Cubic (FCC) lattice. The algorithm relies on a local search initial solution which is then improved by a constraint-programming LNS strategy. Benchmarking studies[7] show the value of our approach, compared with the hydrophobic core threading approach of S. Will. For problem instances for which a suitable, precomputed hydrophobic core exists and for which Will's algorithm converges within a required time, Will's method is clearly optimal. In contrast, our method immediately furnishes useful approximations by local search, which then improve with additional computational time by the repeated application of large neighborhood search. Results of this paper show that the hybridization of local search and constraint programming has great potential for application to the highly combinatorial problem of structure prediction, in a manner that can be viewed as complementary to hydrophobic core threading over precomputed cores.

The goal of this paper is to apply constraint programming (CP) to compute approximations for solutions to instances of the problem of protein structure prediction for the HP-

7. Full sequence information and predicted structures for the randomizations obtained using the Altschul-Erikson algorithm [**?**], [**?**] can be found at http://bioinformatics.bc.edu/clotelab/FCCproteinStructure/.

Seq.	Native E.	LNS-MULT	LNS-3D	Time
H1	69	***69** (66.77)	***69** (67.68)	75 s + 10 m
H2	69	***69** (66.60)	***69** (66.73)	75 s + 10 m
H3	72	***72** (68.02)	71 (68.06)	75 s + 10 m
H4	71	***71** (67.31)	***71** (67.61)	75 s + 10 m
H5	70	***70** (66.98)	***70** (67.04)	75 s + 10 m
H6	70	***70** (67.49)	***70** (67.43)	75 s + 10 m
H7	70	***70** (66.55)	69 (66.68)	75 s + 10 m
H8	69	***69** (65.80)	***69** (65.81)	75 s + 10 m
H9	71	***71** (67.95)	***71** (67.92)	75 s + 10 m
H10	68	***68** (65.76)	***68** (65.67)	75 s + 10 m
S1	357	349 (332.37)	***351** (336.74)	300 s + 30 m
S2	360	349 (328.98)	***353** (334.17)	300 s + 30 m
S3	367	351 (323.77)	***353** (329.80)	300 s + 30 m
S4	370	346 (323.98)	***354** (334.22)	300 s + 30 m
R1	384	313 (287.98)	***330** (305.54)	500 s + 30 m
R2	383	331 (289.83)	***333** (308.31)	500 s + 30 m
R3	385	325 (288.49)	***334** (307.76)	500 s + 30 m
F90_1	168	164 (156.83)	***165** (157.39)	180 s + 30 m
F90_2	168	***163** (155.05)	***163** (155.81)	180 s + 30 m
F90_3	167	***163** (156.23)	***163** (157.20)	180 s + 30 m
F90_4	168	***164** (156.20)	163 (156.54)	180 s + 30 m
F90_5	167	163 (155.77)	***164** (157.46)	180 s + 30 m
F180_1	?	289 (264.06)	***293** (269.07)	300 s + 30 m
F180_2	?	302 (280.84)	***312** (287.21)	300 s + 30 m
F180_3	378	306 (286.78)	***313** (295.31)	300 s + 30 m

TABLE 2: Greatest number of contacts found for each Hybrid Large Neighborhood Search based algorithm showing the average contacts over 100 runs in parenthesis. Boldface indicates the highest value found. Asterisks indicate highest number of contacts overall. Native E. is the optimal number of contacts, LNS-MULT is Multiple Sequence Reoptimized LNS and LNS-3D is 3D Structure Reoptimized LNS. Computation times are as follows: *(i)* Harvard instances: 75 sec. LS + 10 min. LNS; *(ii)* S: 300 sec. + 30 min. of LNS; *(iii)* R: 500 sec. + 30 min. LNS; *(iv)* F90: 180 sec. LS + 30 min. LNS; *(v)* F180: 300 sec. + 30 min. LNS.

model on the FCC lattice.[8] Experimental results are meant only to benchmark the LNS algorithm. Our long term interest is the application of local search and CP to real biomolecular structure prediction. Bradley et al. [**?**] argue that protein structure prediction consists of two aspects: (1) a good search strategy (2) adequate fragment library. Skolnick and others have argued that due to the Structural Genome Initiative (high-throughput X-ray diffraction studies of proteins having less than 30% homology to any existent proteins), the fragment library is essentially currently adequate. While most search strategies (including that of Bradley, Misura and Baker) are Monte Carle (possibly with simulated annealing, possibly with replica exchange), our goal is to develop algorithms such as LNS that ultimately will play a role in biomolecular structure prediction. This is the long-term justification of the current work.

Our current work in progress explores more complex energy models and off-lattice setups. Preliminary results show that changing the energy (i.e., adding weights to contacts) can be achieved with minimal modification and with similar performance. The algorithm can be adapted to RNA structure prediction, which we are currently exploring and validating from a biological standpoint.

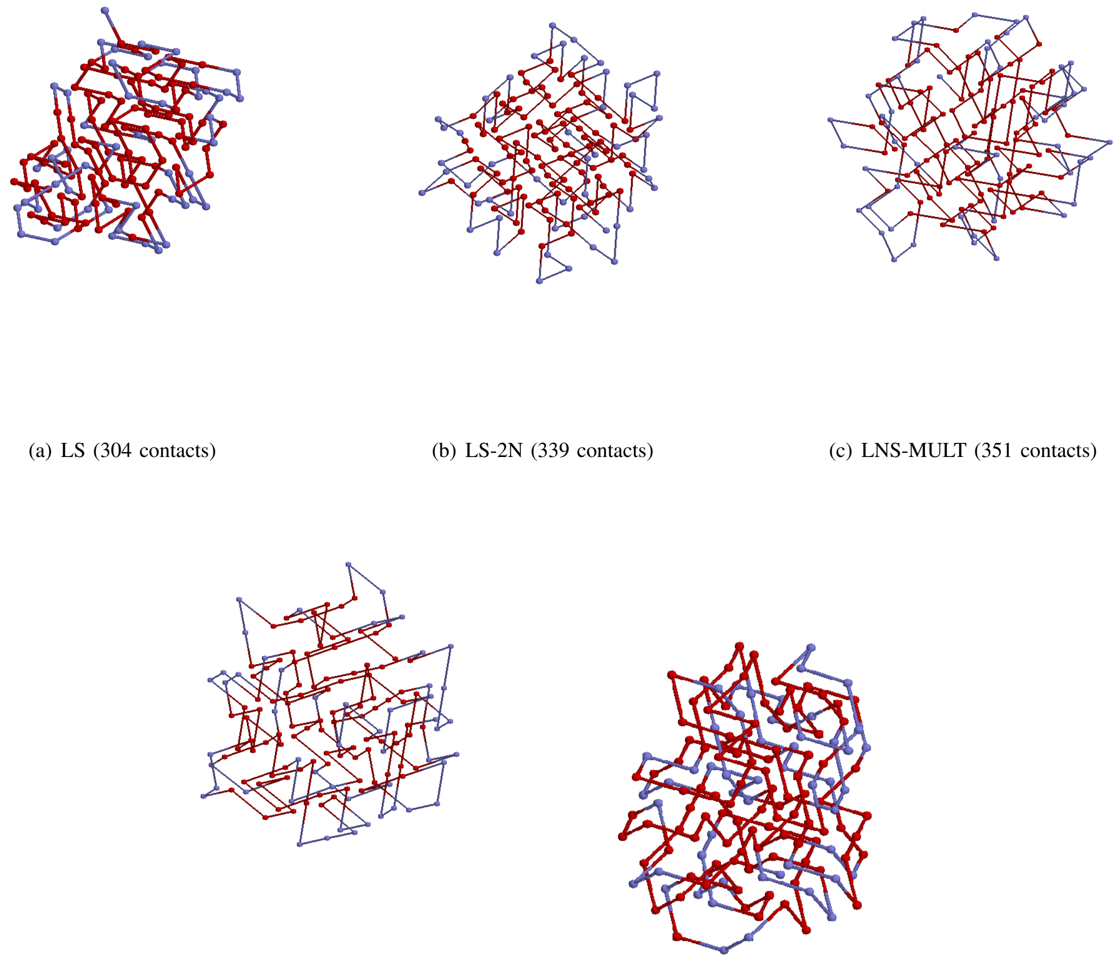
8. Structure prediction for the HP-model on the cubic lattice is known to be NP-complete [**?**], although this appears to be yet unproved for the FCC lattice.

Seq.	Num seq	Len	Failure rate
F90	50	90	10%
F90-doubled	50	180	78%
F180	30	180	50%

TABLE 3: Nonconvergence rate for Will's hydrophobic core threading algorithm, for randomizations of the F90 and F180 instances with a run time bound of 30 minutes. To produce longer instances, we doubled the length of the F90 instances by concatenating a copy of the same sequence. For each of 5 F90, 5 F90-doubled, and 3 F180 sequences, we produced 10 randomized sequences having the *same* diresidues, by running our implementation of the Altschul-Erikson diresidue shuffle algorithm [**?**] using our code described in [**?**]. For each instance class, the table displays the number of sequences tested, their length and the failure rate of `HPstruct` with 30 minute time bound. Tests were performed on a cluster of Dell Power Edge 1950 4-core Intel E5430 processors with 2.66 GHz and 16 Gb of RAM. (PBS/Torque was used to batch the runs; however, no algorithmic parallelism was used, and each run used only one core.) Number of H-H contacts obtained by our method within 30 minute time bound can be found at our web site, cited in Section **??**.

ACKNOWLEDGEMENTS

Sebastian Will's publicly available web server http://cpsp.informatik.uni-freiburg.de:8080/StructJSP.jsp was used to determine the optimal number of H-H contacts for the Harvard instances H1-H10. We would like to thank Sebastian Will for providing us with additional problem instances S, R, $F90$, and $F180$, and especially for providing us precomputed hydrophobic cores and the executable code for his program `HPstruct` that allowed us to perform benchmarking stud-



(a) LS (304 contacts) (b) LS-2N (339 contacts) (c) LNS-MULT (351 contacts)

(d) LNS-3D (353 contacts) (e) Native Configuration (360 contacts)

Fig. 6: Lowest Energy Configurations for Instance S2. Native is the Optimal Configuration.

ies. Sequence data used in the benchmarking tests is available at http://bioinformatics.bc.edu/clotelab/SUPPLEMENTS/HPsequencesForProteinsPaper/. Finally, we would like to thank the reviewers for their helpful comments.

Ivan Dotú is supported by a "Fundacion Caja Madrid" grant and Manuel Cebrián by grant TSI 2005-08255-C07-06 of the Spanish Ministry of Education and Science. Peter Clote is partially supported by a Chaire d'Excellence from Digiteo Foundation and by NSF grants DBI-0543506 and DMS-0817971. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] R.A. Abagyan, M.M. Totrov, and D.A. Kuznetsov. ICM: a new method for structure modeling and design: Applications to docking and structure prediction from the distorted native conformation. *J. Comp. Chem.*, 15:488–506, 1994.

[2] S.F. Altschul and B.W. Erikson. Significance of nucleotide sequence alignments: A method for random sequence permutation that preserves dinucleotide and codon usage. *Mol. Biol. Evol*, 2(6):526–538, 1985.

[3] C. B. Anfinsen. Principles that govern the folding of protein chains. *Science*, 181:223–230, 1973.

[4] K. Arnold, L. Bordoli, J. Kopp, and T. Schwede. The SWISS-MODEL workspace: a web-based environment for protein structure homology modelling. *Bioinformatics*, 22(2):195–201, January 2006.

[5] R. Backofen. The protein structure prediction problem: A constraint optimization approach using a new lower bound. *Constraints*, 6(2-3):223–255, 2001.

[6] R. Backofen, S. Will, and E. Bornberg-Bauer. Application of constraint programming techniques for structure prediction of lattice proteins with extended alphabets. *Bioinformatics*, 15(3):234–242, March 1999.

[7] R. Backofen, S. Will, and P. Clote. Algorithmic approach to quantifying the hydrophobic force contribution in protein folding. *Pacific Symposium on Biocomputing*, 5:92–103, 2000.

[8] Rolf Backofen. Using constraint programming for lattice protein folding. In *Workshop on Constraints and Bioinformatics/Biocomputing*, 1997. Held in conjunction with *Third International Conference on Principles and Practice of Constraint Programming (CP97)*.

[9] Rolf Backofen and Sebastian Will. A constraint-based approach to structure prediction for simplified protein models that outperforms other

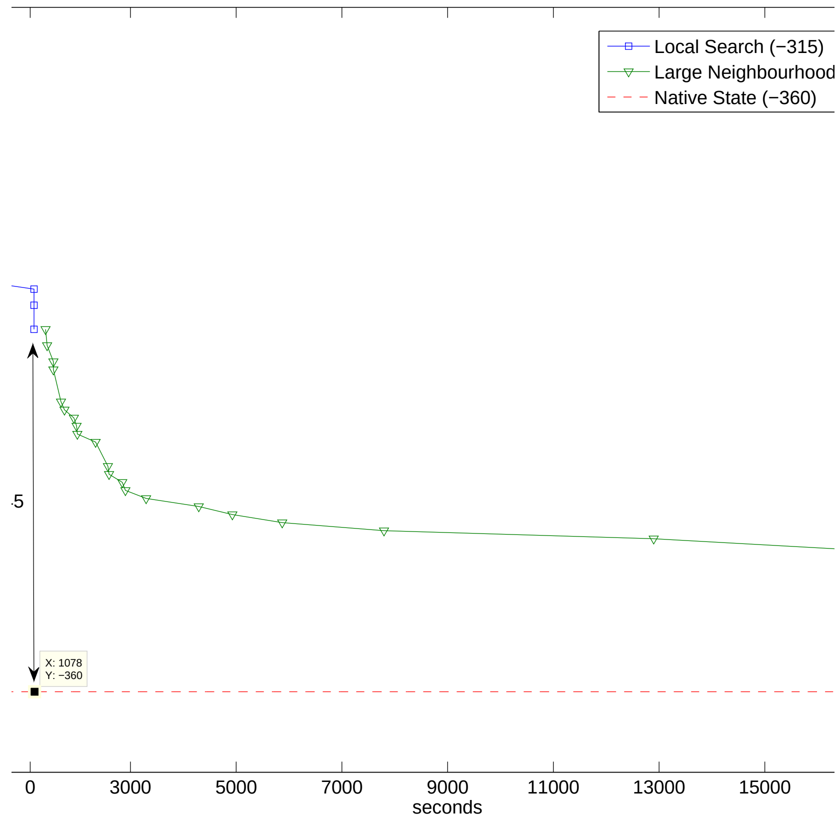


Fig. 7: Execution trace of LN-2D (20,000 iterations) plus LNS-3D (2 hours) on Will's instance S2.

existing methods. In *Proceedings of the 19th International Conference on Logic Programming (ICLP 2003)*, pages 49–71, 2003.

[10] B. Berger and T. Leighton. Protein folding in the hydrophobic-hydrophilic (hp) model is NP-complete. *Journal of Computational Biology*, 5:27–40, 1998.

[11] H. M. Berman, T. Battistuz, T. N. Bhat, W. F. Bluhm, P. E. Bourne, K. Burkhardt, Z. Feng, G. L. Gilliland, L. Iype, S. Jain, P. Fagan, J. Marvin, D. Padilla, V. Ravichandran, B. Schneider, N. Thanki, H. Weissig, J. D. Westbrook, and C. Zardecki. The Protein Data Bank. *Acta Crystallogr. D. Biol. Crystallogr.*, 58(Pt):899–907, June 2002.

[12] H. J. Bernstein. Recent changes to RasMol, recombining the variants. *Trends Biochem. Sci.*, 25(9):453–455, September 2000.

[13] E. Bornberg-Bauer. Chain growth algorithms for HP-type lattice proteins. In *RECOMB*, pages 47–55. ACM Press, 1997.

[14] P. Bradley, K. M. Misura, and D. Baker. Toward high-resolution de novo structure prediction for small proteins. *Science*, 309(5742):1868–1871, September 2005.

[15] B.R. Brooks, R.E. Bruccoleri, B.D. Olafson, D.J. States, S. Swaminathan, and M. Karplus. CHARMM: A program for macromolecular energy, minimization, and dynamics calculations. *J. Comput. Chem.*, 4:187–217, 1983.

[16] B. Cipra. Packing challenge mastered at last. *Science*, 281:1267, 1998.

[17] P. Clote, F. Ferré, E. Kranakis, and D. Krizanc. Structural RNA has lower folding energy than random RNA of the same dinucleotide frequency. *RNA*, 11(5):578–591, 2005.

[18] A. Condon. Using constraint programming and local search methods to solve vehicle routing problems. *CP'98*, pages 565–575, 1998.

[19] J.H. Conway and N.J.A. Sloane. *Sphere Packing, Lattices and Groups*. Springer-Verlag, 1998. Third edition.

[20] D.G. Covell and R.L. Jernigan. Conformations of folded proteins in restricted spaces. *Biochemistry*, 27:3287–3294, 1990.

[21] P. Crescenzi, D. Goldman, C. Papadimitriou, A. Piccolboni, and M. Yannakakis. On the complexity of protein folding. *J. Comp. Biol.*, 5(3):523–466, 1998.

[22] A. Dal Palu, A. Dovier, and F. Fogolari. Constraint Logic Programming approach to protein structure prediction. *BMC. Bioinformatics*, 5:186, November 2004.

[23] J. A. Dalton and R. M. Jackson. An evaluation of automated homology modelling methods at low target template sequence similarity. *Bioinformatics*, 23(15):1901–1908, August 2007.

[24] F. Ding, J. M. Borreguero, S. V. Buldyrey, H. E. Stanley, and N. V. Dokholyan. Mechanism for the alpha-helix to beta-hairpin transition. *Proteins.*, 53(2):220–228, November 2003.

[25] I. Dotu, M. Cebrian, P. Van Hentenryck, and Peter Clote. Protein structure prediction with large neighborhood constraint programming search. In P.J. Stuckey, editor, *International Conference on Principles and Practice of Constraint Programming (CP'08)*, volume 5202, pages 82–96. Springer Lecture Notes in Computer Science, 2008.

[26] I. Dotú, M. Cebrián, P. van Hentenryck, and P. Clote. A local search approach to protein structure prediction on the face centered cubic lattice. In *Twenty-Third Conference of the Association for the Advancement of Artificial Intelligence (AAAI-08)*, Jul 13-17 2008.

[27] Y. Duan, C. Wu, S. Chowdhury, M. C. Lee, G. Xiong, W. Zhang, R. Yang, P. Cieplak, R. Luo, T. Lee, J. Caldwell, J. Wang, and P. Kollman. A point-charge force field for molecular mechanics simulations of proteins based on condensed-phase quantum mechanical calculations. *J. Comput. Chem.*, 24(16):1999–2012, December 2003.

[28] M. S. Evans, I. M. Sander, and P. L. Clark. Cotranslational folding

promotes beta-helix formation and avoids aggregation in vivo. *J. Mol. Biol.*, 383(3):683–692, November 2008.
[29] F. Ferré and P. Clote. Disulfide connectivity prediction using secondary structure information and diresidue frequencies. *Bioinformatics*, 21(10):2336–2346, 2005.
[30] F. Ferre and P. Clote. Dianna 1.1: an extension of the DiANNA web server for ternary cysteine classification. *Nucleic. Acids. Res.*, 34(Web):W182–W185, July 2006.
[31] C. A. Floudas. Computational methods in protein structure prediction. *Biotechnol. Bioeng.*, 97(2):207–213, June 2007.
[32] G. Helles. A comparative study of the reported performance of ab initio protein structure prediction algorithms. *J. R. Soc. Interface*, 5(21):387–396, April 2008.
[33] P. Van Hentenryck and L. Michel. *Constraint-Based Local Search*. MIT Press, 2005.
[34] L. Holm and C. Sander. Database algorithm for generating protein backbone and side-chain co-ordinates from a C $alpha$ trace application to model building and detection of co-ordinate errors. *J. Mol. Biol.*, 218(1):183–194, March 1991.
[35] B. John and A. Sali. Comparative protein structure modeling by iterative alignment, model building and model assessment. *Nucleic. Acids. Res.*, 31(14):3982–3992, July 2003.
[36] A. H. Khodabakhshi, J. Manuch, A. Rafiey, and A. Gupta. Inverse protein folding in 3D hexagonal prism lattice under HPC model. *J. Comput. Biol.*, 16(6):769–802, June 2009.
[37] J.L. Klepeis and C.A. Floudas. Prediction of β-sheet topology and disulfide bridges in polypeptides. *Journal of Computational Chemistry*, 24(2):191–208, 2002.
[38] J. Kyte and R. F. Doolittle. A simple method for displaying the hydropathic character of a protein. *J. Mol. Biol.*, 157(1):105–132, May 1982.
[39] P. Y. Lam, P. K. Jadhav, C. J. Eyermann, C. N. Hodge, Y. Ru, L. T. Bacheler, J. L. Meek, M. J. Otto, M. M. Rayner, Y. N. Wong, et al. Rational design of potent, bioavailable, nonpeptide cyclic ureas as HIV protease inhibitors. *Science*, 263(5145):380–384, January 1994.
[40] R. H. Lathrop. The protein threading problem with sequence amino acid interaction preferences is NP-complete. *Protein. Eng.*, 7(9):1059–1068, September 1994.
[41] R.H. Lathrop and T.F. Smith. Global optimum protein threading with gapped alignment and empirical pair score functions. *J. Mol. Biol.*, 255(4):641–665, 1996.
[42] K.F. Lau and K. A. Dill. A lattice statistical mechanics model of the conformational and sequence spaces of proteins. *Journal of the American Chemical Society*, 22:3986–3997, 1989.
[43] B. Lee and F. M. Richards. The interpretation of protein structures: estimation of static accessibility. *J. Mol. Biol.*, 55(3):379–400, February 1971.
[44] N. Madras and G. Slade. *The Self-Avoiding Walk*. Birkh?user, Boston, 1996. Series: Probability and its Applications, 448 p., ISBN: 978-0-8176-3891-7.
[45] L. Michel, A. See, and P. Van Hentenryck. Parallelizing constraint programs transparently. In *CP'2007*, 2007.
[46] S. Miyazawa and R. L. Jernigan. Self-consistent estimation of inter-residue protein contact energies based on an equilibrium mixture approximation of residues. *Proteins.*, 34(1):49–68, January 1999.
[47] M. Paluszewski, T. Hamelryck, and P. Winter. Reconstructing protein structure from solvent exposure using tabu search. *Algorithms. Mol. Biol.*, 1:20, 2006.
[48] C. Papadimitriou. *Computational Complexity*. Addison Wesley, 1994.
[49] P. Pokarowski, A. Kloczkowski, R. L. Jernigan, N. S. Kothari, M. Pokarowska, and A. Kolinski. Inferring ideal amino acid interaction forms from statistical protein contact potentials. *Proteins.*, 59(1):49–57, April 2005.
[50] G. Pollastri, P. Baldi, P. Fariselli, and R. Casadio. Prediction of coordination number and relative solvent accessibility in proteins. *Proteins.*, 47(2):142–153, May 2002.
[51] A. Šali, E. Shakhnovich, and M. Karplus. How does a protein fold? *Nature*, 369:248–251, May 1994.
[52] A. Šali, E. Shakhnovich, and M. Karplus. Kinetics of protein folding: A lattice model study of the requirements for folding to the native state. *Journal of Molecular Biology*, 235:1614–1636, 1994.
[53] N. Siew and D. Fischer. Convergent evolution of protein structure prediction and computer chess tournaments: CASP, Kasparov, and CAFASP. *IBM Systems Journal*, 40(2):410–425, 2001.
[54] M. Sippl. Calculation of conformation ensembles from potentials of mean force. *J. Mol. Biol.*, 213:859–883, 1990.
[55] J. Skolnick and A. Kolinski. Simulations of the Folding of a Globular Protein. *Science*, 250(4984):1121–1125, November 1990.
[56] R. Unger and J. Moult. Genetic algorithms for protein folding simulations. *Journal of Molecular Biology*, 231:75–81, 1993.
[57] S. Will. Constraint-based hydrophobic core construction for protein structure prediction in the face-centered-cubic lattice. In Russ B. Altman, A. Keith Dunker, Lawrence Hunter, and Teri E. Klein, editors, *Pacific Symposium on Biocomputing*, volume 7, pages 661–672, 2002. World Scientific Publishing Co., Singapore.
[58] Sebastian Will. *Exact, Constraint-Based Structure Prediction in Simple Protein Models*. PhD thesis, Friedrich-Schiller-Universität Jena, April 2005.
[59] S. Wu, J. Skolnick, and Y. Zhang. Ab initio modeling of small proteins by iterative TASSER simulations. *BMC. Biol.*, 5:17, 2007.
[60] K. Yue and K. A. Dill. Sequence-structure relationships in proteins and copolymers. *Phys. Rev. E*, 48(3):2267–2278, September 1993.
[61] K. Yue and K. A. Dill. Folding proteins with a simple energy function and extensive conformational searching. *Protein. Sci.*, 5(2):254–261, February 1996.
[62] K. Yue, K. M. Fiebig, P. D. Thomas, H. S. Chan, E. I. Shakhnovich, and K. A. Dill. A test of lattice protein folding algorithms. *Proc. Natl. Acad. Sci. U.S.A.*, 92(1):325–329, January 1995.
[63] K. Yue, K.M. Fiebig, P.D. Thomas, H.S. Chan, E.I. Shakhinovich, and K.A. Dill. A test of lattice protein folding algorithms. *Proc. Natl. Acad. Sci. USA*, 92:325–329, 1995.
[64] M.J. Zaki. *Protein Structure Prediction*. Humana Press, 2007. second edition.
[65] Y. Zhang. I-TASSER server for protein 3D structure prediction. *BMC. Bioinformatics*, 9:40, 2008.
[66] Y. Zhang and J. Skolnick. The protein structure prediction problem could be solved using the current PDB library. *Proc. Natl. Acad. Sci. U.S.A.*, 102(4):1029–1034, January 2005.