Ann Oper Res
DOI 10.1007/s10479-009-0666-5

A filter-and-fan approach to the 2D HP model of the protein folding problem

César Rego · Haitao Li · Fred Glover

Abstract We examine a prominent and widely-studied model of the protein folding problem, the two-dimensional (2D) HP model, by means of a filter-and-fan (F&F) solution approach. Our method is designed to generate compound moves that explore the solution space in a dynamic and adaptive fashion. Computational results for standard sets of benchmark problems show that the F&F algorithm is highly competitive with the current leading algorithms, requiring only a single solution trial to obtain best known solutions to all problems tested, in contrast to a hundred or more trials required in the typical case to evaluate the performance of the best of the alternative methods.

Keywords Metaheuristics · Tabu search · Compound neighborhoods · Filter-and-fan · Protein folding · Bioinformatics

1 Introduction

A protein is a linear polymer molecule that may consist of thousands of monomer units. The monomers are the 20 known amino acids. The typical length of a protein chain ranges from 50 to 1000 amino acids. In the natural environment a protein adopts a specific *tertiary structure* called a *conformation*. There are essentially three types of proteins existing in nature: fibrous, membrane and globular. Most studies on protein structure prediction have

C. Rego (✉)
School of Business Administration, University of Mississippi, Oxford, MS 38677, USA
e-mail: crego@bus.olemiss.edu

H. Li
College of Business Administration, University of Missouri–St. Louis, One University Blvd., St., Louis, MO 63121-4400, USA
e-mail: lihait@umsl.edu

F. Glover
University of Colorado, Boulder, CO 80309-0419, USA
e-mail: fred.glover@colorado.edu

Published online: 30 October 2009

focused on globular proteins due to the fact that, unlike the other two, a globular protein exhibits an extremely compact and unique conformation in its native state (Chan and Dill 1993). The native state of a protein is often thermodynamically stable, which means that it is associated with a global minimal Gibbs free energy. Also, it is well established that the amino acid sequence is a major factor in determining the folding conformation of a protein (Anfinsen et al. 1961). Other factors such as enzymes catalyze the folding process but do not play a determinant role as the amino acid sequence does in yielding the structure ultimately attained (Richards 1991).

The Protein Folding Problem (PFP) is the problem of predicting the three-dimensional (3D) structure of a protein given only the protein's sequence of amino acids. This is a fundamental yet open problem in the fields of biological chemistry and protein science, and has attracted attention in the areas of bioinformatics and computational biology. Understanding and predicting the native conformation of a protein is of both theoretical and practical importance. A protein's function is closely related to its 3D structure, and therefore one can determine how a protein functions by predicting its 3D conformation. Now that the Human Genome Project has mapped the complete human genome sequence (represented by over 3 billion DNA subunits), advanced bio-computing models and techniques are crucial for interpreting human gene functions and so proteins. The PFP is central in a number of practical applications including the designing of new proteins having desirable functions in pharmaceutical, food, and agriculture industry (Lengauer 1993). We refer to Richards (1991) and Chan and Dill (1993) for an overview of the PFP and its applications.

The PFP is a notoriously difficult combinatorial problem due to the combinatorial explosion of valid conformations as the number of amino acids in the chain increases. It has been estimated that using current computer technologies, finding all possible conformations for a 100-aminoacid protein would require 10^{27} years of computer processing (Krasnogor et al. 1998), a length of time that is more than a thousand billion times the age of the universe. Notably the PFP has been identified as a National Grand Challenge in biochemistry (Committee on Physical, Mathematical and Engineering Science 1992) in the U.S. Even in its simplified version of the lattice HP model (Dill 1985), the PFP has been proved to be NP-complete (Crescenzi et al. 1998) and therefore no polynomial algorithm is available for its solution (and theoretically none can exist unless $P = NP$).

Two experimental methods of determining the protein structure are X-ray crystallography and Nuclear Magnetic Resonance (NMR). Most of the structural information in the Protein Data Bank (PDB) is obtained through these two approaches (Berman et al. 2000). A drawback of these methods is that they are exceedingly expensive in their demands on equipment and processing time. Bioinformatics supported by advanced optimization models and techniques provides an alternative approach with the potential to predict the native structure of proteins in a much more cost-effective manner. Since current exact solution methods are unable to solve even the smallest instances of the PDB, effective heuristic (approximation) algorithms are required to find optimal or near-optimal conformations. Also, due to the complex nature of the PFP, the so-called HP lattice model proposed by Dill (1985) constitutes a well established simplification for algorithm assessment.

This paper considers the two-dimensional (2D) version of the HP lattice model and proposes a new algorithm for the solution of the associated PFP. The key contribution is the development of an effective mechanism for adapting simple neighborhood structures to enable them to explore the solution space more effectively. The approach is based on a filter-and-fan procedure that augments a simple neighborhood in a dynamic and adaptive fashion using a truncated form of tree search. Computational testing carried out on the standard benchmarks and Functional Model Proteins (FMP) testbeds demonstrates the competitiveness of the proposed filter-and-fan algorithm relative to alternative approaches in the literature. The remainder of the paper is organized as follows. Section 2 introduces the 2D HP lattice model and Sect. 3 develops the filter-and-fan algorithm. Computational results are presented in Sect. 4, followed by a summary of conclusions and a discussion of promising avenues for future research in Sect. 5.

2 The 2D HP model

The HP lattice model (Dill 1985) is the most widely studied protein folding model. Although relatively simple the model captures many global aspects of protein structure (Chan and Dill 1993). The lattice serves as a tool to discretize the conformational space in real-world protein folding, while allowing freedom of the chain to have different backbone conformations. As discussed by Lau and Dill (1989), the lattice model provides a way to explore the nature of the full conformational and sequence spaces of proteins. The model additionally helps to answer fundamental questions about the conformation space of real proteins such as its (landscape) structure, the density of native states with lowest energy, the local minima distribution and their correlation to reaching a compact globular state with a hydrophobic core. Under the assumption that the hydrophobic interaction is the dominant force in protein folding, the model considers two types of amino acids: hydrophobic monomers (denoted by H), which are oil-like and interact poorly with water, and hydrophilic (or *polar*, denoted by P), which interact favorably with water. A sequence of H and P amino acids is configured as a path on a two-dimensional (2D) square lattice to define a valid *conformation*. The path designation implies that the conformation is both connected and self-avoiding, i.e., no amino acids can collide in the same cell of the lattice. (In graph theory terminology, such a path is called *node simple*.) The energy function is defined by the number of pairs of H nodes that are *adjacent* in the lattice and not *consecutive* in the chain. Each of these pairs, generally called an H–H contact, decreases the energy value by one unit and all the other contacts have zero contribution. The energy interaction in the Functional Model Proteins (FMP, Hirst 1999) while considering repulsive forces is different: an H–H contact has an energy contribution of -2 and all the other contacts contribute $+1$ to the energy value to be minimized. The objective is to find a conformation that minimizes the total energy of the given amino acid sequence, which therefore corresponds to maximizing the number of H–H contacts.

Figure 1 illustrates a protein with seven hydrophobic (H) and two hydrophilic (P) monomers represented by black and white nodes, respectively. Nodes 1 and 9 denote the two terminal nodes in the chain.

In the native state of globular proteins, the polar monomers tend to reside on the surface attached to water molecules, while the lack of favorable interaction between polar residues and solvent (water) drives the hydrophobic residues into the core.

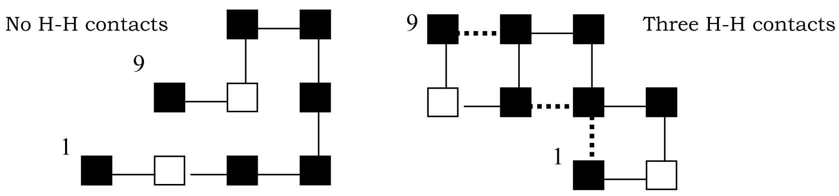

Fig. 1 An HP lattice model of protein with 7 hydrophobic (H) and 2 hydrophilic (P) monomers

Various computational approaches have been proposed to tackle this simplified yet NP-complete combinatorial problem. Notable approximation algorithms include chain growth algorithms by Hart and Istrail (1996, 1997) and Bornberg-Bauer (1997). Better computational results have been found by metaheuristic algorithms. Evolutionary algorithms include Genetic Algorithms (GA) by Unger and Moult (1993), Dandekar and Argos (1994), Krasnogor et al. (1999), and Konig and Dandekar (1999), Memetic Algorithm by Krasnogor et al. (2002) and Pelta and Krasnogor (2004), and Immune Algorithm by Cutello et al. (2005, 2006). Classical Monte Carlo Simulation (MC) approaches were first developed by Covell and Jernigan (1990) and Skolnick and Kolinski (1990). More recently, variants of the MC approach have been introduced in the Pruned Enriched Rosenbluth Method (PERM) by Grassberger (1997), and Hsu et al. (2003a, 2003b), the dynamic Monte Carlo algorithm by Ramakrishnan et al. (1997), the evolutionary Monte Carlo (EMC) algorithm by Liang and Wong (2001) and the Multi-Self-Overlap-Ensemble (MSOE) by Chikenji et al. (1999). Lesh et al. (2003) apply a randomized multi-start Tabu Search algorithm that obtains several best results for the standard 2D HP benchmark problems. A hybrid approach combining tabu search and genetic algorithm is proposed by Jiang et al. (2003). Shmygelska et al. (2002) Shmygelska and Hoos (2003, 2005) propose an Ant Colony Optimization algorithm and report satisfying results. Other approaches include Constraint Programming (CP) by Backofen (2001).

3 The filter-and-fan algorithm

Generally speaking, the success of a metaheuristic algorithm is driven by two fundamental components, the neighborhood structure used for the local search and the strategy to guide the search beyond local optimality. Depending on the size and difficulty of the problem, different levels of sophistication may be required for each of these components. In general, large scale problems require highly effective neighborhood structures to generate moves that are capable of exploring large neighborhood spaces. A special class of such neighborhoods is represented by the so-called ejection chain methods, which are chiefly designed to explore exponential neighborhoods in polynomial time (see Glover 1992, 1996a, 1996b, and Rego and Glover 2002). Although very powerful, ejection chain methods are typically more complex than other more traditional neighborhood designs. In cases where ejection chain approaches are not available or the problem size is not exceedingly large, simpler neighborhoods accompanied by appropriate search guidance may be adequate. We pursue the approach of seeking an effective guidance strategy within a simpler neighborhood by extending the so-called pull-move neighborhood (Lesh et al. 2003) by means of a filter-and-fan method. The idea of pull-moves also shares some similarities with the work of Nunes et al. (1996). To elaborate the algorithm we first describe the pull-move neighborhood structure.

3.1 The pull-move neighborhood

A filter and fan algorithm requires the definition of component moves used to generate trial solutions throughout the search process. Component moves are characteristically simple moves serving as building blocks for the construction of an extended filter and fan neighborhood. We make use of component moves defined by the pull-move neighborhood introduced in the tabu search implementation of Lesh et al. (2003). A pull-move is initiated by moving one node of the current conformation to its empty diagonal adjacent position in the square induced by the node and its neighbor(s) in the sequence. Depending on the structure of the

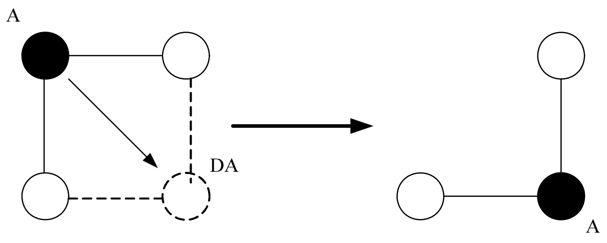

Fig. 2 Pull move—*filling*

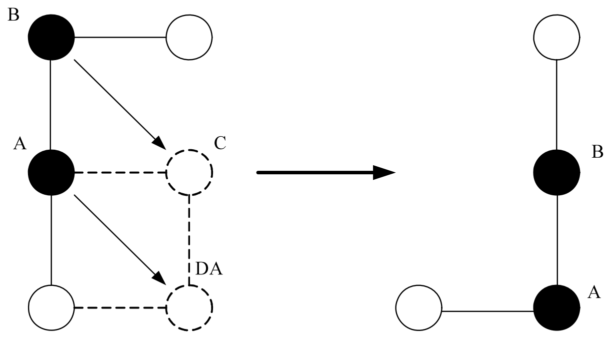

Fig. 3 Pull move—*single-pull*

conformation the displacement of the initiating node may require other nodes to change their current positions in order to preserve connectivity. In a pull-move, displaced nodes are only allowed to occupy vacant adjacent positions in the lattice. Consequently, the preservation of connectivity also results in a self-avoiding path. We differentiate only three types of pull-moves designated by *filling*, *single-pull* and *multiple-pull,* according to the number of nodes that are pulled by the first displaced node. Figures 2, 3 and 4 provide an illustration of each type of these moves and associated reference structures. To simplify the notation in the following, we allow labels be attached to the nodes that denote both the name of the node and its position in the lattice.

The *filling* move is the simplest pull-move, displacing a single node in the structure. As shown in Fig. 2, a valid conformation is obtained by simply moving node A to its diagonal adjacent position (DA). The *filling* move is also known as the "corner move" (Socci and Onuchic 1994).

A *single-pull* requires the succeeding node move to the remaining corner of the square after the initial move. Consider the conformation shown on left side of Fig. 3. Moving node A to its adjacent diagonal position DA disconnects node B from node A as they are no longer adjacent nodes in the lattice. However, a connected conformation can be obtained by moving node B to position C, the remaining position in the square identified by the dotted lines.

The *multiple-pull* move extends the pull-move to achieve connectivity in more complex structures that become disconnected upon performing a single-pull move. An illustration of such type of structure is provided in Fig. 4.

The single-pull move represented by moving A to position DA and B to position D disconnects B from C. However, connectivity can be established by moving C to fill the position vacated by A. If the latter move brings about a new disconnected structure the process is repeated until connectivity is achieved. A universal rule to implement a multiple-pull consists of shifting successive nodes two positions ahead along the original chain until connectivity is reached. In the example, node C (i.e. the successor of B) is shifted through position B and

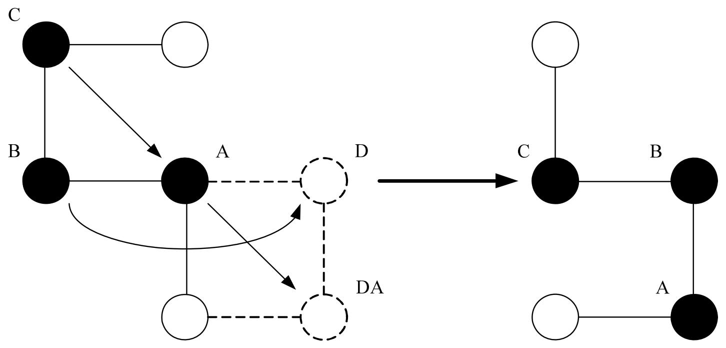

Fig. 4 Pull move—*multiple-pull*

then A in the original chain. Since at this point the structure is connected the process stops and the move is completed.

It should be noted that the P nodes (shown in white) in the preceding illustrations could just as well be H nodes. It may also be observed that even though the single-pull move of Fig. 3 can be performed as a succession of two filling moves, this is not always the case for general structures. Likewise, a single-pull move is not a special case of a multiple-pull move. As it turns out, generalizations of these three types of moves, and of combinations of them, are automatically generated by the method we describe next.

3.2 The filter and fan model and application

The filter and fan (F&F) method was initially proposed in Glover (1998) as a method for refining solutions obtained by scatter search, and was further extended in Rego and Glover (2002). In the latter, the method is proposed as an alternative to ejection chain methods and as a means for creating combined neighborhood search strategies. Conceptually, it integrates the *filtration* and the *sequential fan* candidate list strategies used in tabu search (Glover and Laguna 1997), and can be viewed as a restrictive form of tabu search that generates multiple paths in a breadth-first search strategy. From a neighborhood search perspective, the method generates compound moves as a sequence of more elementary *component moves* (or submoves).

Graphically, the F&F model can be illustrated by means of a neighborhood tree where branches represent submoves and nodes identify solutions produced by these moves. An exception is made for the root node, which represents the starting solution to which compound moves are to be applied. The maximum number of levels L permitted in a single sequence of moves defines the depth of the tree. The neighborhood tree is explored breadth-first, level by level. Each level is governed by the *filter candidate list* strategy that selects a subset of moves induced by the *fan candidate list* strategy. The method incorporates two fundamental components: a *local search* to identify a local optimum and a *filter and fan search* to explore larger neighborhoods in order to overcome local optimality. Any time a new local optimum is found in one search strategy the method switches to the other strategy and keeps alternating this way until the filter and fan search fails to improve the current best solution.

More advanced versions allow for the combination of different types of neighborhood structures and the use of adaptive memory programming as introduced in tabu search. For a detailed description of the method and the use of advanced strategies we refer to Glover (1998) and Rego and Glover (2002). The filter and fan approach used in this study employs component moves identified by the pull-move neighborhood described in the previous section. This neighborhood is likewise used in the local search phase. The algorithm is equipped with a tabu search short-term memory aimed at enhancing the local search and providing the tree search with appropriate legitimacy restrictions to drive the search and keep the method from generating duplicated conformations.

3.2.1 The filter-and-fan construction

Our F&F search can be sketched as follows. Once a locally optimal solution X_o is found (in the local search phase) the best η_1 currently available moves (among the moves evaluated to establish local optimality) are used to create the level 1 of the F&F neighborhood tree. As a basis for creating the next levels, for a given level indexed by k, η_1 denotes the number of solutions that are chosen from all solutions available at level k, as a foundation for generating solutions at level $k+1$. (For $k=1$, there are just η_1 solutions available, so all are chosen.) For each of these η_1 solutions, denoted $X_i(k)$ $(i=1,\dots,\eta_1)$, we apply η_2 moves to generate η_2 descendant solutions, thereby generating a total of $\eta=\eta_1\times\eta_2$ trial solutions for level $k+1$. At this stage, η_1 of the resulting η solutions are chosen to launch the process for the next level. The values η_1 and η_2 are input parameters, e.g. $\eta_1=2\eta_2$. (In more general versions of the approach, which we do not consider here, these parameters can vary adaptively from one level of search to the next.) If an improved solution (better than the local optimum X_o) is found among the trial solutions, then the method stops branching and switches back to the local search phase, taking this newly improved solution as a starting point. Otherwise, another selection takes place over the set of moves available and the process repeats for a maximum of L levels. We found empirically that the performance of the algorithm depends much more on the depth of the tree than its width, and that larger values of η_1 or η_2 not always impact solution quality while it usually increases the running times. Indeed, keeping η_1 and η_2 very small and increasing L to relatively large values according to the problem size turned out to be the best compromise between solution quality and running times. We consider $\eta_1=\eta_2=4$ for all problems tested and $L=10000$, 20000, and 30000 for chains up to 48, 60, and 100 amino acids, respectively.

The process of selecting η_2 moves has to obey a set of legitimacy restrictions that assure compatibility of the component moves used for the construction of a valid compound move. The *fan candidate list strategy* is embedded in the generation of the η trial solutions, whereas the selection of the η_1 best solutions from this collection constitutes the *filter candidate list strategy.* A tie-breaking procedure may be necessary to decide about solutions having the same objective function value. We have tested different randomized rules to generate variability in the tie-breaking process and found empirically that these choices did not impact the quality of the final solution nor were significant differences observed in the running times. Therefore, in our algorithm, ties are broken deterministically by selecting the first η_1 best solutions, among the array of solutions sorted in an ascending order of their energy value.

A filter and fan neighborhood extends and generalizes the pull move neighborhood in multiple senses. Since the outcome of a move chosen at one node of the tree is transmitted to subsequent nodes of the same branch, the method is *adaptive* in the sense that each pull-move is chosen according to the current state of the search. Also, the method is *dynamic* since the number of pull-moves used to compose a compound move depends on the level of the tree where the best trial move was found, which usually varies from one iteration to another, again depending on the state of the search. This produces a generalized form of a so-called variable-depth neighborhood. Another dynamic feature of the filter and fan method is its flexibility in varying the tree width and branch width throughout the search in the case where the values for η_1 and η_2 are changed from level to level. (In additional variants of the procedure, as when making use of constructive or destructive neighborhoods, a solution can refer to a partial solution, having some components undetermined. Local optimality is then defined in a special sense relative to the determined components, or by employing a default trial completion that fills in the values of the undetermined components.)

Figure 5 shows an example of the filter and fan neighborhood for a conformation of a protein with 20 amino acids in the 2D HP square lattice model, where $\eta_1 = \eta_2 = 4$ and $L = 5$.

A conformation of energy -6 (represented by the root node) denotes a local optimum determined by the local search phase. The first level of the filter and fan neighborhood is then generated by applying the $\eta_1 = 4$ best moves to the root conformation. (Note that at this stage, these best moves have already been identified by the local search method during the evaluation of the neighborhood that established the local optimum.) Due to local optimality, none of the conformations obtained in the first level improves the local conformation. The next level is created by applying the $\eta_2 = 4$ best pull-moves to each of the conformations in the current level, thus generating $\eta_1 \times \eta_2 = 16$ trial conformation from which a new set of $\eta_1 = 4$ best conformations is chosen to initiate the next level.

At each level of the tree, we sort the sixteen conformations in an ascending order of their energy value. That is, the conformation with lowest index has the best energy value. Then the first four conformations among the ordered conformations are selected to generate the next level and to break ties (if necessary). For instance, at the second level of the tree, there happens to be exactly four moves leading to energy of -6, thus no tie-breaking is needed. At the third level of the tree, however, only two out of sixteen conformations have energy -6, thus tie-breaking is needed to choose the additional two moves out of the remaining fourteen, all having energy -5. The two moves are the third and fourth in the deterministically sorted array of sixteen conformations at level three.

In the figure, the η_2 different conformations derived from the same parent conformation are contained within the rectangles delimited by solid lines whereas the η_1 best conformations selected at each level are contained within "interior rectangles" delimited by dotted lines. The method continues expanding the neighborhood until the improved conformation of energy -7 is found in level 5 of the filter and fan tree. The compound move leading to the improved conformation is then identified by the path indicated by the bold arrows.

3.3 The filter-and-fan implementation

3.3.1 Basic data structures

The design of appropriate data structures is crucial for the efficiency of the algorithm. In our approach, an *adjacency structure* is created to capture the relevant attributes of an amino acid in the conformation such as its ordering position (or index), its type (H or P), its Cartesian coordinates, and its neighboring amino acids. From this we create an *array of adjacency structures* to model the entire conformation.

Specifically, let an array A denote a protein consisting of a chain of amino acids $A[i]$ ($i = 1$ to $|C|$, where $|C|$ denotes the number of amino acids in the chain). We then model $A[i]$ by an *adjacency structure* consisting of a set of components that are relevant for representing the conformation of a protein, writing $A[i]$ as $A[i] = \{\mathit{index}(i), \mathit{type}(i), \mathit{coordinates}(i), \mathit{east}(i), \mathit{south}(i), \mathit{west}(i), \mathit{north}(i)\}$, where *index* provides the sequential order of an amino acid in a chain, *type* records the type of the corresponding

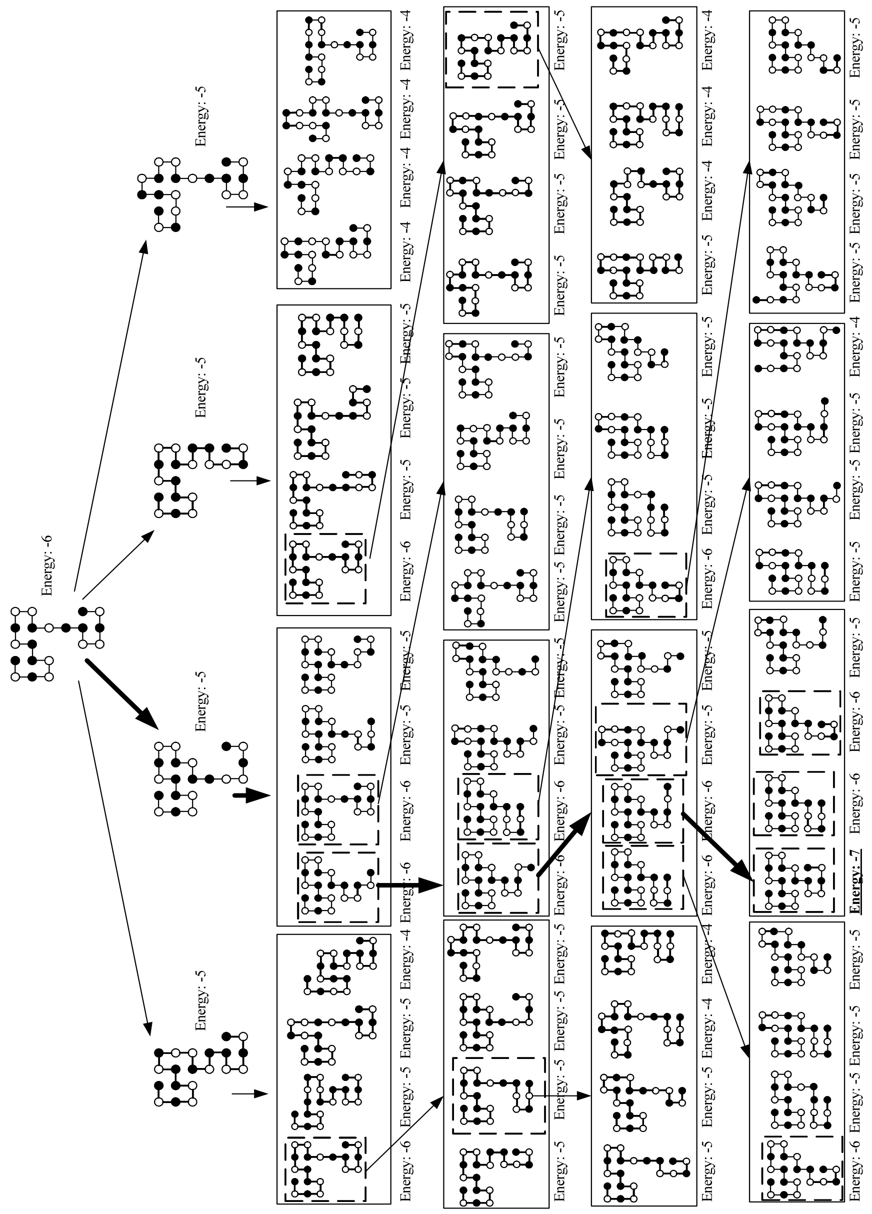
Energy: -6
Energy: -5
Energy: -5
Energy: -5
Energy: -5
Energy: -6 Energy: -5 Energy: -5 Energy: -4
Energy: -6 Energy: -6 Energy: -5 Energy: -5
Energy: -6 Energy: -5 Energy: -5 Energy: -5
Energy: -4 Energy: -4 Energy: -4 Energy: -4
Energy: -5 Energy: -5 Energy: -5 Energy: -5
Energy: -6 Energy: -6 Energy: -5 Energy: -5
Energy: -5 Energy: -5 Energy: -5 Energy: -5
Energy: -5 Energy: -5 Energy: -5 Energy: -5
Energy: -5 Energy: -5 Energy: -4 Energy: -4
Energy: -6 Energy: -6 Energy: -5 Energy: -5
Energy: -6 Energy: -5 Energy: -5 Energy: -5
Energy: -5 Energy: -4 Energy: -4 Energy: -4
Energy: -6 Energy: -5 Energy: -5 Energy: -5
Energy: -7 Energy: -6 Energy: -6 Energy: -5
Energy: -5 Energy: -5 Energy: -5 Energy: -4
Energy: -5 Energy: -5 Energy: -5 Energy: -5

Fig. 5 A Filter & Fan neighborhood for 2D HP model

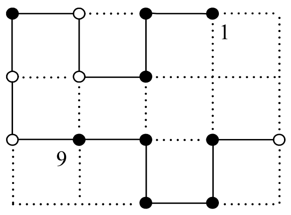

Fig. 6 Conformation of 14 amino acids with black nodes denoting 1 (H) and white nodes denoting 0 (P)

Table 1 The adjacency table representing the 14-node conformation in Fig. 6

Node			Neighboring node indexes			
Index	Type	Coordinates	East	South	West	North
1	1	$(0, 0)$	0	0	2	0
2	1	$(-1, 0)$	1	3	5	0
3	1	$(-1, -1)$	0	10	4	2
4	0	$(-2, -1)$	3	9	7	5
5	0	$(-2, 0)$	2	4	6	0
6	1	$(-3, 0)$	5	7	0	0
7	0	$(-3, -1)$	4	8	0	6
8	0	$(-3, -2)$	9	0	0	7
9	1	$(-2, -2)$	10	0	8	4
10	1	$(-1, -2)$	13	11	9	3
11	1	$(-1, -3)$	12	0	0	10
12	1	$(0, -3)$	0	0	11	13
13	1	$(0, -2)$	14	12	10	0
14	0	$(1, -2)$	0	0	13	0

amino acid with 1 referring to H and 0 referring to P, *coordinates* is the structure holding the Cartesian coordinates (x_i, y_i) indicating the position of an amino acid in a 2D lattice, and the remaining components record the indexes of the four neighboring amino acids.

We refer to the array A of adjacency structures as an *adjacency table*. Consider for illustration a protein consisting of 14 HP amino acids shown in Fig. 6 that has the following conformation.

The *adjacency table* representing the above conformation is shown in Table 1. Each row of the adjacency table is an *adjacency structure* representing one amino acid in the protein.

A zero index in the neighboring list indicates that the corresponding neighboring position is empty. The origin of the Cartesian coordinate system is fixed at the position of the first amino acid in the initial conformation. As new conformations are generated the adjacency structure is updated with the coordinates of the amino acids in the new conformation. It is important to notice that an absolute coordinate system is used so that only points that change their position need to be updated in the structure. However, to prevent the coordinate values from becoming exceedingly large, we set upper and lower bounds for x and y components that are given by the size of an integer data type for the computer in use. Hence, any time a pull-move results in taking an amino acid to a position that is out of the pre-defined range the entire conformation is shifted back to relocate node 1 at the origin point $(0, 0)$. In addition to specifying the coordinates of an amino acid it is important to identify its adjacent amino acids in the lattice. As explained later such adjacency information is fundamental for detecting valid pull moves as well as for their execution. Furthermore, the fact that the adjacency

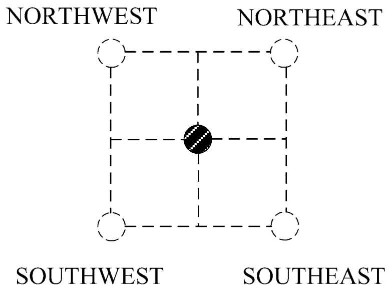

Fig. 7 Possible assignments to the *location* data member of a PullMove structure

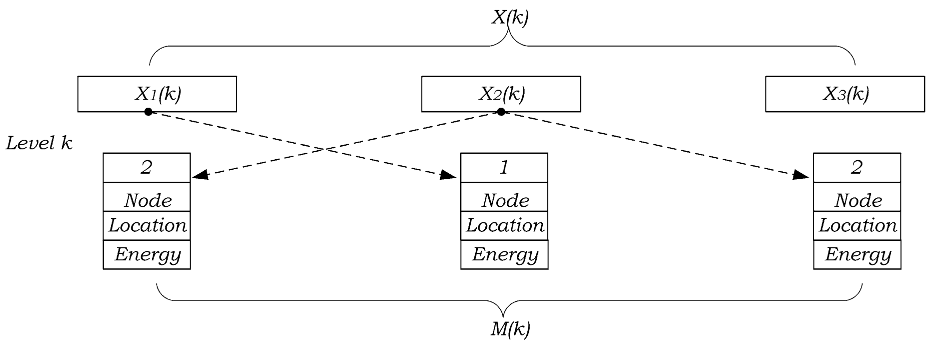

Fig. 8 Data structures for the filter and fan neighborhood

list refers to indexes in the adjacency table associated with amino acids, the coordinates of displaced nodes and the adjacency information can be simultaneously updated.

Another basic data structure is the one for representing a simple pull move. A pull move can be completely defined by reference to three basic components, the pull move initiating node, the location associated with the initial move and the energy value of the new conformation after completing the pull move. The first two components, which define the execution of the move as single- and multiple-pull, are implicitly determined by the pull-move rules as explained in Sect. 3.1. Since the evaluation of a pull move requires the identification of the resulting conformation, it is useful to record the corresponding energy. Specifically, a pull move structure is defined as $PullMove = \{node, location, energy\}$. There are 4 possible locations for a node to move to in the execution of a pull move, defined by the four diagonal positions adjacent to the node, which we denote by NORTHWEST, NORTHEAST, SOUTHEAST, and SOUTHWEST as depicted in Fig. 7.

3.3.2 *Filter-and-fan data structures*

In order to model the filter-and-fan neighborhood tree presented in Sect. 3.2, we extend the information associated with the pull move to include a new element, an integer data type named *conformation*, that identifies the conformation on which the move is carried out. Hence the pull move structure becomes $\text{PullMove} = \{conformation, node, location, energy\}$. From this point onward we say that a move m is a pull move if it is an instance of the PullMove structure. Also, we refer to $x(node)$ and $y(node)$ as the x and y coordinates of the pull move initiating *node*. An illustration of the data structures for one level of the filter and fan search is depicted in Fig. 8.

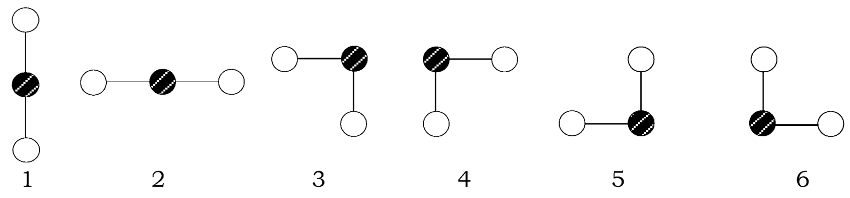

Fig. 9 Six possible instances of a *local conformation*

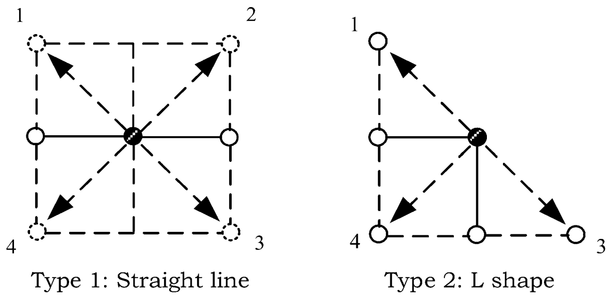

Fig. 10 Detecting a valid pull moves for *local conformations* of type 1 (*left*) and type 2 (*right*)

Following the conventions introduced in Sect. 3.2, $X(k)$ denotes the set of best conformations at level k, $X_i(k)$ the i-th best conformation in $X(k)$, $M(k)$ the set of best pull moves associated with $X(k)$ and m_{ik} the i-th best pull move in $M(k)$. The logical relationship between $X_i(k)$ and m_{ik} has been identified by the index of the *conformation* in the generalized pull move structure. From the figure we can see that the best and the third best pull moves were derived from the second best conformation at level k while the second best pull move derived from the first conformation.

We now describe the procedures that are essential for the efficiency of the algorithm.

3.3.3 Procedures for pull move neighborhood

The implementation of a pull move neighborhood requires three basic functions associated respectively with: (a) detecting all valid pull moves available, (b) executing the pull move, and (c) computing the energy value of the resulting conformation.

We define a *local conformation* of node i, denoted by *localConformation*(i), to be the structure in the current conformation formed by node i and its adjacent nodes in the amino acid sequence. Figure 9 illustrates all six possible instances of a local conformation for the stripped black node appearing in the middle of the structure. Indexes 1 to 6 are used to code each of these possibilities.

As can be seen, a *local conformation* is either a straight line (type 1) or an L shape (type 2). In a 2D lattice, there are two possibilities for a straight line (indexes 1 and 2) and four possibilities for an L shape (indexes 3 to 6). As shown in Fig. 10, for a straight line, all the four moving directions are inspected; for an L shape, only three of them are possible.

We employ the following notation and definitions.

pullMoveList: a list storing all valid pull moves associated with a conformation.

GetLocalConformation (i, A): returns the local conformation code associated with the i-th node.

IsLegal $(i, j, A, \mathit{localConf}(i))$: returns true if displacing node i to location j is a valid pull move, and false otherwise.

AddPullMove (*node, location, pullMoveList*): attaches a valid pull move to the pull move list with initial energy value of zero. (Actual energy values are then evaluated after executing the pull move.)

A straightforward method for detecting all valid pull moves is given by the following procedure:

```
Procedure DetectAllPullMoves (A, pullMoveList)
Begin
  Initialization
    localConformation(index) = 0
    pullMoveList = ∅
  For index = 1 to |C|
    localConformation(index) = GetLocalConformation (index, A)
    For location = 1 to numOfLocations
      If IsLegal (index, location, A, localConformation(node)) then
        AddPullMove (index, location, pullMoveList)
      End If
    End For
  End For
End Procedure
```

It is important to realize that the evaluation of a pull move requires the execution of the pull move. Likewise, as the pull move process unfolds, adjacency information of the current active conformation is necessary to detect further valid pull moves and evaluate the corresponding complete moves associated with nodes of the filter-and-fan tree. Consider for implementation purposes an adjacency table A representing the current active conformation and A' be a clone of A on which a pull move m is executed.

GetConnected (*node*, *location*, A, A'): displaces as many nodes as needed to make the conformation connected after moving the initiating *node* to the specified *location* according to the pull move rules described in Sect. 3.1. Once the move is completed, both the pull-move initiating node and the last displaced node are set *tabu-active* for a predefined number of iterations. We keep η_1 independent tabu lists throughout the tree search.

```
Procedure ExecutePullMove ($A, m$)
Begin
  Initialization Step
    $A' = A$
  Switch (location)
    Case NORTHWEST: $x(node) = x'(node) - 1$
                    $y(node) = y'(node) + 1$
    Case NORTHEAST: $x(node) = x'(node) + 1$
                    $y(node) = y'(node) + 1$
    Case SOUTHEAST: $x(node) = x'(node) + 1$
                    $y(node) = y'(node) - 1$
    Case SOUTHWEST: $x(node) = x'(node) - 1$
                    $y(node) = y'(node) - 1$
  End Switch
  GetConnected (node, location, $A$, $A'$)
End Procedure
```

Let $H(A)$ denote a list of indexes of type H amino acids in the chain and let $N(h)$ denote the indexes of nodes adjacent to node h. Then the energy E of a conformation A in the HP lattice model can be efficiently computed in $O(|C|)$ time using the following procedure.

```
Begin
Procedure EvaluateEnergy ($A$)
    Initialization Step
        $E = 0$
    For $h \in H(A)$
        For $j \in N(h)$
            If $j \in H(A)$ and $h > j$ then
                $E = E - 1$
            End If
        End For
    End For
Return $E$
End Procedure
```

3.3.4 Procedures for the filter-and-fan neighborhood

The procedures needed for the construction of the filter-and-fan neighborhood tree are as follows. Let $V(k) = \{V_1(k), V_2(k), \dots, V_{\eta_1}(k)\}(|V_i(k)| = \eta_2)$ denote the set of η moves evaluated at the level k from which the η_1 best moves for the (new) collection $M(k) = \{m_{1k}, m_{2k}, \dots, m_{\eta_1 k}\}$ are selected ($M(k) \subset V(k), k > 0$), thus building conformations $X(k+1)$.

EvaluatePullMoveList ($X_i(k)$, $V_i(k)$): evaluates the energy value for all pull moves in $V_i(k)$ associated with the i-th best conformation in $X(k)$ and retains the resultant energy value in the corresponding data member of the PullMove structure. The standard aspiration criterion is used, which overrides the move tabu status when it improves the best conformation found so far.

SortPullMoves ($V(k)$, η): sorts the η valid pull moves in $V(k)$ in ascending order of the associated energy values.

The following procedure is use to set up the first level of the filter-and-fan tree.

```
Procedure InitializeTree ($A$)
Begin
    DetectAllPullMoves ($A$, $V(k)$)
    EvaluatePullMoveList ($A$, $V(k)$)
    SortPullMoves ($V(k)$, $\eta_2$)
    For $i = 1$ to $\eta_1$
        $X_i = A$
        ExecutePullMove ($X_i$, $m_{i1}$)
    End For
End Procedure
```

Further levels of the tree are created as follows. Define A^* as the current best conformation associated with the lowest energy value E^*. Consider also the following basic procedures.

AppendSets ($V(k)$, $V_i(k)$, η_2): appends the set $V_i(k)$ of η_2 moves to the set $V(k)$.

UpdateTabuList (T_i): updates the tabu list T_i associated with the i-th conformation at the current depth of the tree search.

The following procedure is used to generate L levels of a filter-and-fan neighborhood tree. The first-improvement strategy can be used to terminate the tree search once an improvement is found.

```
Procedure GenerateTree $(X, \eta_1, \eta_2, L)$
Begin
  While $k < L$
    For $i = 1$ to $\eta_1$
      DetectAllPullMoves $(X_i(k), V_i(k))$
      EvaluatePullMoveList $(X_i(k), V_i(k))$
      SortPullMoves $(V_i(k), |V_i(k)|)$
      AppendSets $(V(k), V_i(k), \eta_2)$
    End For
    SortPullMoves $(V(k), \eta_1 * \eta_2)$
    For $i = 1$ to $\eta_1$
      ExecutePullMove $(X_i(k), m_{ik})$
      UpdateTabuList $(T_i)$
      $E =$ EvaluateEnergy $(X_i(k))$
      If $E < E^*$ then:
        $E^* = E$
        $A^* = X_i(k+1)$
      End If
    End For
    $k = k + 1$
  End While
End Procedure
```

Now the filter-and-fan procedure can simply be written as:

```
Procedure Filter-and-Fan $(A, \eta_1, \eta_2, L)$
Begin
  InitializeTree $(A)$
  GenerateTree $(X, \eta_1, \eta_2, L)$
End Procedure
```

3.3.5 The tabu search procedure

The local search phase of the algorithm is carried out by a simple tabu search that utilizes only a short-term memory component. Simple pull moves are used to explore the local search under tabu restrictions embodied in a tabu list T and using the classical aspiration criterion explained earlier. The search is performed until encountering a sequence of NMAX iterations that fail to decrease the lowest energy value found so far. The first-improvement strategy is adopted in order to speed-up the search. To simplify the notation we assume that m^* is a global variable that identifies the best pull-move determined by the *EvaluatePull-MoveList* function with associated energy E.

```
Procedure *TabuSearch* $(A)$
Begin
  Initialization
    $A^* = A$
    $T = \varnothing$
    NI = 0
  While NI < NMAX
    *DetectAllPullMoves* $(A, pullMoveList)$
    *EvaluatePullMoveList* $(A, pullMoveList)$
    *ExecutePullMove* $(A, m^*)$
    *UpdateTabuList* $(T)$
    If $E < E^*$ then
      $A^* = A$
      $E^* = E$
      NI = 0
    End If
  End While
  Return $A^*$
End Procedure
```

The main algorithm can now be sketched as:

```
Algorithm *Filter-and-Fan* $(A, \eta_1, \eta_2, L)$
Begin
  *TabuSearch* $(A)$
  *InitializeTree* $(A^*)$
  *GenerateTree* $(A^*, \eta_1, \eta_2, L)$
End Algorithm
```

4 Computational results

This section analyzes the performance of the proposed Filter-and-Fan algorithm in relation to the performances of the current best alternative approaches. Computations were carried out first on the 11 standard 2D HP benchmark problems. We also test our algorithm on the 11 most studied Functional Model Proteins (FMP).[1] Certain FMP instances are believed to be challenging because of the fact that each sequence has a unique native structure (i.e., only one optimal solution) and many *first excited states* (FES, or best suboptimal conformations). The choice of these 11 FMP sequences, however, seems purely random and does not include instances where a large number of FES exist (the maximum FES of these 11 FMP is 299). Hence we further test 25 potentially more difficult FMP instances containing over 530 FES's.

[1]There are more than 15000 FMP sequences on the website http://www.cs.nott.ac.uk/~nxk/HP-PDB/2dfmp.html. Each instance has a length of 23. The optimal energies of all FMP instances were obtained by complete enumeration algorithms.

4.1 Experiments with standard 2D HP benchmark

Comparisons are established in terms of the *effectiveness* and *efficiency* of the algorithms. We first discuss the effectiveness of the algorithms based upon the results provided in Table 2. The first three columns present the identification number of each problem in the test bank and their characteristics in terms of the number of H and P amino acids in the chain. The fourth column provides the best-known energy value (E^*) for each instance and the remaining columns report the energy values of the best conformations found by the algorithms. The energy values for instances 1–4 are known to be optimal (Hart and Istrail 1996). These algorithms are[2]: the Filter-and-Fan approach proposed here (F&F), the Tabu Search approach (GTabu) by Lesh et al. (2003), the Genetic Algorithm (GA) by Unger and Moult (1993), the Evolutionary Monte Carlo (EMC) algorithm by Liang and Wong (2001), two variants of the Ant Colony Optimization algorithm (New-ACO and ACO-HPPFP-3)[3] by Shmygelska and Hoos (2003, 2005) and the Pruned Enriched Rosenbluth Method (PERM) by Hsu et al. (2003a, 2003b). (To simplify the notation we will refer to the ACO-HPPFP-3 algorithm simply as ACO-3.) Boldfaced numbers correspond to cases where an algorithm found the best-known energy value and dashed cells indicate that the algorithm has not been tested on the corresponding problem.

Results for GA, EMC, New-ACO, ACO-3 and PERM reported in this section are directly taken from Shmygelska and Hoos (2003, 2005). For PERM we consider the expected time to solve a given problem which has captured the effect of different starting point of the given HP sequence. We refer to Shmygelska and Hoos (2005) for parameter settings of PERM and the function for calculating the expected time. Results for GTabu are taken from Lesh et al. (2003), which represent the expected time to find the best solutions for the four largest problems. The last three rows of Table 2 provide information on the total deviation in units of energy above the best known energy values.

For an accurate comparative analysis, it is important to consider the conditions under which the different algorithms obtain their best solutions. Solutions for GA and EMC correspond to the best energy values found in 5 independent runs of the algorithms. New-ACO performs between 100 and 500 runs on each instance and reports the best energy value over all runs. More specifically, the ACO algorithm performs 500 runs for chains up to 50 amino acids, 300 runs for chains containing more than 50 and up to 64 amino acids, and 100 runs for larger instances. The same run scheme is employed to the ACO-3 algorithm, except for instances larger than 64 on which only 20 runs are executed. Results for PERM are the best energy values obtained in a number of 20 to 200 runs. The results of GTabu are based on 200 runs. We report the best energy values found by the F&F in a single run.

The last two rows of Table 2 provide the deviations for the partial testbed considered by GA and EMC to allow for a direct comparison with the other algorithms on the corresponding subsets. We can see that only F&F, ACO-3 and PERM find the best known energy values for the problems tested by GA and EMC algorithms, yielding a zero unit deviation from the best know energy values.

[2]The selection of alternative algorithms to compare with was based on whether they attempted to solve all the 11 standard benchmark problems with size up to 100. Some recent approaches, e.g., Krasnogor et al. (2002), Pelta and Krasnogor (2004), Cutello et al. (2005, 2006) and Jiang et al. (2003), also studied these standard 2D HP instances, but none of them solves instances of size larger than 64. Therefore, they are not included in Tables 1 and 2 but are considered later on in the FMP benchmarks.

[3]New ACO and ACO-3 are two improved versions of the original Ant Colony algorithm initially proposed in Shmygelska et al. (2002).

Table 2 Comparison of algorithms' effectiveness

Problem				Best known	Best energy values						
Number	$\lvert H\rvert$	$\lvert P\rvert$	$\lvert C\rvert$	E[a]	F&F	GTabu	GA	EMC	New ACO	ACO-3	PERM
1	10	10	20	**−9**	**−9**	–	**−9**	**−9**	**−9**	**−9**	**−9**
2	10	14	24	**−9**	**−9**	–	**−9**	**−9**	**−9**	**−9**	**−9**
3	9	16	25	**−8**	**−8**	–	**−8**	**−8**	**−8**	**−8**	**−8**
4	16	20	36	**−14**	**−14**	–	**−14**	**−14**	**−14**	**−14**	**−14**
5	25	23	48	**−23**	**−23**	–	−22	**−23**	**−23**	**−23**	**−23**
6	24	26	50	**−21**	**−21**	–	**−21**	**−21**	**−21**	**−21**	**−21**
7	43	17	60	**−36**	**−36**	–	−34	−35	**−36**	**−36**	**−36**
8	42	22	64	**−42**	**−42**	**−42**	−37	−39	**−42**	**−42**	**−42**[a]
9	59	26	85	**−53**	**−53**	**−53**	–	−52	−51	**−53**	**−53**
10	56	44	100	**−48**	**−48**	**−48**	–	–	−47	−47	**−48**
11	55	45	100	**−50**	**−50**	**−50**	–	–	−47	−49	**−50**
Total deviation above best known				0	0	0	8	5	6	2	0
GA				0	0	–	8	4	0	0	0
EMC				0	0	–	–	5	2	0	0

[a]This best known energy −42 was found by Hsu et al. (2003a, 2003b) by means of "special tricks" (problem-specific rules). With their general rules, the lowest energy reached by Hsu et al. (2003a, 2003b) is −39. Shmygelska and Hoos (2003) report −38. Shmygelska and Hoos (2005) report −42 with 78 hours of running time

In an overall analysis of the complete testbed, the GA and EMC algorithms are clearly not competitive. Even for relatively small instances these algorithms fail to find the best solutions in all 5 runs and none of these methods attempts to solve the larger instances. The New-ACO and ACO-3 algorithms fail to find three and two of the best solutions, respectively. Although GTabu finds best solutions to the four largest instances, it does not do so with 100% success rate. Lesh et al. (2003) report a success rate of 40%, 12% and 7.5% for problems 9, 10 and 11, respectively. Only PERM and F&F find the best known solutions to all problems. However, as stated in Hsu et al. (2003a) a fundamental drawback of PERM as any other chain growth algorithm is the difficulty of solving instances with certain types of structures such as problem 8. In the original implementation of Hsu et al. (2003a, 2003b), the best energy reached by PERM without the aid of supplementary "tricks" (i.e using problem-specific rules to replace what they call "blind" search) is −39. In the implementation of Shmygelska and Hoos (2005), −42 is reached after 78 hours of run time on a 2.4 GHz Pentium IV machine. Our F&F algorithm, in contrast, is able to handle this instance very efficiently. One possible reason contributing to this difference in performance may be attributed to both the type of neighborhood employed by each of the algorithms and the static versus dynamic nature of these neighborhoods. As will be discussed later in more detail, while PERM restricts the local search to moves obtained by joining segments of a pre-defined set of partial conformations, our F&F search allows for segments to be built and joined dynamically within variable-depth neighborhoods that operate on complete conformations, thus giving no bias on the shape of the new conformations.

Table 3 Comparison of algorithms' efficiency

Problem				Running times (seconds)						
Number	$\|H\|$	$\|P\|$	$\|C\|$	F&F	GTabu[a]	GA	EMC	ACO New	ACO-3	PERM
1	10	10	20	<1	–	–	–	1.67	<1	<1
2	10	14	24	<1	–	–	–	1.26	<1	<1
3	9	16	25	<1	–	–	–	5.31	<1	2
4	16	20	36	2	–	–	–	5.91	4	<1
5	25	23	48	4	–	–	–	202.90	60	2
6	24	26	50	11	–	–	–	2476.46	15	3
7	43	17	60	23	–	–	–	**31235.62**	1200	4
8	42	22	64	12	<1800	–	–	2922.47	5400	**78 hrs**
9	59	26	85	38	<9000	–	–	10950.67	**86400**[b]	60
10	56	44	100	30888	<30600	–	–	5417.76	36000	480
11	55	45	100	32503	<48600	–	–	14853.61	43200	1200
Average of running times				5771	8231			6188.51	15662	25681

[a]These values refer to the expected time for finding the best known solutions reported in Lesh et al. (2003) on a 1 GHz Alpha processor

[b]This value corresponds to a cut-off time of 1 day for each run of the algorithm in a 2.4 GHz Pentium IV processor

We now assess the relative efficiency of the algorithms for which computational times have been reported in the literature. No running times are available for the GA and EMC algorithms. For the remaining algorithms, times are in seconds on a 2.4 GHz Pentium IV computer processor, except for New-ACO whose test runs were executed on a 1 GHz Pentium III machine. Table 3 summarizes these results, where times for New-ACO have been divided by approximately a factor of 2 (as can be derived from Dongarra 2006) to reflect the relative speed of the different computer processors. Boldfaced figures in Table 3 indicate excessive running times for the problems considered, identifying cases in which the associated algorithms encounter significant difficulty in finding the best known solutions.

F&F reports times to find its lowest energy conformation values in one run. GTabu reports the expected time to find the best solutions obtained by dividing the computational time by the success rate achieved on the corresponding instance (Lesh et al. 2003). New ACO and ACO-3 report average times per run (computed over multiple statistically independent runs) required by the algorithms to find a solution of the quality specified in Table 2. As for PERM, the figures refer to the expected time for solving a given problem which has captured the effect of starting from N or C terminus of the given HP sequence. (In this sense, PERM can be seen as a multi-start probabilistic algorithm where statistical information gathered in one run is carried forward to start the search in subsequent runs.)

New-ACO and ACO-3 reveal no significant difference in efficiency when solving relatively small problems 1 to 4. For larger problems the efficiency of these two algorithms varies with no specific pattern. ACO-3 is clearly more efficient on problems 6 and 7 while New-ACO performs better on problems 8 and 10. Although New-ACO and ACO-3 algorithms significantly improve the performance of their original version (Shmygelska et al. 2002) these approaches remain inefficient compared to PERM in all but one problem.

F&F, on the other hand, provides the smallest average of running times while finding all best known solutions. PERM's efficiency is greatly hampered by solving problems with a symmetric optimal structure such as problem 8. As reported in Shmygelska and Hoos (2003), after running for 48 hours of wall clock time on a 1 GHz Pentium III CPU machine, the PERM algorithm was sill unable to improve its best solution −38 for problem 8. In fact, to find a solution having the best known energy −42 for this problem, Shmygelska and Hoos (2005) determined that the most effective PERM variant needed 78 hours of wall clock time on a 2.4 GHz Pentium IV CPU processor. By contrast, the F&F algorithm only requires 12 seconds to obtain such a solution. GTabu's expected time to find the best solutions is relatively large due to a low success rate on large problems (less than 20% for the two 100-node instances).

In summary, the results on the 11 standard HP benchmark problems show that the F&F approach clearly outperforms GA, EMC, New-ACO and ACO-3 in terms of solution quality. GTabu potentially matches F&F in solution quality but with a relatively low success rate. Only PERM matches the solution quality of F&F, but requires significantly greater computation time on average due to its extreme inefficiency when dealing with problems with a symmetric optimal structure.

4.2 Experiments with functional model proteins

We now present the computational results on a set of Functional Model Proteins (FMP) recently used for computational testing by some leading evolutionary algorithms for the 2D HP Protein Folding problem, namely the Memetic Algorithm by Krasnogor et al. (2002) and the Immune Algorithms by Cutello et al. (2005, 2006). Table 4 provides a comparative analysis on the 11 FMP benchmark problems.

The table above shows that F&F finds optimal energies for all the 11 FMP instances in less than 2 seconds of computational time on average. F&F also outperforms IM in solution

Table 4 Results on the 11 FMP benchmark instances

Problem	Opt. energy	FES energy	# FES	F&F Energy	F&F Run time	IM[a] Best energy	IM[a] SR	MA[b] Best energy
1	**−20**	−18	1	**−20**	<1	**−20**	100%	**−20**
2	**−17**	−15	1	**−17**	<1	**−17**	100%	**−17**
3	**−16**	−14	101	**−16**	3	**−16**	56.67%	**−16**
4	**−20**	−18	7	**−20**	1	**−20**	100%	**−20**
5	**−17**	−15	8	**−17**	<1	**−17**	100%	**−17**
6	**−13**	−11	15	**−13**	5	**−13**	100%	**−13**
7	**−26**	−24	78	**−26**	<1	**−26**	100%	**−26**
8	**−16**	−14	30	**−16**	2	**−16**	100%	**−16**
9	**−15**	−13	144	**−15**	<1	**−15**	100%	**−15**
10	**−14**	−12	77	**−14**	<1	**−14**	100%	**−14**
11	**−15**	−13	299	**−15**	<1	**−15**	100%	**−15**

[a]Based on Cutello et al. (2005, 2006): best results obtained from 30 independent runs with the best parameter setting

[b]Based on Krasnogor et al. (2002): best results obtained from multiple runs

quality, noting that the success rate of IM for problem 3 is only 56.67%. This relative difference between the methods is further reflected in the performance of IM on the standard 2D HP benchmark problems in Table 1. The latest efforts by Cutello et al. (2006) to solve instances up to size 64 yields −35 and −39 as the best energy values found by the IM algorithm for problems 7 and 8, a deviation of 1 and 3 units of energy from the best know value for each of these problems, respectively. Only by incorporating long range moves is the IM algorithm able to attain the best energy −42 for problem 8, but even then the method exhibited a success rate of only 3.33%, yielding an average energy value of −39.3. The MA algorithm finds all the optimal energies for the 11 FMP benchmark problems; however, its success rate for these instances is unclear.[4] The MA algorithm of Krasnogor et al. (2002) finds the energy −39 (a deviation of 3 units from the best known) for problem 8 in the standard 2D HP benchmark set while their improved version proposed in Pelta and Krasnogor (2004) finds −35 (a deviation of 1 unit of energy) for problem 7. Comparison of efficiency with IM and MA is again very difficult as the authors only report the number of evaluations before reaching the best.

We note that the choice of these 11 FMP sequences seems purely random and they are far from being the most challenging FMP instances available (the maximum number of FES among these 11 FMP benchmarks is only 299). Hence we further studied 25 FMP sequences whose number of FES exceeds 530. To the best of our knowledge no problems of this dimension have been yet addressed in the literature. The characteristics of this additional testset along with the computational results for the F&F algorithm are presented in Table 5.

As it can be seen, the F&F algorithm finds optimal energy values to all the 25 instances tested. Not surprisingly, the analysis also discloses an increase of running time required on average relative to the time necessary to solve the 11 benchmark FMP instances, due to the much larger number suboptimal conformations existent in these 25 instances.

Finally, we should mention that after our paper has been submitted a new Monte Carlo based algorithm was developed by Zhang et al. (2007). Essentially, this algorithm is a combination of configuration-bias Monte Carlo (CBMC) due to Siepmann and Frenkel (1992) and the multigrid Monte Carlo (MGMC) approach by Goodman and Sokal (1986). Just like our F&F algorithm, Zhang et al.'s algorithm finds the best known energy values for all instances of the standard testbed of Table 2. Their algorithm is faster than ours for the two 100 amino acid instances, but apparently not as fast as our F&F algorithm for the hard 85 amino acid instance.

5 Conclusions

The 2D HP lattice model of the Protein Folding Problem (PFP) has brought new challenges to the optimization community. Although the problem has been extensively studied for more than a decade, to the best of our knowledge none of the attempts to create exact solution methods has succeeded in solving even the smallest instance (containing 20 amino acids) of the standard testbed considered here. Therefore, the challenge has been passed to advanced metaheuristic approaches to provide high quality solutions for problems of reasonable size. Classical genetic algorithms, probabilistic multi-start algorithms based on Monte Carlo simulations, hybrid ant colony optimization approaches and tabu search approaches have all entered the fray in an effort to find the best possible solutions to these protein folding models.

[4] Krasnogor et al. (2002) reported the success rate for one particular 2D triangular lattice HP instance based on various parameter settings of their Memetic Algorithm. But no such information was provided on their testing of the FMP benchmark instances.

Table 5 Results for selected 25 FMP instance whose number of FES exceeds 530

Problem	Sequence of amino-acids	Optimal energy	FES energy	Number of FES	F&F	Run time
1	phppphphphhhphhhhhphhhh	**−20**	−18	534	**−20**	8
2	phppphphphhhphhhhhhphph	**−20**	−18	545	**−20**	<1
3	hhphhphppphpppphhhhhhh	**−16**	−14	546	**−16**	1050
4	phpppphphhhhphhphhhhhh	**−20**	−18	595	**−20**	6
5	phppphhpphhhpphhhhphhhh	**−20**	−18	609	**−20**	<1
6	phpphhpppphhphphhhppphp	**−14**	−12	618	**−14**	3
7	phpphpphhphpppphhphpph	**−14**	−12	625	**−14**	1789
8	phpppphpphhhphhphphhhh	**−18**	−16	635	**−18**	3
9	phhphpppppphphpphhpphph	**−14**	−12	635	**−14**	1
10	phpphhphhpppphphhphhhp	**−14**	−12	650	**−14**	838
11	phhphphhhpphpppphhphhp	**−14**	−12	650	**−14**	5
12	phphhphpppppphpphhpphph	**−14**	−12	680	**−14**	8
13	phphhphppppphhpphhpphph	**−14**	−12	699	**−14**	9
14	phhphppppppphpphphhhphhp	**−14**	−12	712	**−14**	8
15	hphppphppppphhhhhhhhhh	**−16**	−14	778	**−16**	1032
16	phpphppppppphpppppphhhp	**−8**	−6	801	**−8**	22
17	phhphpphhphpppppphphpph	**−14**	−12	892	**−14**	1308
18	phhphpppppphhpphpphhpph	**−14**	−12	946	**−14**	9
19	phpphpphhppppppphphpppph	**−11**	−9	1037	**−11**	1780
20	pphphpphhpphphppppphpphp	**−11**	−9	1164	**−11**	6
21	phpphphhpphhppppphphpph	**−13**	−11	1261	**−13**	917
22	phpphpppphhhppppphhphhp	**−11**	−9	1710	**−11**	1168
23	phhphhpppphhpphpphhpph	**−14**	−12	1720	**−14**	1089
24	phhpphhpphhphpppppphhph	**−14**	−12	1764	**−14**	1089
25	phphhphpphhphppppphhph	**−14**	−12	2532	**−14**	1229

(These are all probabilistic approaches whose performance needs to be evaluated through multiple runs, whereas our F&F is the first deterministic approach that requires only a single run.)

It is worth noting that although advanced metaheuristics are currently capable of finding optimal and near-optimal solutions for other path-based permutation problems in graphs containing hundreds and sometimes millions of nodes, none of the metaheuristics proposed for the protein folding problem has proved capable of finding a best known solution within a reasonable time span for each of the problems in the testbed of this study, where the largest instance contains only 100 nodes. In fact, as noted earlier the F&F algorithm performs more robustly and efficiently than the current leading algorithms requiring only a single solution trial and approximately 10 seconds on average to obtain best known solutions to 9 out of 11 problems. By contrast, the PERM algorithm, which is the best of the alternative algorithms, requires a hundred or more trials in the typical case to obtain best solutions to these 9 problems. On the remaining 2 largest problems, F&F obtains the best known solutions after five re-starts from diverse solutions generated in the course of the algorithm, which accounted for approximately 9 hours of computation time. Notably, only a single run of 15 minutes on average was necessary for the F&F algorithm to find a solution that is just one unit away from the best known solution obtained by the best alternative method within similar amount of time. By comparison, this best alternative algorithm required hundreds of runs and more than 3 days (approximately 78 hours) of wall clock time to find the best solution for instance 8 (with 64 residues) in the standard testbed, though the F&F algorithm required approximately 12 seconds on an equivalent computer to find such a solution.

The computational analysis carried out in this study clearly indicates that further research is needed to solve larger instances of the protein folding model more effectively. In pursuit of this goal, we offer some tentative speculations about the conditions exploited by our approach (and other approaches) that lead to a successful algorithm for protein folding, and ways that our method may be improved. In the context of the HP PFP problem, it is intriguing to note that both PERM and the ACO variants incorporate a number of strategies resembling those that have also been proposed in a variety of tabu search applications and that we have found relevant in our approach for this problem. We anticipate that these commonalities, and variations in the way such strategies have been applied, may provide clues about differences in the way the various methods perform, and more importantly may offer insights into the development of more effective algorithms.

The main feature of the PERM framework is its ability to search a restricted solution space by two processes: (1) operating on selected subsets of promising partial conformations recorded throughout the search (a statistical weighting process called *enrichment*), and (2) eliminating conformations of low weight (a process called *pruning*). Currently active partial conformations are then used as building blocks to create complete (feasible) conformations. Such iterative constructive processes of PERM have interesting links to the *vocabulary building strategy* of tabu search (e.g. Glover 1996a, 1996b, 1999) in which partial solution "fragments" are used to create new complete solutions, and the evaluation of these fragments is given by weights derived from conditional frequency memory. When statistical information is used in place of explicit memory structures, as done in PERM, additional links are established between such strategies and *Probabilistic Tabu Search* (Glover and Laguna 1993; Glover 1989, 1995). Finally, the *pool* of high-evaluation partial solutions maintained by the associated *enrichment* and *pruning* PERM strategies shares an intriguing connection with the process of elite solution filtering, which is the foundation of scatter search (Glover 1977) and is exploited in several TS *candidate list strategies*.

Similar principles are also implicit in the design of the New ACO and ACO-3 algorithms. As is the case of conventional evolutionary approaches, including traditional genetic algorithms (GAs), the customary ant colony optimization (ACO) approaches often rely on incorporating supplementary procedures in order to enhance their performance when encountering challenging problems. Genetic algorithms and evolutionary Monte Carlo approaches applied to protein folding problems, for example, lack such enhancement and have been shown to lag far behind the alternative approaches for these problems. Similarly, in their analysis of ant colony approaches for protein folding, Shmygelska et al. (2002) state "It is worth noting that without a local search phase, the performance of our ACO method is abysmal." As a result, the authors augmented their method with "probabilistic iterative improvement local search" to create the appreciably more effective New-ACO and ACO-3 algorithms. In addition, these algorithms make use of intensification and diversification strategies by exploring paths of elite ants (i.e. high quality solutions) by local search and allowing for a larger number of ants to be subjected to local search, thus opening up the neighborhood to new regions of the solution space. This augmentation of ant colony ideas by introducing local search utilizing intensification and diversification strategies likewise share similarities with TS proposals, which operate in intimate association with local search rather than utilizing a design that treats it as something apart. In particular, the marriage of local search with intensification and diversification in tabu search takes the form of decision rules to foster the inclusion of attributes of elite solutions and focus the search in regions around these solutions, joined by complementary rules to induce the search to visit regions not previously visited. (See Glover and Laguna 1997 for a summary of such rules.) An important observation stems from the fact that our simple deterministic tie-breaking rule used in the F&F search worked just as well as a variety of randomized rules that we tested empirically. Other tie-breaking rules, which we believe are interesting to explore, may incorporate long-term memory considerations so as to generate an appropriate strategic oscillation between intensification and diversification with respect to searches that derive from conformations with the same energy value.

In sum, local search that (a) utilizes memory of elite solutions and their attributes (either in direct or statistical form) and (b) strategically drives the search into new regions, plays a critical role in the performance of the leading metaheuristic algorithms for the PFP. In this paper we explore mechanisms for achieving these aspects by means of a filter-and-fan approach incorporating a simple tabu search structure. This approach alternates between single-path and multiple-path tabu searches using component moves provided by the pull-move neighborhood, subject to short-term memory controls. The success of the algorithm compared to alternative state-of-the-art algorithms owes to two fundamental components: (i) the dynamic and adaptive feature of the search method in exploiting the pull-move neighborhood structure; and (ii) the interplay between the tabu search and the tree search phases that creates a strategic oscillation between intensification and diversification. In recognition of this, we anticipate that further improvements in efficiency may result by incorporating longer-term tabu search memory components to achieve higher levels of intensification, and by means of vocabulary building strategies that incorporate ejection chain methods and path-relinking. Promising areas for future investigation consist of extending the proposed approaches to the solution of other variants of the 2D HP square lattice model such as the honeycomb lattice, the 3D HP square lattice and the diamond lattice.

References

Anfinsen, C. B., Haber, E., Sela, M., & White, F. H. (1961). The kinetics of formation of native ribonuclease during oxidation of the reduced polypeptide chain. *Proceedings of the National Academy of Sciences*, *47*(9), 1309–1314.

Backofen, R. (2001). The protein structure prediction problem: a constraint optimization approach using a new lower bound. *Constraints*, *6*, 223–255.

Berman, H. M., Westbrook, J., Feng, Z. K., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalov, I. N., & Bourne, P. E. (2000). The protein data bank. *Nucleic Acids Research*, *28*(1), 235–242.

Bornberg-Bauer, E. (1997). Chain growth algorithms for HP-type lattice proteins. In *Proceedings of the first annual international conferences on computational molecular biology (RECOMB97)* (pp. 47–55). New York: ACM Press.

Chan, H. S., & Dill, K. A. (1993). The protein folding problem. *Physics Today*, *46*(2), 24–32.

Chikenji, G., Kiduchi, M., & Iba, Y. (1999). Multi-self-overlap ensemble for protein folding: ground state search and thermodynamics. *Physical Review Letters*, *83*(9), 1886–1889.

Covell, D. G., & Jernigan, R. L. (1990). Conformation of folded proteins in restricted spaces. *Biochemistry*, *29*, 3287–3294.

Crescenzi, P., Goldman, D., Papadimitriou, C., Piccolboni, A., & Yanakakis, M. (1998). On the complexity of protein folding. In *Proceedings of the 13th annual ACM symposium of theory of computing* (STOC 98) (pp. 597–603).

Cutello, V., Morelli, G., Nicosia, G., & Pavone, M. (2005). Immune algorithms with aging operators for the string folding problem and the protein folding problem. In *Lecture notes in computer sciences* (Vol. 3348, pp. 80–90). Berlin: Springer.

Cutello, V., Nicosia, G., Pavone, M., & Timmis, J. (2006). An immune algorithm for protein structure prediction on lattice models. *IEEE Transaction on Evolutionary Computation*.

Dandekar, T., & Argos, P. (1994). Folding the main chain of small proteins with genetic algorithm. *Journal of Molecular Biology*, *236*, 844–861.

Dill, K. A. (1985). Theory for the folding and stability of globular proteins. *Biochemistry*, *24*(6), 1501–1509.

Dongarra, J. J. (2006). *Performance of various computers using standard linear equations software* (Linpack Benchmark Report). University of Tennessee Computer Science Technical Report, CS-89-85.

Glover, F. (1977). Heuristics for integer programming using surrogate constraints. *Decision Sciences*, *8*(1), 156–166.

Glover, F. (1989). Tabu search—part I. *ORSA Journal on Computing*, *1*(3), 190–206.

Glover, F. (1992). New ejection chain and alternating path methods for traveling salesman problems. *Computer Science and Operations Research*, 449–509.

Glover, F. (1995). Tabu thresholding: improved search by nonmonotonic trajectories. *ORSA Journal on Computing*, *7*(4), 426–442.

Glover, F. (1996a). Ejection chains, reference structures and alternating path methods for traveling salesman problems. *Discrete Applied Mathematics*, *65*, 223–253.

Glover, F. (1996b). Tabu search and adaptive memory programming—advances, applications and challenges. In Barr, Helgason, & Kennington (Eds.), *Interfaces in computer science and operations research* (pp. 1–75). Dordrecht: Kluwer Academic.

Glover, F. (1998). A template for scatter search and path relinking. In J.-K. Hao, E. Lutton, E. Ronald, M. Schoenauer, & D. Snyers (Eds.), *Lecture notes in computer science: Vol. 1363. Artificial evolution* (pp. 3–51). Berlin: Springer.

Glover, F. (1999). Scatter search and path relinking. In D. Corne, M. Dorigo, & F. Glover (Eds.), *New ideas in optimization* (pp. 297–316). New York: McGraw–Hill.

Glover, F., & Laguna, M. (1993). Tabu search. In C. Reeves (Ed.), *Modern heuristic techniques for combinatorial problems* (pp. 71–140). Oxford: Blackwell.

Glover, F., & Laguna, M. (1997). *Tabu search*. Boston: Kluwer Academic.

Goodman, J., Sokal, A. D. (1986). Multigrid Monte Carlo method for lattice field theories. *Physics Review Letters*, *56*(10), 1015–1018.

Grassberger, P. (1997). Pruned-enriched Rosenbluth method: simulations of theta polymers of chain. *Physical Review*, *56*(3), 3682–3693.

Hart, W. E., & Istrail, S. (1996). Fast protein folding in the hydrophobic-hydrophilic model within three-eighth of optimal. *Journal of Computational Biology*, *3*(1), 53–96.

Hart, W. E., & Istrail, S. (1997). Lattice and off-lattice side chain models of protein folding: linear time structure prediction better than 86% of optimal. *Journal of Computational Biology*, *4*(3), 241–259.

Hirst, J. D. (1999). The evolutionary landscape of functional model proteins. *Protein Engineering*, *12*, 721–726.

Hsu, H. P., Mehra, V., Nadler, W., & Grassberger, P. (2003a). Growth algorithms for lattice heteropolymers at low temperatures. *Journal of Chemical Physics*, *118*(1), 444–451.

Hsu, H. P., Mehra, V., Nadler, W., & Grassberger, P. (2003b). Growth-based optimization algorithm for lattice heteropolymers. *Physical Review E*, *68*(2), 021113.

Jiang, T., Cui, Q., Shi, G., & Ma, S. (2003). Protein folding simulations of the hydrophobic-hydrophilic model by combining tabu search with genetic algorithms. *Journal of Chemical Physics*, *119*(8), 4592–4596.

Konig, R., & Dandekar, T. (1999). Improving genetic algorithms for protein folding simulation by systematic crossover. *BioSystems*, *50*, 17–25.

Krasnogor, N., Pelta, D., Lopez, P. M., Mocciola, P., & de la Canal, E. (1998). Genetic algorithms for the protein folding problem: a critical review. In *Proceedings of engineering of intelligence systems* (pp. 353–360). ICSC Academic Press.

Krasnogor, N., Hart, W. E., Smith, J. E., & Pelta, D. A. (1999). Protein structure prediction with evolutionary algorithms. In *Proceedings of the 1999 international genetic and evolutionary computation conference (GECCO99)*, San Mateo CA (pp. 1596–1601).

Krasnogor, N., Blackburnem, B., Pelta, D. A., & Burk, E. K. (2002). Multimeme algorithms for protein structure prediction. In *Lecture notes in computer science: Vol. 2439. Proceedings of parallel problem solving from nature* (pp. 769–778). Berlin: Springer.

Lau, K. F., & Dill, K. A. (1989). A lattice statistical mechanics model of the conformational and sequence spaces of proteins. *Macromolecules*, *22*, 2986–3997.

Lengauer, T. (1993). Algorithmic research problems in molecular bioinformatics. In *Proceedings of the second Israel symposium on theory of computing systems (ISTCS)*, Natanya, Israel (pp. 177–192).

Lesh, N., Mitzenmacher, M., & Whitesides, S. (2003). A complete and effective move set for simple protein folding. In *Proceedings of the 7th annual international conference on research in computational molecular biology (RECOMB)* (pp. 188–195). New York: ACM Press.

Liang, F., & Wong, W. H. (2001). Evolutionary Monte Carlo for protein folding simulations. *Journal of Chemical Physics*, *115*(7), 3374–3380.

Nunes, N. J., Chen, K., & Hutchinson (1996). Flexible lattice model to study protein folding. *Journal of Physical Chemistry*, *100*(24), 10443–10449.

Pelta, D. A., & Krasnogor, N. (2004). Multimeme algorithms using fuzzy logic based memes for protein structure prediction. In *Recent advances in memetic algorithms*. Berlin: Springer.

Ramakrishnan, R., Ramachandran, B., & Pekny, J. F. (1997). A dynamic Monte Carlo algorithm for exploration of dense conformational spaces in heteropolymers. *Journal of Chemical Physics*, *106*(6), 2418–2424.

Rego, C., & Glover, F. (2002). Local search and metaheuristics for the travelling salesman problem. In G. Gutin & A. Punnen (Eds.), *The travelling salesman problem and its variations* (pp. 309–368). Dordrecht: Kluwer Academic.

Richards, F. M. (1991). The protein folding problem. *Scientific American*, *264*(1), 54-7, 60-3.

Shmygelska, A., & Hoos, H. H. (2003). An improved ant colony optimization algorithm for the 2D HP protein folding problem. In *Lecture notes in computer science*. *Proceedings of advances in artificial intelligence, AI 2003* (pp. 400–417). Berlin: Springer.

Shmygelska, A., & Hoos, H. H. (2005). An ant colony optimization algorithm for the 2D and 3D hydrophobic polar protein folding problem. *BMC Bioinformatics*, *6*(1), 30.

Shmygelska, A., Hernandez, R., & Hoos, H. H. (2002). An ant colony algorithm for the 2D HP protein folding problem. In *Lecture notes in computer science: Vol. 2463*. *Proceedings of the 3rd workshop on ant algorithms* (pp. 40–52). Berlin: Springer.

Siepmann, J. I., Frenkel, D. (1992). Configurational-bias Monte Carlo: a new sampling scheme for flexible chains. *Molecular Physics*, *75*, 59–70.

Skolnick, J., & Kolinski, A. (1990). Simulations of the folding of globular proteins. *Science*, *250*, 1121–1125.

Socci, N. D., & Onuchic, J. N. (1994). Folding kinetics of protein like heteropolymers. *Journal of Chemical Physics*, *101*(2), 1519–1528.

Unger, R., & Moult, J. (1993). Genetic algorithms for protein folding simulations. *Journal of Molecular Biology*, *231*, 75–81.

Zhang, J., Kou, S. C., & Liu, J. S. (2007). Biopolymer structure simulation and optimization via fragment regrowth Monte Carlo. *Journal of Chemical Physics*, *126*, 225101(1)–225101(7).