Artificial Intelligence in Medicine (2007) **39**, 49—63

ELSEVIER

ARTIFICIAL INTELLIGENCE IN MEDICINE

http://www.intl.elsevierhealth.com/journals/aiim

Side chain placement using estimation of distribution algorithms

Roberto Santana*, **Pedro Larrañaga**, **Jose A. Lozano**

Department of Computer Science and Artificial Intelligence, University of the Basque Country, CP-20080, Donostia-San Sebastián, Spain

Received 21 December 2005; received in revised form 26 April 2006; accepted 28 April 2006

KEYWORDS
Protein folding;
Estimation of distribution algorithms;
Protein structure prediction;
Rotamers

Summary

Objective: This paper presents an algorithm for the solution of the side chain placement problem.
Methods and materials: The algorithm combines the application of the Goldstein elimination criterion with the univariate marginal distribution algorithm (UMDA), which stochastically searches the space of possible solutions. The suitability of the algorithm to address the problem is investigated using a set of 425 proteins.
Results: For a number of difficult instances where inference algorithms do not converge, it has been shown that UMDA is able to find better structures.
Conclusions: The results obtained show that the algorithm can achieve better structures than those obtained with other state-of-the-art methods like inference-based techniques. Additionally, a theoretical and empirical analysis of the computational cost of the algorithm introduced has been presented.

1. Introduction

Proteins are essential components of living organisms. They consist of a set of amino acids or residues which, under suitable conditions, fold to form a tertiary structure. Inferring the protein tertiary structure from its sequence is an essential problem in molecular biology [1]. Computational models of proteins are important components for the solution of the protein structure problem.

The type of models used to investigate the protein structure problem range from coarse-grained models [2—4], to more detailed atom-based ones [5—8]. These models allow the description of different candidate protein structure configurations, and have an associated energy function that enables the evaluation of the quality of the candidate protein structures [9]. Usually, the search of the best configuration is regarded as an optimization problem: to find the solution that optimizes a predefined fitness function. The design of algorithms for predicting the native

* Corresponding author. Tel.: +34 943018070; fax: +34 943219306.
E-mail address: rsantana@si.ehu.es (R. Santana).

0933-3657/$ — see front matter
doi:10.1016/j.artmed.2006.04.004

structure of a protein from its amino acid sequence is an area that is currently receiving an increasing attention in the field of optimization [6,8,10–13]. In this paper, we approach the protein structure problem by focusing on a related problem, that of protein side chain placement.

An amino acid has a peptide backbone and a distinctive side chain. Assuming that the position of the backbone is fixed, and considering fixed bond lengths, the location of the protein can be completely determined by the bond angles.

One of the approaches to address the protein structure problem is based on homology modeling. In this approach, a database of proteins with known structures is searched by looking for a homologous sequence (with a relevant degree of similarity with the target sequence). Once a candidate is found, its structure is used as a starting point to find the target protein structure. One possibility is to use the backbone of the found structure as a model and to search the best side chain configuration of the target protein.

The problem of finding an optimal positioning for the side chain residues is called side chain placement or side chain prediction [14–16] and its discrete version is known to be NP-hard [17]. The problem is important not only for homology modeling but also for protein design [13], where the goal is to find a protein able to fulfil a given function or to satisfy a number of structural features.

A way to address the problem is to constrain the search to the discrete space by means of discrete configurations of the angles, known as rotamers [5,18]. The inclusion of these discrete configurations implies an important problem reduction. Nevertheless, the problem remains exponential. Therefore, the conception of efficient search procedures arises as an important research problem.

Deterministic and stochastic methods have been proposed to cope with the side chain placement problem. In this paper, we introduce a stochastic optimization algorithm for the solution of this problem. This algorithm, which is based on the use of probability distributions, belongs to the family of estimation of distribution algorithms (EDAs) [19,20]. EDAs are evolutionary algorithms. They resemble genetic algorithms (GAs) [21,22] in the use of populations, but instead of employing genetic operators, they construct, at each generation, an explicit probability model of a set of selected solutions, and use this model to sample new solutions.

The paper is organized as follows. In the next section, the biological basis of the side chain placement problem is reviewed. An introduction to EDAs is presented in Section 3. In that section, the univariate marginal distribution algorithm (UMDA) is described. An analysis of its main steps is presented. Section 4 presents the UMDA approach to side chain placement. In Section 5, numerical results of the application of the algorithm to a set of 425 proteins are presented and discussed. Section 6 analyzes in detail the relationship between the UMDA approach to side chain prediction and previous proposals to this problem. The conclusions of the paper are outlined in Section 7 along with lines for further research.

2. Side chain placement problem

2.1. Rotamers and rotamer libraries

Proteins are macromolecules made up of up to 20 different amino acids, also referred to as residues. The protein configuration is defined by the choice of the amino acids for all the n residues.

An amino acid has a peptide backbone and a distinctive side chain. The peptide bond is defined by an amino group and a carboxyl group connected to an alpha carbon to which is attached a hydrogen atom, and a side chain group. A peptide bond is formed by the dehydration of the carboxyl group of one amino acid and the amino group of the next.

The dihedral angles between the amino group and the alpha carbon and carboxyl group are free to rotate. These angles are respectively referred as $\phi - \psi$ angles. Amino acids can connect to the backbone in many different ways. The backbone of the protein is the set of amino acid peptide backbones.

Fig. 1 shows[1](a) the complete native structure of the *pdb*1*mr j* protein,[2](b) only the backbone of the protein, and (c) only the side chains.

A rotamer, short for rotational isomer, is a single side chain conformation represented as a set of discrete values, one for each dihedral angle degree of freedom [5]. A rotamer library is a collection of rotamers for each residue type. Rotamer libraries can be backbone-independent [24] or backbone-dependent [25]. The distinctions are made according to whether the dihedral angles of the rotamers and/or their frequencies depend on the local backbone conformation.

The set of rotamers for an amino acid can be seen as a set of statistically significant conformations of the most probable configurations. In the side chain placement problem, the search for the protein

[1] All the images of protein structures displayed in this paper have been made using the Prekin and Mage softwares to construct molecular kinemages from PDB-format coordinate files. These programs are available from http://www.kinemage. biochem.-duke.edu/index.php (accessed: 7 April 2006).

[2] All the proteins used in our research are referenced in this paper using their protein data bank identifier (PDB ID) [23].

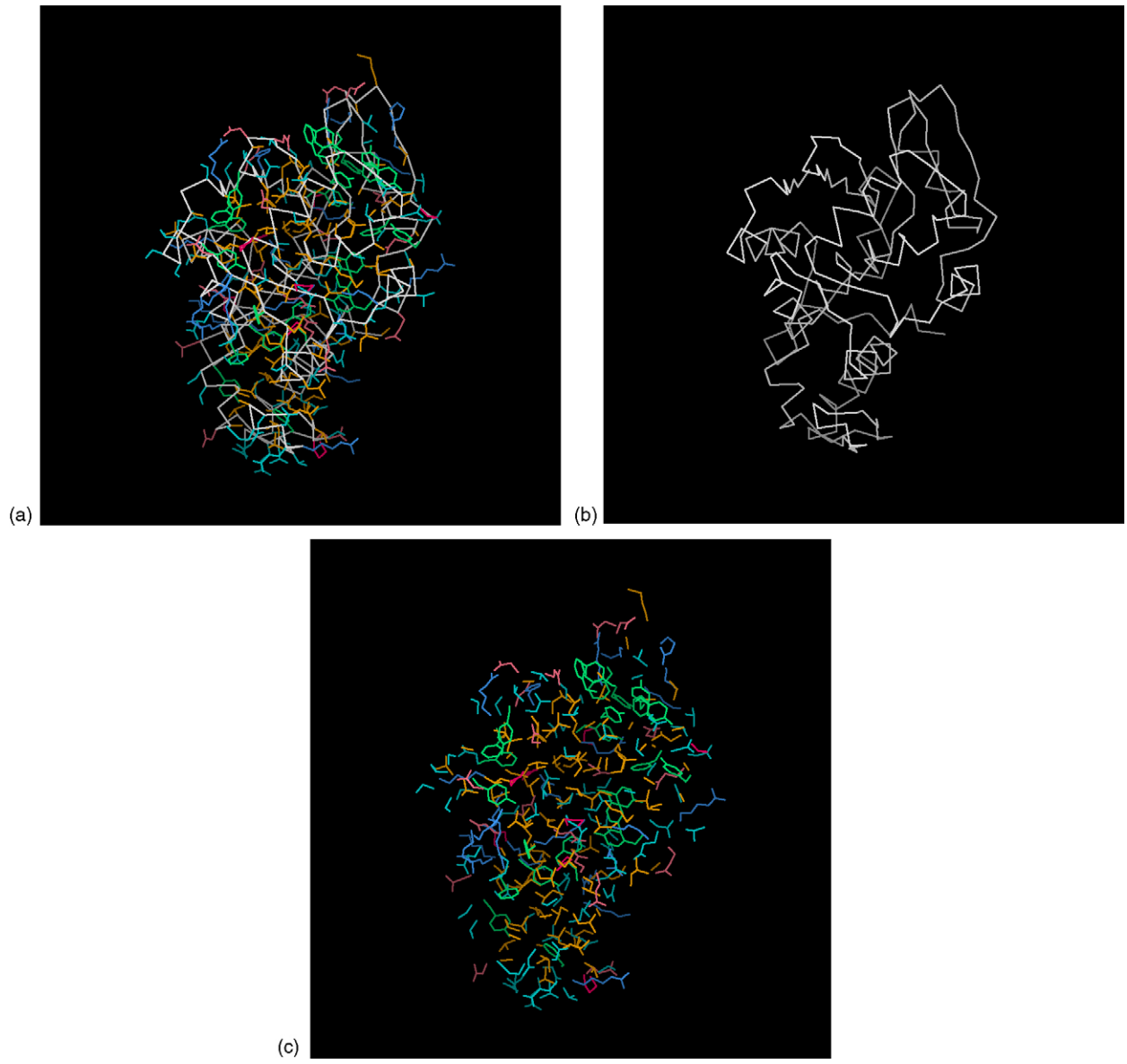

Figure 1 (a) Native structure of the *pdb*1*mr j* protein, (b) backbone of the protein and (c) side chains.

structure is ''reduced'' to the search of a set of rotamers (one for each residue) that optimizes the objective function. Although the side chain placement problem considerably reduces the complexity of the protein structure problem for many proteins, the dimension of the search space remains, in most of the cases, huge. Therefore, the use of brute force algorithms would be unaffordable.

2.2. Fitness functions

The evaluation of a side chain conformation (an assignment of a set of angles for each residue) is usually the combination of several terms that include [26] van der Waals interactions, hydrogen bonds, solvation terms, and terms representing residue secondary structure propensities. Fitness functions have been proposed and tuned taking into consideration protein domain specificities.

We use X_i to represent a discrete random variable. A possible value of X_i is denoted x_i. Similarly, we use $\mathbf{X} = (X_1, \ldots, X_n)$ to represent an n-dimensional random variable and $\mathbf{x} = (x_1, \ldots, x_n)$ to represent one of its possible values. x_i will be interpreted as the rotamer configuration associated with the ith residue.

When the backbone is fixed, the energy of a sequence folded into a defined structure can be expressed [27] as:

$$E(\mathbf{x}) = \sum_{i=1}^{n} E(x_i) + \sum_{i=1}^{n-1} \sum_{j>i}^{n} E(x_i, x_j), \tag{1}$$

where $E(x_i)$ represents the energy interaction between the rotamer and the backbone as well as the intrinsic self-energy of the rotamer. $E(x_i, x_j)$ is the interaction energy between the couple of rotamers. For two set of atoms, the interaction energy is a sum of the pairwise atom interactions. We have adopted the van der Waals energy function as implemented in [28,29]. This energy function approximates the repulsive portion of Lennard—Jones 12—6 potential. It penalizes steric clashes between atoms. Residues that do not interact at all have energy $E(x_i, x_j) = 0$ for every possible rotamer configuration.

The function represented by Eq. (1) is used in this paper to evaluate the quality of the side chain configurations. There are several factors that influence the complexity of the function. These include the number of variables, the number of possible configurations for each variable and the number of interactions.

It is important to notice that, while structure-based pairwise potentials are fast and have shown to be useful for fold prediction, they lack sensitivity to local structure at an atomic level [13]. On the other hand, since certain non-additive energy contributions cannot be treated exactly, this pairwise expression of the energy is just a simplification of the general case [27]. Therefore, some authors [11] have pointed out that the real obstacle for side chain prediction is the definition of appropriate scoring functions. Current approaches give good results for certain types of residues, but not for others. For instance, some steric clashes are not accounted for in the current proposals to approximate energy functions.

Different optimization approaches to optimal side chain prediction have been proposed. Among the most common approaches used for side chain prediction are dead-end elimination (DEE) algorithms [30], the self consistent mean field approach (SCMF) [31], and side chain placement with rotamer library (SCWRL) [5]. Inference-based methods [28,29] can be also used to find the exact solutions of the side chain prediction problem. For a more complete review of these methods, see [27].

3. Estimation of distribution algorithms

Estimation of distribution algorithms are a class of population-based stochastic search algorithms that have been recently applied to different problems of bioinformatics [32—36]. They replace the traditional crossover and mutation operators used in GAs by probabilistic models. These algorithms construct, in each generation, a probabilistic model that estimates the probability distribution of the selected solutions. The probabilistic model must be able to capture, in the form of statistical dependencies, a number of relevant relationships between the variables. The induced probabilistic models are then used to generate solutions during a simulation step. This way, the search leads to promising areas of the search space.

EDAs can be seen as a development of GAs. By recombining a subset of selected solutions, GAs are able to process the information learned during the search, and to orient the exploration to promising areas of the search space. EDAs inherit this attribute but the use of the probabilistic model allows them to explicitly represent the regularities captured during the search. The success of EDAs in the solution of different practical problems has been documented in [19].

The selection method employed by EDAs can be any of those traditionally used by GAs. In the literature, truncation, Boltzmann, and tournament selection are commonly used with EDAs. A characteristic and crucial step of EDAs is the construction of the probabilistic model. These models may differ in the order and number of the probabilistic dependencies that they represent.

Different classifications of EDAs can be used to analyze these algorithms. Relevant to our research is a classification according to the complexity of the models used to capture the interdependencies between the variables [37]. Regarding the way in which learning in the probability model takes place, EDAs can be divided into two classes. One class groups the algorithms that only do a parametric learning of the probabilities, and the other class comprises those algorithms where a structural learning of the model also occurs. Parametric and structural learning are also known as model fitting and model selection. Examples of EDAs belonging to the first class are population based incremental learning (PBIL) [38], compact GA (cGA) [39], the univariate marginal distribution algorithm (UMDA) [20], and the factorized distribution algorithm that uses a fixed model of the interactions in all the generations (FDA) [40]. Examples of EDAs that do a structural learning of the model are, among others, the mutual information maximization for input clustering algorithm (MIMIC) [41], the extended compact GA (EcGA) [42], and EDAs that use Bayesian networks [43—45].

3.1. Univariate marginal distribution algorithm

We will focus on the univariate marginal distribution algorithm (UMDA) [20], an EDA that uses a factorized probability model based on the univariate marginals calculated from the population selected. We introduce the following notation.

We will work with positive probability distributions denoted by $p(\mathbf{x})$. Similarly, given $S \in \{1, 2, \ldots, n\}$, $p(\mathbf{x}_S)$ will denote the marginal probability distribution for $\mathbf{X}_S$.

The univariate model assumes that all variables are independent. The configuration of variable X_i does not depend on the configuration of any other variable. $p(\mathbf{x})$ can be factorized as follows:

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i) \tag{2}$$

Algorithm 1 shows the steps of UMDA. In Algorithm 1, $p_i^s(x_i, t)$ is the marginal probability corresponding to value x_i of variable X_i calculated from the selected population at generation t.

Theoretical results derived for the UMDA [20] expose its relationship with GAs, particularly with GAs that use uniform crossover. Mühlenbein and Mahnig [46] have investigated some of the issues that explain the success of UMDA in the optimization of a wide class of functions.

3.2. UMDA and the transformation of fitness landscape

One of the recognized approaches to the optimization of functions with multiple local minima is based upon hypersurface deformation, in which the function is deliberately altered [47]. These methods try to smoothen the fitness landscape of the function and reduce the number of minima, thereby making the global optimization problem easier.

Given a fitness function $f(\mathbf{x})$, UMDA transforms the original fitness landscape defined by $f(\mathbf{x})$ into a fitness landscape defined by $\tilde{W}(p) = p(\mathbf{x})\, f(\mathbf{x})$, where $p(\mathbf{x})$ is the probability mass function determined by the univariate model of UMDA. It associates a probability to each point of the search space. $\tilde{W}$ denotes the average fitness. This transformation smoothes the rugged fitness landscape of $f(\mathbf{x})$. UMDA converges to the local attractors of the average fitness. If there is a tendency towards the global optimum, UMDA may find it [46]. Although in the fitness landscape defined by $\tilde{W}$ many of the original local optima can appear flattened, there are many factors that influence this transformation; among them the number of local optima of the function and the gap between these and the global optimum point.

Algorithm 1. Pseudocode for UMDA

```
1  Set t ⇐ 0. Generate M points randomly.
2  do {
3     Select a set S of N ≤ M points according to a selection method.
4     Compute the univariate marginal frequencies p_i^s(x_i^c t) of S.
5     Generate M new points according to the distribution p(x^c t + 1) =
      ∏_{i=1}^{n} p_i^s(x_i^c t).
6     t ⇐ t + 1
7  } until Termination criteria are met.
```

4. UMDA approach to the side chain placement problem

In this section, we introduce a method to search the optimal solution of the side chain placement problem. The pseudocode of the method is shown in Algorithm 2.

The algorithm starts by calculating the adjacency matrix that represents the graphical model topology inferred from the backbone structure, as described in [29]. The calculation of the matrices simplifies the evaluation of the solutions by considering only the pairwise interactions that exist between neighbor proteins in the graph.

Then, the number of possible configurations for each residue is calculated using the backbone-dependent rotamer library of [25]. This library includes frequencies, mean dihedral angles and variances as a function of the backbone dihedral angles.

In the next step, we apply the Goldstein elimination criterion [30]. It is based on inequality (3) which is usually employed by DEE algorithms to iteratively eliminate rotamers.

$$E(x_i) - E(x_i') + \sum_{\substack{j=1 \\ j \neq i}}^{n} \min_{x_j} (E(x_i, x_j) - E(x_i', x_j)) > 0 \quad (3)$$

Eq. (3) establishes a sufficient condition [30] for rotamer configuration x_i to be absent from the optimal solution. When no condition that further eliminates rotamers can be established, the algorithm stops. If the space of remaining configurations is small enough, the remaining combinations are searched using exhaustive enumeration.

This step considerably contributes to reduce the dimension of the search space, but for medium and large proteins, research remains unaffordable for exact methods. The Goldstein elimination criterion used by DEE is an important component of other optimization algorithms (e.g. SCWRL).

When the application of the Goldstein elimination criterion cannot reduce the number of variable values further, we determine which of the residues that have more than one rotamer configuration are. The corresponding variables are the only ones to be optimized.

4.1. Problem representation and fitness function

We use the following problem representation for the UMDA search: each residue will be represented by a random variable X_i. The number of values of each variable will correspond to the number of possible rotamer configurations for the corresponding residue i (i.e. $x_i = 1, \ldots, K_i$, where K_i is the number of feasible rotamer configurations for residue i).

As the fitness function, the energy function in Eq. (1) is used. The probability model represented by Eq. (2) will represent the probability of a given side chain configuration.

4.2. UMDA parameters

We set the population size $M = 5000$. Truncation selection was applied. In this type of selection, the best $N = TM$ individuals, according to their function evaluations, are selected. T is a parameter called truncation parameter. It determines the selection pressure of the algorithm. In our implementation, $T = 0.15$.

We use best elitism, a replacement strategy where the population selected at generation t is incorporated into the population of generation $t + 1$. Thus, only $M - N$ individuals are generated at each generation except for the first one. The stop criteria considered are that the optimum has been found (when it is known), that the number of different solutions is below 10, or that the maximum number of generations (5000) has been reached. The algorithm, whose pseudocode is shown in Algorithm 2, has been implemented in C++ language.

Algorithm 2. Proposed algorithm for side chain placement

1. Calculate the spatial adjacency matrix that describes the graphical model topology.
2. Calculate the energy interaction between neighboring rotamers.
3. Apply the Goldstein criterion to simplify the number of rotamer configurations.
4. Apply UMDA to find the candidate best solution.

4.3. Computational cost of the algorithm

The analysis of the computational cost of the algorithm can be divided into three stages: (1) calculation of the adjacency matrix, (2) application of the Goldstein criterion, and (3) application of the UMDA approach.

The calculation of the adjacency matrix depends on the distances between every pair of residues. The calculation of these distances has complexity $O(n^2)$. The complexity of dead-end algorithms is analyzed in [10]. The principal determinant of the computational time of DEE is the number of rotamer pairwise interaction energies that must be retrieved for an entire round of eliminations. Let $|K_i|$ be the cardinality of variable X_i. The complexity of the DEE step is $O(r_{\text{MEAN}}^3 n^2)$, where $r_{\text{MEAN}} = 1/n \sum_{i=1}^{n} |K_i|$.

As can be seen from the UMDA pseudocode shown in Algorithm 1, UMDA has a simple structure, with

few and clearly defined steps. These facts allows its computational cost to be calculated.

First, we consider the computational complexity of each generation of UMDA. The initialization step of UMDA consists in assigning the values to all the individuals in the initial population. It has complexity nM. The computational complexity of the evaluation step depends on the number of residues and of interacting neighbors in the graph. Let $|D|$ be the number of edges represented by the adjacency matrix. Then, the running time complexity of this step is $O(n+|D|)$. The complexity of the UMDA selection steps depends on the selection method used. For truncation selection, complexity is related to the ordering of the solutions. In the worst case, the complexity of this step is $M\log(M)$.

The complexity of the learning step is $O(Nn)$. This is the cost of inspecting the values of every variable of the N selected solutions. The complexity of the sampling step is $O((M-N)nr_{MAX})$, where $r_{MAX}=\max_{i\in\{1,\ldots,n\}}|K_i|$ is the highest cardinality among the variables. This value corresponds to the maximum number of rotamer configurations a residue can have.

The actual number of generations needed by UMDA to converge is problem dependent. In general, this parameter is very difficult to estimate, although theoretical results for some classes of functions are available [48]. Let G be the maximal number of allowed generations. The complexity of the UMDA for the side chain problem can be estimated as $O(GM(nr_{MAX}+|D|))$, and the total complexity of the introduced proposal is $O(r^3_{MEAN}n^2+GM(nr_{MAX}+|D|))$.

5. Algorithmic tests

In this section, we present the results of the application of Algorithm 2 to a large set of protein instances. First, we introduce the protein benchmark used for our algorithmic tests. Then, we explain how the experiments were designed, as well as the numerical results of the comparison between UMDA and other optimization algorithms.

5.1. Protein benchmark

To validate our algorithm we have used a set of 463 protein structures.[3] The dataset corresponds to 463 X-ray crystal structures with a resolution better than or equal to 2 Å, R factor below 20%, and mutual sequence identity less than 50%. Each protein consisted of 1—4 chains and up to 1000 residues.

Table 1 Details of the protein instances

Database	Small	Large	Dimer
Total	325	45	93
Min. size	7	311	124
Max. size	267	704	1982
Goldstein	11	0	0
SPRINT conv.	314	31	67
UMDA	284	10	16

For comparison, we have used the Side-chain PRediction INference Toolbox (SPRINT)[4] which is an implementation of the max-product belief propagation algorithm [29]. To simplify the overload related to the calculation of the adjacency matrices, and to focus on the study of the optimization algorithm, we have used the adjacency matrices available from the SPRINT implementation [29].

The database of proteins is divided into three groups: small, large, and dimer proteins. We have used this classification in our experiments. The total number of instances for each group, as well as the minimum and maximum size of the instances in each group are shown in Table 1. In the case of the dimer set, each protein can contain up to four chains of residues. Fig. 2(a), shows the backbone structures of the four chains that form the *pdb1d2e* protein. This is the largest protein in the dimer set.

Additionally, as a preprocessing step, we have determined, for each group, the instances for which the Goldstein criterion eliminates all configurations but one, and those instances for which the SPRINT algorithm converges. This information is summarized in Table 1 together with the number of instances of each group where UMDA is able to find the known optimal solution in at least one of fifty runs.

As can be observed in Table 1, the application of the Goldstein criterion can only solve instances in the first group. Moreover, SPRINT does not converge for 3% of the instances in the small class, 31% of the instances in the large class and 32% of the instances in the dimer class. For the small class of instances, the protein structures obtained from the instances for which SPRINT converged are known to be the optimal ones [29].

[3] These instances have been obtained from the page of Chen Yanover: http://www.cs.huji.ac.il/~cheny/proteinsMRF.html (accessed: 7 April 2006).

[4] http://www.cs.huji.ac.il/~cheny/sprint.html(accessed: 7 April 2006).

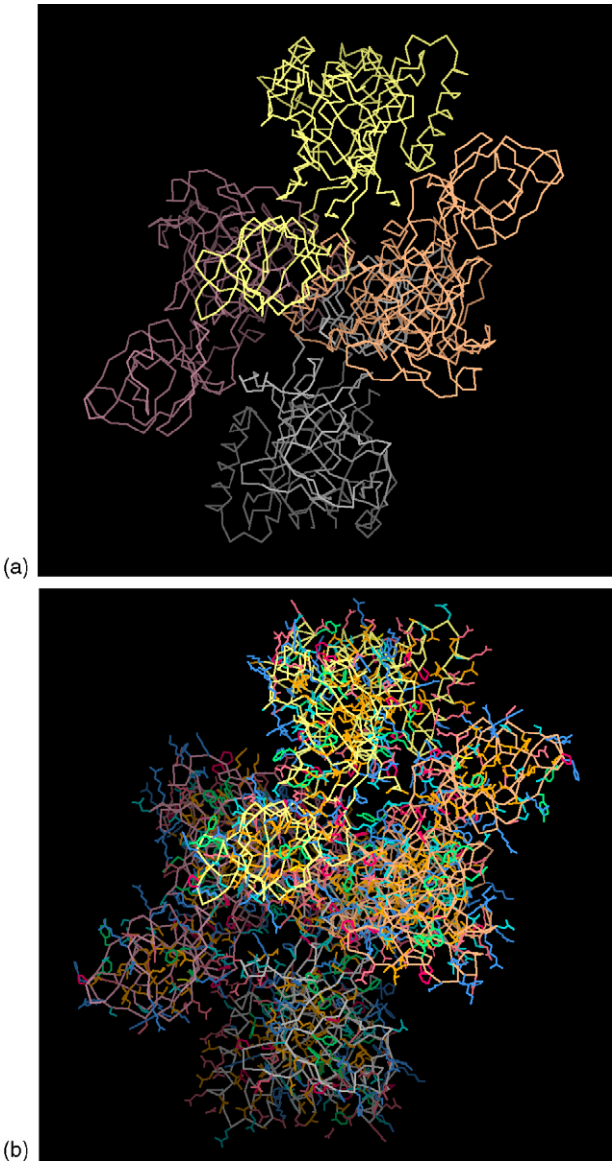

Figure 2 (a) Native backbone structure corresponding to the *pdb*1*d*2*e* protein and (b) side chain configuration found by UMDA.

5.2. Design of the algorithmic tests

Initial algorithmic tests intend to evaluate whether UMDA was able to achieve the optimum for the set of small sequences. We have excluded from the experiments the instances for which the Goldstein criterion eliminates all configurations but one. For the rest of the instances (314), we run the UMDA and find the best solution that the algorithm can find in fifty runs. The last row of Table 1 shows the number of protein instances for which UMDA found the known optimal solution in at least one of the fifty runs.

Results achieved by SPRINT are used as a reference for comparison. For all the instances, we have also calculated the structures found by SCWRL (version 3.0). In [28], the energies obtained by SCWRL (version 2.9) were reported to be strictly higher than those found by SPRINT in the small class of instances. Unfortunately, the SCWRL (version 3.0) implementation does not provide the energy values corresponding to solutions calculated by the algorithm. Therefore, in this paper we constrain the comparison to the results achieved by SPRINT.

To evaluate the performance of UMDA, we use the measures PD (4) and PE (5).

$$\mathrm{PD}(\mathbf{x}) = \frac{\sum_{i=1}^{n} I(x_i, x_i^{\mathrm{opt}})}{n} \tag{4}$$

$$\mathrm{PE}(\mathbf{x}) = \frac{E(\mathbf{x}) - E(\mathbf{x}^{\mathrm{opt}})}{E(\mathbf{x}^{\mathrm{opt}})} \tag{5}$$

PD is the percentage, with respect to the number of side chain residues, of the number of residues different to the best known solution. In (4), $I(x_i, x_i^{\mathrm{opt}})$ is 1 if the side chain rotamer configurations of the solution $\mathbf{x}$ and $\mathbf{x}^{\mathrm{opt}}$ are different for residue i. PE is the percentage, with respect to the energy of the best known solution, of the energy gap between the obtained energy and the energy of the best known solution.

For the sets of instances, we analyze the best and average performance of the algorithm. The best and average performances are respectively calculated using the best solution $\mathbf{x}^{\mathrm{best}}$, found in the 50 experiments ($\mathrm{PD}(\mathbf{x}^{\mathrm{best}})$, $\mathrm{PE}(\mathbf{x}^{\mathrm{best}})$), and the average $\overline{\mathrm{PD}}$, $\overline{\mathrm{PE}}$ of the evaluating measures calculated from the solutions found in all the experiments ($\overline{\mathrm{PD}} = (\sum_{i=1}^{50} \mathrm{PD}(\mathbf{x}^i))/50$, $\overline{\mathrm{PE}} = (\sum_{i=1}^{50} \mathrm{PE}(\mathbf{x}^i))/50$).

5.3. Numerical results

Fig. 3 shows (from left to right, top to bottom) the histograms corresponding to $\mathrm{PD}(\mathbf{x}^{\mathrm{best}})$, $\overline{\mathrm{PD}}$, $\mathrm{PE}(\mathbf{x}^{\mathrm{best}})$ and $\overline{\mathrm{PE}}$ for the small set of instances. Similarly, Figs. 4 and 5, respectively show the same measures for the large and dimer sets.

An analysis of the histograms of $\mathrm{PD}(\mathbf{x}^{\mathrm{best}})$ shows that the vast majority of solutions are less than 4% of the residues apart from the best known solutions. This difference increases when $\overline{\mathrm{PD}}$ is considered. However, in this case, as well, the vast majority of solutions are only 3% away from the best known solution.

A similar behavior can be observed in the case of the energy gap. Nevertheless, for the energy, the best and average energies are more concentrated around the optimal energy. This fact reflects that solutions with a higher distance in terms of the number of residues may be closer in the energy landscape.

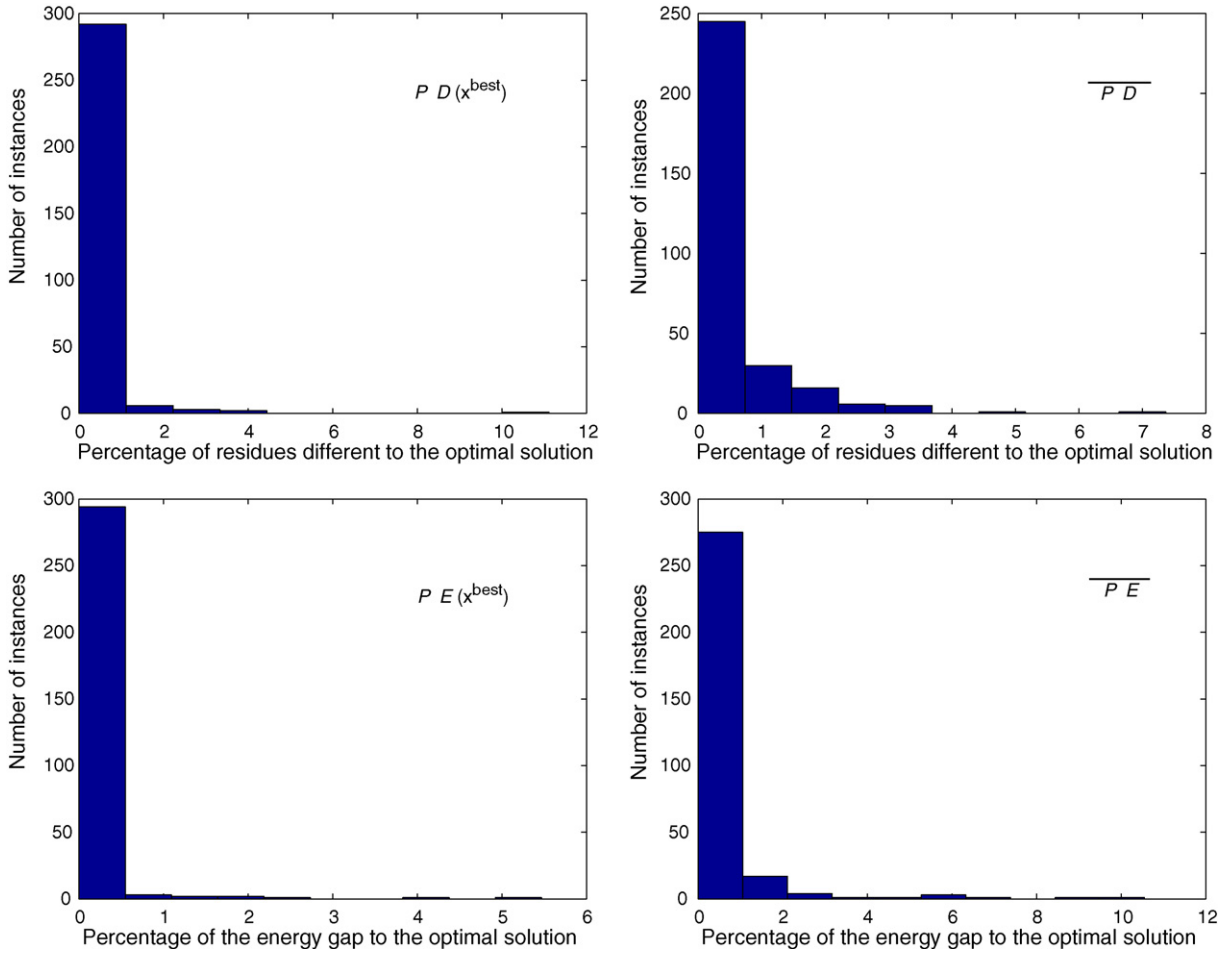

Figure 3 UMDA results for the small set of instances. From left to right, top to bottom, the histograms corresponding to $PD(\mathbf{x}^{best})$, $\overline{PD}$, $PE(\mathbf{x}^{best})$ and $\overline{PE}$ are presented.

5.4. Comparison with other methods

In the following experiments, we concentrate on those instances for which the inference-based algorithm did not converge. As the optimal solutions are unknown for these instances, they constitute a challenge for optimization methods. The first column of Tables 2–4 shows the proteins for which the max-product belief propagation algorithm (SPRINT in the tables) did not converge from the set of small, large and dimer proteins. Columns 2 and 3, respectively, provide the remaining number of residues after the application of the DEE (n) step,[5] and the average number of rotamer configurations of the variables ($\overline{K_i}$). The energies corresponding to the structure found by SPRINT (f_{SPRINT}) and UMDA (f_{UMDA}), which in this case is the energy of the best solution, are shown in columns 4 and 5, respectively. The energy values have been normalized using the original number of residues of each protein. The best energy values corresponding to each instance appear in bold.

The last two columns show the root mean square distances calculated between the positions of the structures found by the max-product belief propagation algorithm (r_{SPRINT}) and UMDA (r_{UMDA}), and the positions of the native structure side chains. The inclusion of these values intends to evaluate the predictions obtained in comparison to the real protein structure. However, there is no total correspondence between the root mean square distance and the function evaluation used during the optimization process. The best root mean square distance value corresponding to each instance appears underlined.

Table 2 shows that UMDA is able to find solutions better than SPRINT in only one of the instances of the small set of solutions for which SPRINT did not converge. Nevertheless, in the case of the large and dimer sets of solutions, the solutions achieved by

[5] For simplicity, we also call n to the number of remaining residues after DEE. However, the application of DEE determines an important reduction of the initial number of residues.

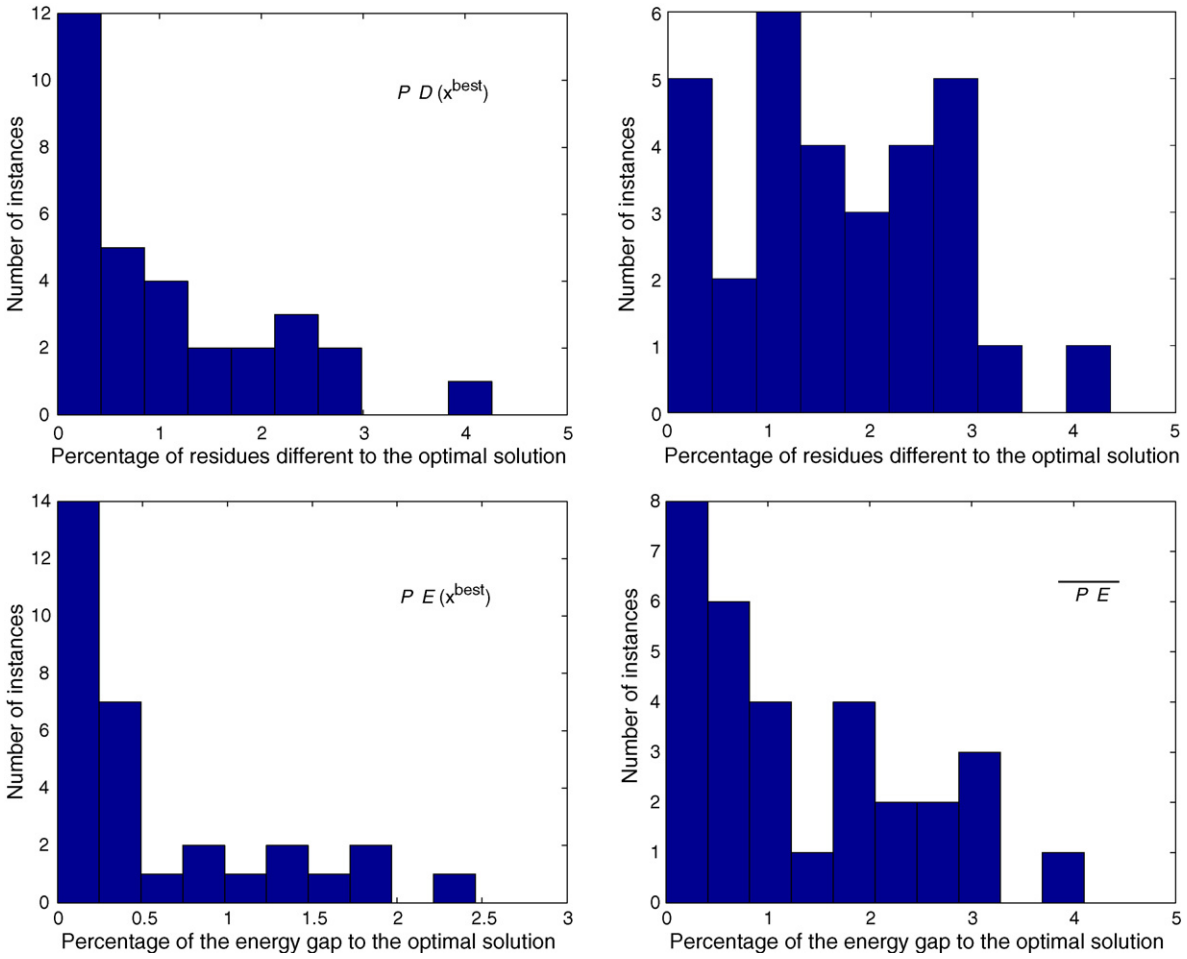

Figure 4 UMDA results for the large set of instances. From left to right, top to bottom, the histograms corresponding to PD($\mathbf{x}^{best}$), $\overline{PD}$, PE($\mathbf{x}^{best}$) and $\overline{PE}$ are presented.

UMDA were better than or equal to those achieved by SPRINT in 10 out of 14 instances, and 21 out of 26 instances, respectively. These results show that UMDA is an alternative for those situations where inference-based methods cannot converge. Considering the root mean square distances, the number of instances were UMDA achieved results equal to, or better than, the SPRINT algorithm for small, large and dimer instances were, respectively, 7 out of 11, 7 out of 14, and 12 out of 26. Notice that the discordance between the results achieved by the algorithm considering the root mean square distance and the energy might be due to the fact that the energy function takes into account other important elements that measure the quality of the prediction, and not only the distances between the atoms.

Nevertheless, considering each protein separately, the correlation between the energy and RMSD holds. In order to illustrate this fact, we have found the protein side chain structures corresponding to 50 random solutions of the protein pdb1d2e (Fig. 2). For these solutions, we have found the energy and the RMSD. Additionally, we have found the energies and RMSD for the best UMDA solutions in 50 independent runs. In Fig. 6, the relationship between the energies and the root mean square deviation (RMSD) values for the 100 solutions is plotted. The correlation between the energy and RMSD is 0.998. Notice the important energy and RMSD gaps between the random solutions and those found by UMDA. To illustrate the quality of the solutions found, Fig. 2(b) shows the best side chain configuration of the *pdb1d2e* protein found by UMDA.

5.5. Analysis of the convergence time

In Section 4.3 we have analyzed the computational cost of UMDA for the protein side chain placement problem. We have acknowledged that computational time critically depends on the number of generations needed by the algorithm to achieve convergence. In this section, we investigate the relationship between the size of the proteins and

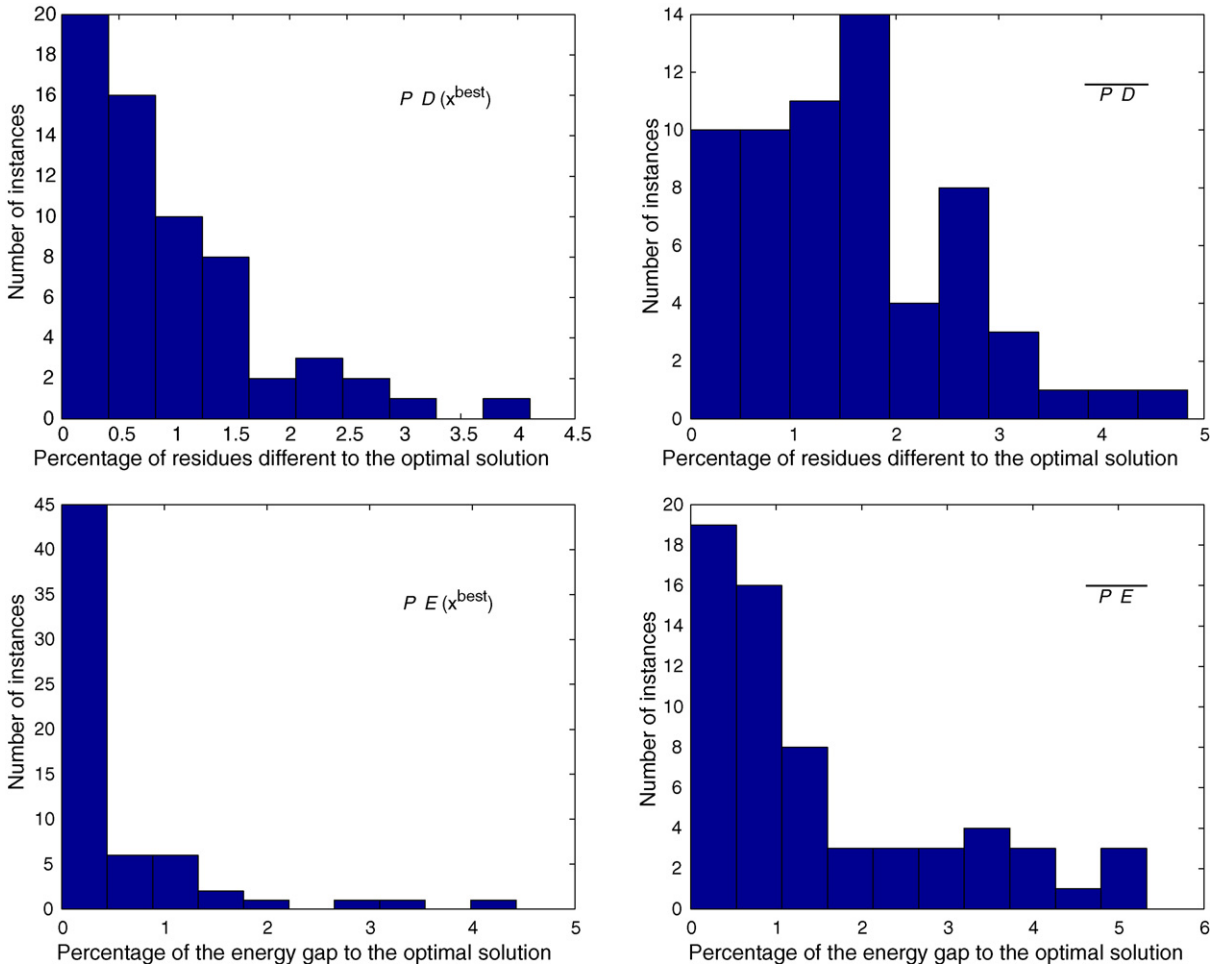

Figure 5 UMDA results for the dimer set of instances. From left to right, top to bottom, the histograms corresponding to PD($\mathbf{x}^{best}$), $\overline{PD}$, PE($\mathbf{x}^{best}$) and $\overline{PE}$ are presented.

the average time needed by UMDA to reach convergence. Although there are other factors that have influence on the convergence of the algorithm, the number of residues can be useful to obtain an initial estimate of the time of convergence.

For our analysis, we have used all instances where inference-based methods converged. We have calculated the average of the time needed by UMDA to reach convergence (i.e. to fulfil one of the termination criteria) in the fifty experiments. Fig. 7 plots the

Table 2 Results achieved by the different algorithms for the subset of small instances for which SPRINT does not converge

pdb file	n	$\overline{K_i}$	f_{SPRINT}	f_{UMDA}	r_{SPRINT}	r_{UMDA}
pdb1buu	27	2.56	**1.5891**	**1.5891**	1.2551	1.2551
pdb1bv1	21	2.52	**0.9891**	**0.9891**	1.2359	1.2359
pdb1ema	45	4.35	**1.4839**	**1.4839**	1.0859	1.0859
pdb1et9	64	4.12	**1.2326**	1.2345	1.3090	1.3251
pdb1h6h	43	4.13	**0.7836**	**0.7836**	1.3986	1.3986
pdb1hh8	63	4.33	**2.2398**	2.2705	1.4601	1.5222
pdb1mrj	70	5.53	1.1219	**1.1186**	1.2253	1.2247
pdb2fcr	51	4.35	**1.6523**	**1.6523**	1.3366	1.3366
pdb2ilk	35	3.17	**0.9757**	**0.9757**	1.6039	1.6039
pdb2tir	28	5.28	**1.0625**	1.0644	1.1198	1.0833
pdb3kvt	35	4.91	**1.7446**	1.7821	0.8470	1.0111

Table 3 Results achieved by the different algorithms for the subset of large instances for which SPRINT does not converge

pdb file	n	$\overline{K_i}$	f_{SPRINT}	f_{UMDA}	r_{SPRINT}	r_{UMDA}
pdb1crz	75	3.84	1.9958	**1.9886**	1.2988	1.2968
pdb1ddt	146	4.22	1.7354	**1.7080**	1.3296	1.3105
pdb1dpe	185	4.69	2.1071	**1.6760**	1.2889	1.2665
pdb1e39	127	5.68	**1.2365**	**1.2365**	1.0239	1.0239
pdb1f5n	166	4.68	**1.1441**	**1.1441**	1.2480	1.2480
pdb1gsk	208	5.36	2.7801	**2.1122**	1.2031	1.2023
pdb1h3n	318	5.62	2.3769	**2.3635**	1.3473	1.3458
pdb1jy1	144	3.68	2.3114	**2.3108**	1.2760	1.2799
pdb1kmo	241	5.68	1.7572	**1.7020**	1.2901	1.3345
pdb1kwh	207	4.88	**2.2622**	2.2927	1.3838	1.4247
pdb1n5u	155	3.60	**1.6874**	1.6909	1.2845	1.2911
pdb1nqe	189	4.70	1.3573	**1.2479**	1.2556	1.3217
pdb1nr0	175	3.64	**1.7804**	1.8531	1.0409	1.0733
pdb2nap	292	5.26	**1.7919**	1.8666	1.2354	1.2656

dependence between the protein size and the time needed for convergence (in seconds). Additionally, and in order to estimate the scalability of the algorithm, the points corresponding to each set of proteins have been fitted using second-order polynomials. It can be seen in the figure that the complexity is near quadratic in the number of variables.

The time needed by UMDA in comparison to the other parts of the algorithm critically depends on the instances. From the curves, it can be seen that the most complex instances are those that belong to the large set. This facts seems to indicate that, considering a fixed number of residues, the complexity of the UMDA approach to the side chain placement problem

Table 4 Results achieved by the different algorithms for the subset of dimer instances for which SPRINT does not converge

pdb file	n	$\overline{K_i}$	f_{SPRINT}	f_{UMDA}	r_{SPRINT}	r_{UMDA}
pdb1b25	916	5.62	2.4318	**2.4284**	1.2349	1.2628
pdb1d2e	281	3.75	1.5534	**1.3853**	1.2218	1.2087
pdb1dxr	353	4.72	2.5651	**1.7747**	1.2828	1.2683
pdb1dz4	288	5.28	1.3078	**1.2529**	1.1513	1.1881
pdb1e3d	454	4.23	1.9915	**1.8614**	1.1147	1.1259
pdb1e61	479	4.68	1.5905	**1.5889**	1.2375	1.2133
pdb1e6p	365	5.32	2.4429	**2.0839**	1.2812	1.3101
pdb1f60	123	4.59	1.2192	**1.2050**	1.2723	1.2548
pdb1fmj	294	4.59	3.1693	**1.7924**	1.2892	1.2370
pdb1fn9	239	5.77	**1.5681**	1.5707	1.3156	1.2304
pdb1fnn	240	4.45	1.1634	**1.1182**	1.2009	1.2358
pdb1giq	265	4.49	1.1335	**1.0754**	1.1998	1.2068
pdb1h0h	934	4.41	**2.4002**	2.4465	1.1085	1.2040
pdb1h3f	206	4.85	1.3477	**1.1468**	1.3351	1.3421
pdb1h4r	227	4.14	1.6786	**1.5519**	1.3567	1.3200
pdb1h80	229	3.88	1.4863	**1.3937**	1.1176	1.0630
pdb1hhs	728	5.97	1.6698	**1.5724**	1.3171	1.2802
pdb1iqc	288	3.88	1.5290	**1.4792**	1.2558	1.2534
pdb1j3b	289	5.06	1.8849	**1.8715**	1.3888	1.3999
pdb1j8f	329	4.10	**1.1815**	1.1874	1.2798	1.2932
pdb1jmx	285	4.66	2.0865	**2.0543**	1.2565	1.2891
pdb1lax	268	4.88	**1.6059**	1.7666	1.1864	1.2136
pdb1lqa	244	4.00	1.6374	**1.2928**	1.1911	1.2171
pdb1lsh	350	5.26	1.1791	**1.1719.7**	1.3524	1.2376
pdb1np7	424	5.24	**2.0621**	2.0831	1.3937	1.4025
pdb1tki	164	4.29	1.8811	**1.4908**	1.2426	1.2226

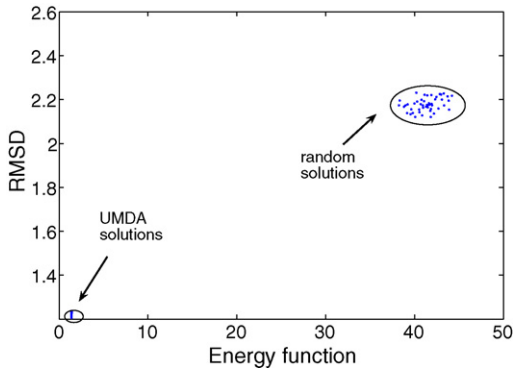

Figure 6 Energies and RMSD values corresponding to 50 random solutions and 50 solutions found by UMDA for the protein side chain placement of protein pdb1d2e.

can be decreased when there is more than one chain in the protein, as is the case of dimer instances. Even if there are interactions between residues that belong to different chains, most of the interactions are concentrated between the residues of the same chain. Therefore, complexity might depend more on the size of the largest chain than on the total number of residues in the dimer.

6. Relationship with previous research

The UMDA approach to side chain placement has a number of contact points with previous algorithms used to solve this problem. The analysis of these similarities helps to illustrate the different aspects of the algorithm and the way they contribute to an efficient search. This analysis is also relevant for the identification of other possible applications of EDAs to Computational Biology.

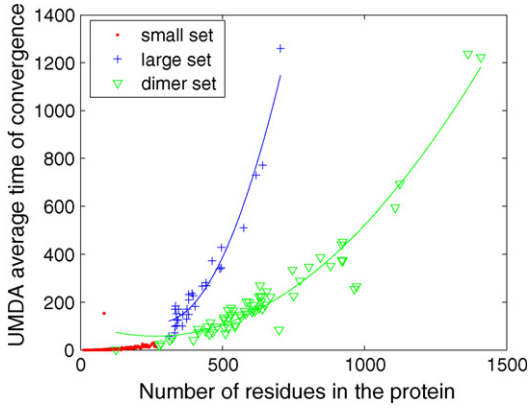

Figure 7 Dependence between the number of residues in all instances and the UMDA time of convergence. Only the instances where inference-based methods converged are included. Additionally, the points are fitted using second order polynomials.

One important aspect of EDAs is their attempt to focus the search in the space of promising solutions. This is a goal shared by evolutionary algorithms, as is the case of the one presented in [49]. This algorithm eliminates values of the variables (corresponding to rotamer configurations) that have not been found within the best percentage of the population, but can be found within the worst population solutions. The way to identify these values is to contrast the best and worst selected sets.

UMDA pursues the same goal, but in a different way. In this case, a set of best individuals is also selected. Nevertheless, no comparison is made with any other selected set. Instead, a probability model of the solutions is constructed. Variables values that are absent in the population have a very low probability in this model. Additionally, the model keeps information about the frequency of each possible rotamer configuration. Configurations more likely to be among the best solutions have a higher probability. The probability model used by UMDA extracts more statistical information than the one implicitly manipulated by GAs.

Another aspect of UMDA is the simplicity of the univariate model it uses. This model is similar to the mean-field model used by SCMF. Obviously, the models used by UMDA and SCMF are only rough approximations of the underlying probability distributions. In the mean-field approximation, the univariate marginals are considered to be variables. UMDA computes the marginals from samples. The relationship between the mean-field approach and the UMDA has been studied in [50].

Compared to SCMF, the strength of UMDA lies in its sampling procedure which adds to the stochastic character of the search. By sampling a set of solutions from the univariate model, the algorithm can explore a higher number of points. The deterministic nature of SCMF can be a drawback for the search. As SCMF must converge to a single configuration of rotamers, the convergence is made difficult by increasing the number of rotamer configurations. In these cases, the probability for SCMF not to be able of converge increases [27]. UMDA can be seen as a non-deterministic SCMF algorithm where a probabilistic model is learned at every iteration, and sampling replaces the role of the search for consistency as is procured by SCMF.

Finally, we consider the relationship between UMDA and inference-based methods. EDAs and optimization algorithms that use inference-based methods are two different ways to use graphical models in optimization. These approaches can be combined to obtain more efficient algorithms.

The main advantage of EDAs over inference-based methods is that the former do not need

previous information about the structure of the problem. Propagation algorithms needed by inference-based methods rest on a given graphical model. In the case of side chain placement, this model corresponds to the adjacency matrix. UMDA learns the parameter of its model from the data. Clearly, the graphical structure constructed from the adjacency matrix stores more information than the univariate model. Therefore, UMDA should not be seen as an alternative to inference-based methods in every scenario. It remains a suitable alternative when inference-based methods do not converge.

7. Conclusions

We have proposed the use of UMDA as a stochastic optimization algorithm for the side chain placement problem. We have carried out a systematic study of the algorithm using a large set of protein instances and comparing the results with state-of-the-art algorithms. For a number of difficult instances where inference algorithms do not converge, it has been shown that UMDA is able to find better structures. We have studied the expected and best performance of the method, considering the energy values as well as the number of correct rotamers of the found solutions. Additionally, we have presented a theoretical and empirical analysis of the computational cost of the algorithm introduced.

We have pointed out the links between some of the most used current methods for side chain prediction and our proposal. UMDA can be seen from the perspective of a good amount of successful applications of evolutionary techniques. However, the simplicity of the UMDA implementation contrasts with common GA implementations that exhibit intricate, and sometimes costly, genetic operators. We have shown that this simplicity makes UMDA suitable for theoretical analysis and enables an estimation of the time complexity of the algorithm for the side chain problem.

Acknowledgements

The authors thank Chen Yanover for useful comments on the side chain placement problem and for providing the set of instances used in our experiments. The authors are also grateful to Alex Mendiburu and Dr. Jose Miguel for providing part of the computational resources used in our experiments. The authors thank two anonymous referees for constructive comments that helped to improve the paper. This work was supported by the SAIOTEK-Autoinmune (II) 2006 and Etortek research projects from the Basque Government. It has been also supported by the Spanish Ministerio de Ciencia y Tecnología under grant TIN 2005-03824. The SGI/IZO-SGIker UPV/EHU (supported by the Spanish Program for the Promotion of Human Resources within the National Plan of Scientific Research, Development and Innovation—Fondo Social Europeo and MCyT) is gratefully acknowledged for generous allocation of computational resources.

References

[1] Al-Lazikani B, Jung J, Xiang Z, Honig B. Protein structure prediction. Curr Opin Chem Biol 2001;5(1):51—6.
[2] Dill KA. Theory for the folding and stability of globular proteins. Biochemistry 1985;24(6):1501—9.
[3] Kolinski A, Skolnick J. Reduced models of proteins and their applications. Polymer 2004;45(2):511—24.
[4] Mongea A, Lathropa EJP, Gunna JR, Shenkina PS, Freisnera RA. Computer modeling of protein folding: conformational and energetic analysis of reduced and detailed protein models. J Mol Biol 1995;247(5):995—1012.
[5] Dunbrack RL. Rotamer libraries in the 21st century. Curr Opin Struct Biol 2002;12:431—40.
[6] Kraemer-Pecore CM, Wollacott AM, Desjarlais JR. Computational protein design. Curr Opin Chem Biol 2001;5:690—5.
[7] Lazar GA, Desjarlais JR, Handel TM. De novo protein design of the hydrophobic core of ubiquitin. Protein Sci 1997; 6:1167—78.
[8] Rohl CA, Strauss CEM, Misura K, Baker D. Protein structure prediction using Rosetta. Methods Enzymol 2004;383:66—93.
[9] Dandekar T, Köenig R. Computational methods for the prediction of protein folds. Biochim Biophys Acta (BBA) - Protein Struct Mol Enzymol 1997;1343(1):1—15.
[10] Canutescu AA, Shelenkov AA, Dunbrack RL. A graph-theory algorithm for rapid protein side-chain prediction. Protein Sci 2003;12:2001—14.
[11] Liang S, Grishin NV. Side-chain modeling with an optimized scoring function. Protein Sci 2002;11:322—31.
[12] Liu Z, Li W, Liang S, Han Y, Lai L. Beyond rotamer library: genetic algorithm combined with disturbing mutation process for upbuilding protein side-chains. Proteins: Structure Funct Genet 2003;50:49—62.
[13] Pokala N, Handel TM. Review: protein design—where we were, where we are, where we're going. J Struct Biol 2001; 134:269—81.
[14] Lee C, Subbiah S. Prediction of protein side-chain conformation by packing optimization. J Mol Biol 1991;217:373—88.
[15] Shenkin PS, Farid H, Fetrow JS. Prediction and evaluation of side-chain conformations for protein backbone structures. Proteins: Structure Funct Genet 1998;26(3):323—52.
[16] Vasquez M. Modeling side-chain conformation. Curr Opin Struct Biol 1996;6(2):217—21.
[17] Pierce NA, Winfree E. Protein design is NP-hard. Protein Eng 2002;15(10):779—82.
[18] Ponder JW, Richard FM. Tertiary templates for proteins. Use of packing criteria in the enumeration of allowed sequence for different structure classes. J Mol Biol 1987;193:775—91.
[19] Larrañaga P, Lozano JA, editors. Estimation of distribution algorithms. A new tool for evolutionary computation. Boston/Dordrecht/London: Kluwer Academic Publishers; 2002.

[20] H. Mühlenbein, G. Paaß, From recombination of genes to the estimation of distributions. I. Binary parameters. In: Eiben AE, Bäck T, Schoenauer M, Schwefel HP, editors. parallel problem solving from nature—PPSN IV, ser. Lecture Notes in Computer Science, vol. 1141. Berlin: Springer Verlag; 1996. p. 178—87.
[21] Goldberg DE. Genetic algorithms in search, optimization and machine learning Reading, MA: Addison-Wesley; 1989.
[22] Holland JH. Adaptation in natural and artificial systems Ann Arbor, MI: University of Michigan Press; 1975.
[23] Berman HM, Westbrook J, Feng Z, Gilliland G, Bhat TN, Weissig H, Shindyalov IN, Bourne PE. The protein data bank. Nucleid Acid Res 2000;28:235—42.
[24] Tuffery P, Etchebest C, Hazout S, Lavery R. A new approach to the rapid determination of protein side chain conformations. J Biomol Struct Dynam 1991;8:1267—89.
[25] Dunbrack RL, Cohen FE. Bayesian statistical analysis of protein side-chain rotamer preferences. Protein Sci 1997;6(8):1661—81.
[26] Wernisch L, Hery S, Wodak S. Automatic protein design with all atom force-fields by exact and heuristic optimization. J Mol Biol 2000;301(3):713—36.
[27] Voigt CA, Gordon DB, Mayo SL. Trading accuracy for speed: a quantitative comparison of search algorithms in protein sequence design. J Mol Biol 2000;299(3):799—803.
[28] Yanover C, Weiss Y. Approximate inference and protein-folding. In: Becker S, Thrun S, Obermayer K, editors. Advances in neural information processing systems 15. Cambridge, MA: MIT Press; 2003. p. 1457—64.
[29] Yanover C, Weiss Y, Approximate inference for side-chain prediction, submitted to Neural Computation journal 2004.
[30] De Maeyer M, Desmet J, Lasters I. The dead-end elimination theorem: Mathematical aspects, implementation, optimizations, evaluation, and performance. Methods Mol Biol 2000;143:265—304.
[31] Koehl P, Delarue M. Building protein lattice models using self consistent mean field theory. J Chem Phys 1998;108:9540—9.
[32] R. Blanco, P. Larrañaga, I. Inza, and B. Sierra, Selection of highly accurate genes for cancer classification by estimation of distribution algorithms. In: Lucas P, van der Gaag L, Abu-Hamma A, editors, Proceedings of the Workshop Bayesian Models in Medicine held within AIME 2001; 2001. p. 29—34 [online]. Available: http://www.csd.abdn.ac.uk/∼plucas/bayesian.pdf (accessed: 7 April 2006).
[33] Peña JM, Lozano JA, Larrañaga P. Unsupervised learning of Bayesian networks via estimation of distribution algorithms: an application to gene expression data clustering. Int J Uncertainty Fuzziness Knowledge-Based Syst 2004;12(1): 63—82.
[34] Saeys Y, Degroeve S, Aeyels D, Rouzé P, Van de Peer Y. Feature selection for splice site prediction: a new method using EDA-based feature ranking. BMC Bioinformat 2004;5: 64—75.
[35] Saeys Y, Degroeve S, Aeyels D, Van de Peer Y, Rouzé P. Fast feature selection using a simple estimation of distribution algorithm: a case study on splice site prediction. Bioinformatics 2003;19(2):ii179—88.
[36] Santana R, Larrañaga P, Lozano JA, Protein folding in two-dimensional lattices with estimation of distribution algorithms, In: Proceedings of the First International Symposium on Biological and Medical Data Analysis, ser. Lecture Notes in Computer Science, vol. 3337. Barcelona, Spain: Springer Verlag; 2004. p. 388—98.
[37] Larrañaga P. An introduction to probabilistic graphical models. In: Larrañaga P, Lozano JA, editors. Estimation of distribution algorithms. A new tool for evolutionary computation. Boston/Dordrecht/London: Kluwer Academic Publishers; 2002. p. 25—54.
[38] S. Baluja, Population-based incremental learning: a method for integrating genetic search based function optimization and competitive learning. Pittsburgh, PA: Carnegie Mellon University. Tech. Re CMU-CS-94—163; 1994.
[39] Harik GR, Lobo FG, Goldberg DE. The compact genetic algorithm. IEEE Transact Evolut Computat 1999;3(4):287—97.
[40] Mühlenbein H, Mahnig T, Ochoa A. Schemata, distributions and graphical models in evolutionary optimization. J Heuristics 1999;5(2):213—47.
[41] De Bonet JS, Isbell CL, Viola P. MIMIC: finding optima by estimating probability densities. In: Mozer MC, Jordan MI, Petsche T, editors. Advances in neural information processing systems, vol. 9. Cambridge: The MIT Press; 1997. p. 424.
[42] G. Harik, Linkage learning via probabilistic modeling in the EcGA. Urbana, IL: University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, IlliGAL Report No. 99010; 1999.
[43] Etxeberria R, Larrañaga P. Global optimization using Bayesian networks. In: Ochoa A, Soto MR, Santana R, editors. Proceedings of the Second Symposium on Artificial Intelligence (CIMAF-99). March 1999. p. 151—73.
[44] Mühlenbein H, Mahnig T. Evolutionary synthesis of Bayesian networks for optimization. In: Advances in Evolutionary Synthesis of Neural Systems. MIT Press; 2001. p. 429—55.
[45] Pelikan M, Goldberg DE, Cantú-Paz E. BOA: The Bayesian optimization algorithm. In: Banzhaf W, Daida J, Eiben AE, Garzon MH, Honavar V, Jakiela M, Smith RE, editors. Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99. San Francisco, CA: Morgan Kaufmann Publishers; 1999. p. 525—32.
[46] Mühlenbein H, Mahnig T. Evolutionary computation and beyond. In: Uesaka Y, Kanerva P, Asoh H, editors. Foundations of real-world intelligence. Stanford, California: CSLI Publications; 2001. p. 123—88.
[47] Wales DJ, Scheraga HA. Global optimization of clusters, crystals, and biomolecules. Science 1999;285(5432):1368—72.
[48] González C, Lozano JA, Larrañaga P. Mathematical modeling of UMDAc algorithm with tournament selection. Behaviour on linear and quadratic functions. Int J Approx Reason 2002;31(4):313—40.
[49] Glick M, Rayan A, Goldblum A. A stochastic algorithm for global optimization for best populations: a test case of side chains in proteins. Proc Natl Acad Sci 2002;99(2):703—8.
[50] Mühlenbein H, Höns R. The estimation of distributions and the minimum relative entropy principle. Evolut Computat 2005;13(1):1—27.