Adding Probabilistic Dependencies to the Search of Protein Side Chain Configurations Using EDAs

Roberto Santana[1], Pedro Larrañaga[2], and Jose A. Lozano[1]

[1] Intelligent Systems Group
[2] Department of Computer Science and Artificial Intelligence
University of the Basque Country
Paseo Manuel de Lardizábal 1, 20018. San Sebastian - Donostia, Spain
[3] Department of Artificial Intelligence, Technical University of Madrid,
28660 Boadilla del Monte, Madrid, Spain
roberto.santana@ehu.es, pedro.larranaga@fi.upm.es, ja.lozano@ehu.es

Abstract. The problem of finding an optimal positioning for the side chain residues of a protein is called the side chain placement or side chain prediction problem. It can be posed as an optimization problem in the discrete domain. In this paper we use an estimation of distribution algorithm to address this optimization problem. Using a set of 50 difficult protein instances, it is shown that the addition of dependencies between the variables in the probabilistic model can improve the quality of the solutions achieved for most of the instances considered. However, we also show that only when information about the known interactions between the residues is considered in the creation of the probabilistic model, the addition of the dependencies contributes to improve the quality of the solutions obtained.

Keywords: estimation of distribution algorithm, protein structure prediction, probabilistic models.

1 Introduction

Estimation of distribution algorithms (EDAs) [12,14,15] are evolutionary algorithms that use probability models instead of genetic operators. Probabilistic modeling allows EDAs to represent relevant features from the search space that can be automatically learned from the data using machine learning methods. The ability of EDAs to solve hard optimization problems and the capacity to extract previously unknown features of the problem domains have contributed to a recent upsurge of their applications in Bioinformatics [2,11,19,21].

In this paper, we address the important issue of the influence that the use of probabilistic dependencies has in the solution of the protein side chain placement problem. In [19], an EDA approach, based on the application of the univariate marginal distribution algorithm (UMDA) [15], has been applied to this protein problem. The UMDA approach uses the simplest probabilistic model where all variables are considered independent.

G. Rudolph et al. (Eds.): PPSN X, LNCS 5199, pp. 1120–1129, 2008.

In EDAs, the capacity of the model to represent complex interactions between the problem components usually influences its accuracy. On the other hand, increasing the model complexity has also an associated computational cost. Therefore, central to the notion of efficient modeling is the achievement of an appropriate balance between the adequate complexity of the model (enough to represent the relevant features of the problem) and a feasible computational cost (the time and the storage requirements must be affordable).

Although UMDA's capacity of representation is highly limited, for a number of difficult protein side chain placement instances where other state-of-the-art algorithms failed to converge, UMDA was able to find better structures than the other algorithms [19].

In this paper, we investigate the question of whether the representation of interactions, by means of probability dependencies between the variables determines a real improvement in the efficiency of EDAs for the protein side chain placement problem. To this end, we apply a tree-based EDA and compare its results with UMDA. In addition, we introduce a proposal to improve the quality of the solutions found and diminish the computational burden of the algorithms by using the protein structure information available.

2 Protein Side Chain Placement Problem

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.

Assuming that the position of the protein backbone is fixed, and considering fixed bond lengths, the location of the protein side chain residues can be completely determined by the sidechain dihedral angles. The problem of finding an optimal positioning for the side chain residues is called side chain placement or side chain prediction [13].

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 [6]. 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 [6]. A rotamer library is a collection of rotamers for each residue type.

The inclusion of these discrete configurations implies an important problem reduction. The search for the protein structure is 'reduced' to the search of a set of rotamers (one for each residue) that optimizes the fitness function. However, the combinatorial problem is NP-hard [17] and, in general, the use of brute force algorithms is unaffordable.

In the protein side chain problem, variable X_i will represent the set of dihedral angles corresponding to the i-th residue. x_i will be interpreted as one of the indexed set of rotamer configurations from the rotamer library. Each configuration encodes one value for each of the dihedral angles of the i-th residue. The number of values of each variable will correspond to the number of rotamer configurations for the corresponding residue i (i.e. $x_i \in \{1, \ldots, k_i\}$, where k_i is the number of feasible rotamer configurations for residue i).

When the backbone is fixed, the energy of a sequence of n amino acids folded into a defined structure can be expressed 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 all pairs of sidechains. For two sets of atoms, the interaction energy is the sum of the pairwise atom interactions. We have adopted the van der Waals energy function as implemented in [23]. This energy function approximates the repulsive portion of Lennard-Jones 12-6 potential. It penalizes steric clashes between atoms. The energy of residues that do not interact is zero for every possible rotamer configuration.

Different optimization approaches to optimal side chain prediction have been proposed. Among the most common approaches used are dead-end elimination (DEE) algorithms [5], the self consistent mean field approach (SCMF) [10], and side chain placement with rotamer library (SCWRL) [6]. Inference-based methods [23] can be also used to find the exact solutions of the side chain prediction problem but they may fail to converge for some difficult instances. In these cases, the application of heuristic approaches is necessary.

3 Estimation of Distribution Algorithms

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

In the EDA approach we follow, each individual represents one possible solution and it is encoded using the vector representation introduced above. The selection step is based on the evaluation of a predefined fitness function. A key 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.

UMDA is the simplest EDA approach to the problem treated in this paper. The probability assigned by the model to each solution is equal to the product of the variables' univariate probabilities. UMDA's estimation step consists of calculating the univariate frequencies of each value for every variable. In the sampling step, new solutions are generated by independently sampling each variable.

In this paper, we propose the application of a model that captures bivariate dependencies between the variables. This probabilistic model is based on a tree where each variable may depend on at most another variable that is called the parent. A probability distribution $p_{Tree}(\mathbf{x})$ that is conformal with a tree is defined as:

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

where $Pa(X_i)$ is the parent of X_i in the tree, and $p(x_i|pa(x_i)) = p(x_i)$ when $Pa(X_i) = \emptyset$, i.e. X_i is the root of the tree. The distribution $p_{Tree}(\mathbf{x})$ itself will be called a tree model when no confusion is possible. Probabilistic trees are represented by acyclic connected graphs.

The construction of the tree structure from data implies the detection of the most important bivariate interactions between the variables. This can be done applying statistical independence tests [16] or methods based on the analysis of the mutual information between variables as in [1]. We follow the second approach using the tree-based EDA (Tree-EDA)[1].

Initially, the bivariate probabilities are calculated for every pair of variables. From these bivariate probabilities, the mutual information between variables is found. To construct the tree structure, an algorithm introduced in [4], that calculates the maximum weight spanning tree from the matrix of mutual information between pairs of variables, is used. Probabilistic logic sampling [8] is used to sample new solutions from the tree. New solutions are generated starting from the root of the tree and sampling each variable conditioned by its parent. More details about the algorithm and its computational cost could be found in [22]. Tree-EDA has been successfully applied to other classes of protein problems [18,21].

3.1 Using Problem Information to Increase Efficiency of EDAs

The energy function used to evaluate the protein side chain problem assigns a zero contribution to pairs of residues that do not interact. We could expect that the probabilistic dependencies between the corresponding pairs of variables will be weaker. Therefore, one variant of the tree learning algorithm constrains the calculation of bivariate probabilities and mutual information to those pairs of variables corresponding to residues that are interacting in the backbone (those that actually interact in the crystal structure). We call this algorithm Tree-EDA^r. During the learning phase of Tree-EDA^r, the computation of the mutual information is done only for the previously fixed subset of variables pairs. The tree structure only includes pairs of variables that belong to this subset.

This approach, which has been previously tested for other class of protein problems [18], helps to reduce the number of spurious correlations that contribute to deteriorate the accuracy of the models and negatively influence the efficiency of the search.

The computational complexity of EDAs is mainly dependent on the complexity of the learning algorithm, but also depends on the population size and number of generations needed for convergence, which are problem-dependent. While the computational complexity of UMDA is linear in the number of variables, for Tree-EDA it is quadratic [22]. Nevertheless, the use of problem structure, as done by Tree-EDA^r, reduces the time spent to learn the probabilistic model.

[1] C++ (EDA program) and Matlab (MATEDA) implementations of UMDA, Tree-EDA, and other EDAs are respectively available from endika@si.ehu.es and http://www.sc.ehu.es/ccwbayes/members/rsantana/software/matlab/

4 Experiments

First, we introduce the protein benchmark and the parameters used by the algorithms. Then, we explain how the experiments were designed. Finally, the results of the experiments are presented.

4.1 Protein Benchmarks

To test our algorithms, we started with a protein dataset of 493 X-ray crystal structures[2] with a resolution better than or equal to $2\mathring{A}$, an R factor below 20%, and a mutual sequence identity lower than 50%. Each protein consisted of 1-4 chains and up to 1000 residues. As a pre-processing step, we determined the instances in which the Goldstein criterion [5] eliminated all configurations but one, and those instances in which the inference-based algorithm for structure prediction (SPRINT) [23] converged.

SPRINT is one of the state-of-the-art algorithms for protein side chain placement. In [23], the energies obtained by SCWRL [3,6] (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. Proteins that were solved using the Goldstein criterion and those for which SPRINT converged were removed from the original database. The number of the remaining instances, which were used for our experiments, was 50. They serve as an appropriate testbed to focus the investigation of EDAs on a representative set of difficult instances were other efficient algorithms have failed.

4.2 Parameters of the Algorithms

To work, EDAs require the definition of several parameters. We have used the same settings for all instances of the problems treated. The quality of the results achieved by the algorithms will depend on these settings. Since, in this paper, we focus on the role played by dependencies, no attempt has been made to tune the parameters to achieve an optimal performance. The parameters of the EDAs have been set as follows. The population size was set at 5000. The maximum number of generations is 500. Truncation selection with parameter $T = 0.15$ has been used. In this selection scheme, the best $T \cdot N$ individuals of the population are selected to construct the probabilistic model. By setting a rather low value of truncation, a strong selection pressure is induced, forcing the algorithm to discard poor solutions as a faster pace.

We apply a replacement strategy called best elitism in which the selected population at generation t is incorporated into the population of generation $t + 1$, keeping the best individuals found so far and avoiding to revaluate their fitness function. The algorithm stops when the maximum number of generations

[2] These instances have been obtained from Chen Yanover's page: http://www.cs.huji.ac.il/∼cheny/proteinsMRF.html

is reached or the selected population has become too homogeneous (no more than 10 different individuals).

EDAs incorporate an additional problem reduction step to decrease the number of variables and their number of values. This step starts from the application of a dead-end elimination step [5], based on the iterative use of the Goldstein elimination criterion, which establishes a sufficient condition 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. This step considerably contributes to reduce the dimension of the search space, but the search space remains huge.

4.3 Design of the Experiments

To compare the results of the algorithms we conducted, in most of cases, 50 experiments for each instance and algorithm. For a number of complex instances, the number of experiments for each algorithm was reduced to 30. The performance of the algorithms was evaluated considering the fitness of the best solution found in each experiment, the best fitness among all the best solutions found, and the number of experiments in which the best fitness was found.

To determine whether differences between the fitness of the solutions found by the algorithms are statistically significant the Kruskal-Wallis test [9] was employed. The test significance level was 0.05. To compare the algorithms according to the best fitness, we determined for each instance, which was the best solution found by each algorithm in all the experiments done.

4.4 Numerical Results

We compared the quality of the solutions obtained by UMDA, Tree-EDA and Tree-EDA^r using the protein benchmark. Tables 1 and 2 shows the 27 instances for which Tree-EDA^r found solutions with energies strictly lower than those found by SPRINT, UMDA and Tree-EDA[3]. Table 1 shows the results corresponding to those instances for which 50 experiments were conducted and Table 2 those for which 30 experiments were done. The best solution found by Tree-EDA^r has higher energy than the best solution found by SPRINT in only 5 of the 50 instances. The table shows the size of each instance after the application of the Goldstein criterion (size), the number of experiments conducted (exp.), the best value of the fitness found in all the experiments (best), the number of times the best solution was found (S), and the average fitness (mean) of the best solutions of each experiment. The standard deviation of the EDAs results changed according to the instance used. For sake of space these values are not reported.

The application of the Kruskal-Wallis test found significant statistical differences between UMDA and Tree-EDA for 26 of the 50 instances. For 15 of these instances, UMDA was better than Tree-EDA. Significant statistical differences between UMDA and Tree-EDA^r were found for 45 of the 50 instances. For 43

[3] Detailed results for all the instances are available as additional documentation from http://www.sc.ehu.es/ccwbayes/EDA/EDAProteinProblems.html

Table 1. Results achieved by UMDA, Tree-EDA and Tree-EDA^r for the selected instances

			UMDA			Tree-EDA			Tree-EDA^r		
pdb id	*size*	*exp.*	*best*	*S*	*mean*	*best*	*S*	*mean*	*best*	*S*	*mean*
pdb1crz	75	50	626.41	1	627.25	626.12	7	627.54	626.12	24	626.56
pdb1ddt	146		754.93	1	760.02	754.30	1	760.88	753.38	26	754.05
pdb1dpe	185		727.37	2	750.51	725.83	1	739.73	725.50	18	729.94
pdb1gsk	208		939.94	1	947.77	934.79	1	945.74	934.01	12	935.62
pdb1h3n	318		1626.09	1	1639.00	1617.70	1	1653.47	1611.76	1	1625.08
pdb1jy1	144		861.92	1	870.34	856.88	4	860.76	856.84	33	856.92
pdb1kmo	241		925.90	1	943.09	890.85	1	924.32	875.20	1	886.86
pdb1kwh	207		972.11	1	988.21	960.73	2	980.09	959.17	1	965.53
pdb1nqe	189		570.29	1	593.49	563.36	1	588.67	563.30	16	566.70

of these instances, Tree-EDA^r outperformed UMDA. The statistical tests confirmed what can be seen from Tables 1 and 2: Tree-EDA^r consistently found better solutions than UMDA. However, the performance of Tree-EDA is not always better than UMDA. Furthermore, a detailed analysis of the experiments (see supplementary material), can reveal that UMDA beats the other two EDAs for the largest problem instances. This might be explained by the fact that the population sizes used by Tree-EDA and Tree-EDA^r are still insufficient to learn an accurate model of the interactions in very large problems. An exhaustive analysis of this question is beyond the scope and space limitations of this paper.

Table 2. Results achieved by UMDA, Tree-EDA and Tree-EDA^r for the selected instances

			UMDA			Tree-EDA			Tree-EDA^r		
pdb id	*size*	*exp.*	*best*	*S*	*mean*	*best*	*S*	*mean*	*best*	*S*	*mean*
pdb1dxr	353	30	1703.73	1	1722.79	1701.61	1	1716.89	1695.16	5	1699.57
pdb1dz4	288		875.77	1	884.79	868.92	1	880.96	867.01	3	872.16
pdb1d2e	281		1839.67	1	1847.65	1829.95	1	1841.83	1823.87	2	1829.18
pdb1e61	454		1936.92	1	1958.89	1942.94	1	1992.96	1911.46	1	1918.09
pdb1e6p	365		1681.67	1	1694.86	1681.93	1	1703.08	1673.47	1	1680.59
pdb1f60	123		537.42	3	540.78	536.69	1	540.68	535.14	6	536.41
pdb1fmj	294		1100.51	1	1121.42	1089.81	1	1105.23	1088.80	8	1092.90
pdb1fn9	239		989.51	3	993.92	988.82	1	1004.05	987.13	1	990.61
pdb1fnn	240		735.75	1	749.53	732.90	1	751.41	732.01	4	735.82
pdb1giq	265		806.53	1	823.58	801.38	1	815.62	800.07	3	800.95
pdb1h3f	206		785.56	1	795.11	784.95	1	794.09	782.98	7	785.68
pdb1h4r	227		825.64	1	830.12	816.96	1	824.09	815.84	8	817.45
pdb1h80	229		1036.90	1	1040.45	1034.96	1	1039.07	1034.77	9	1035.15
pdb1iqc	288		1538.37	1	1546.85	1531.80	1	1536.20	1530.18	5	1532.12
pdb1jmx	285		1518.10	1	1545.51	1510.21	1	1534.77	1504.70	3	1510.91
pdb1j3b	289		1600.16	1	1625.67	1592.75	1	1616.81	1584.68	2	1598.96
pdb1j8f	329		957.08	1	964.50	942.69	3	954.67	942.62	16	943.56
pdb1lqt	268		935.95	1	967.32	926.16	1	941.13	926.16	12	927.22
pdb1lsh	350		1125.04	1	1135.00	1120.77	1	1132.27	1117.76	1	1119.78
pdb1tki	164		858.67	1	867.24	856.62	1	860.97	855.56	5	856.98

Additional experiments were conducted to investigate the structural differences between the best solutions found by UMDA and those obtained using Tree-EDA and Tree-EDA^r. This type of experiments is illustrated using dimer protein pdb1tki. This protein has two symmetric chains and the number of residues before the application of the Goldstein criterion is 576. The energies corresponding

to best side chain structures found by UMDA, Tree-EDA and Tree-EDA^r were respectively 858.67, 856.62 and 855.56.

The structures found by Tree-EDA and Tree-EDA^r are respectively shown in Figure 1 (left) and Figure 1 (right). The protein structures found by UMDA and Tree-EDA are different in only 7 residues. These residues are identified by its corresponding sequence number in Figure 1 (left). Their rotamer configurations are accountable for the improvement in the energy. In the figure, those residues that interact are linked. On the other hand, the structures found by UMDA and Tree-EDA^r are different in only 9 of the residues which are identified in Figure 1 (right). Also in this case, every residue is linked at least to another one.

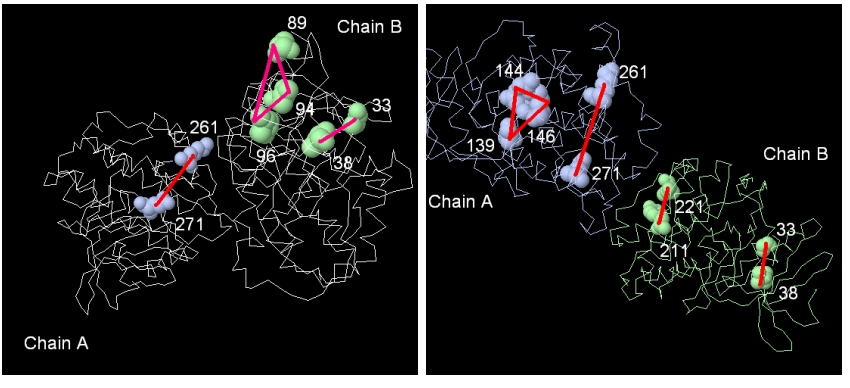

Fig. 1. Backbone and best side chains configurations found by EDAs for a subset of protein pdb1tki's residues. Learned by Tree-EDA (left). Learned by Tree-EDA^r (right).

In general, changes between the rotamer configurations of the best solutions found by UMDA and those found using probabilistic dependencies occur for clusters of interacting residues. UMDA is supposed to find the optimal configurations for pair of residues with weak interactions but it is not able to find the optimal configurations for clusters of variables that strongly interact. For several of these cases, the use of probabilistic dependencies improve the results. In this problem, the contrastive analysis of the best solutions found with and without the use of dependencies helps to identify clusters of variables with strong patterns of interactions which in turn correspond to residues that interact in the protein structure.

5 Conclusions and Future Work

The results presented in this paper show how results achieved in the protein side chain placement problem can be noticeably improved by the use of probabilistic models able to represent interactions between the variables. The results obtained by Tree-EDA^r could be further improved by the application of local search methods as those used in [20] to refine the solutions found by UMDA. Other EDAs

could be applied, however the application of EDAs that use more complex probabilistic models, such as Bayesian networks or Markov networks, does not seem to be a good option due to the high cardinality of the problem variables.

Finally, we point to the fact that contrastive analysis of solutions found using different classes of models could be added to the set of available techniques [7,18,21] used to extract problem information from the probabilistic models learned during the search.

Acknowledgements

The authors thank Chen Yanover for providing the set of instances used in the experiments. This work has been partially supported by the Etortek, Saiotek and Research Groups 2007-2012 (IT-242-07) programs (Basque Government), TIN2005-03824 and Consolider Ingenio 2010 - CSD2007-00018 projects (Spanish Ministry of Education and Science) and COMBIOMED network in computational biomedicine (Carlos III Health Institute).

References

1. Baluja, S., Davies, S.: Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space. In: Proceedings of the 14th International Conference on Machine Learning, pp. 30–38. Morgan Kaufmann, San Francisco (1997)
2. Belda, I., Madurga, S., Llorá, X., Martinell, M., Tarragó, T., Piqueras, M., Nicolás, E., Giralt, E.: ENPDA: An evolutionary structure-based de novo peptide design algorithm. Journal of Computer-Aided Molecular Design 19(8), 585–601 (2005)
3. Canutescu, A.A., Shelenkov, A.A., Dunbrack, R.L.: A graph-theory algorithm for rapid protein side-chain prediction. Protein Science 12, 2001–2014 (2003)
4. Chow, C.K., Liu, C.N.: Approximating discrete probability distributions with dependence trees. IEEE Transactions on Information Theory 14(3), 462–467 (1968)
5. De Maeyer, M., Desmet, J., Lasters, I.: The dead-end elimination theorem: Mathematical aspects, implementation, optimization, evaluation, and performance. Methods in Molecular Biology 143, 265–304 (2000)
6. Dunbrack, R.L.: Rotamer libraries in the 21st century. Current Opinion in Structural Biology 12, 431–440 (2002)
7. Echegoyen, C., Lozano, J.A., Santana, R., Larrañaga, P.: Exact Bayesian network learning in estimation of distribution algorithms. In: Proceedings of the 2007 Congress on Evolutionary Computation CEC 2007, pp. 1051–1058. IEEE Press, Los Alamitos (2007)
8. Henrion, M.: Propagating uncertainty in Bayesian networks by probabilistic logic sampling. In: Lemmer, J.F., Kanal, L.N. (eds.) Proceedings of the Second Annual Conference on Uncertainty in Artificial Intelligence, pp. 149–164. Elsevier, Amsterdam (1988)
9. Hsu, J.C.: Multiple Comparisons: Theory and Methods. Chapman and Hall, Boca Raton (1996)
10. Koehl, P., Delarue, M.: Building protein lattice models using self consistent mean field theory. Journal of Chemical Physics 108, 9540–9549 (1998)

11. Larrañaga, P., Calvo, B., Santana, R., Bielza, C., Galdiano, J., Inza, I., Lozano, J.A., Armañanzas, R., Santafé, G., Pérez, A., Robles, V.: Machine learning in bioinformatics. Briefings in Bioinformatics 7, 86–112 (2006)
12. Larrañaga, P., Lozano, J.A. (eds.): Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation. Kluwer Academic Publishers, Boston (2002)
13. Lee, C., Subbiah, S.: Prediction of protein side-chain conformation by packing optimization. Journal of Molecular Biology 217, 373–388 (1991)
14. Lozano, J.A., Larrañaga, P., Inza, I., Bengoetxea, E. (eds.): Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms. Springer, Heidelberg (2006)
15. Mühlenbein, H., Paaß, G.: From recombination of genes to the estimation of distributions I. Binary parameters. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 178–187. Springer, Heidelberg (1996)
16. Pelikan, M., Mühlenbein, H.: The bivariate marginal distribution algorithm. In: Roy, R., Furuhashi, T., Chawdhry, P. (eds.) Advances in Soft Computing - Engineering Design and Manufacturing, London, pp. 521–535. Springer, Heidelberg (1999)
17. Pierce, N.A., Winfree, E.: Protein design is NP-hard. Protein Engineering 15(10), 779–782 (2002)
18. Santana, R., Larrañaga, P., Lozano, J.A.: The role of a priori information in the minimization of contact potentials by means of estimation of distribution algorithms. In: Marchiori, E., Moore, J.H., Rajapakse, J.C. (eds.) EvoBIO 2007. LNCS, vol. 4447, pp. 247–257. Springer, Heidelberg (2007)
19. Santana, R., Larrañaga, P., Lozano, J.A.: Side chain placement using estimation of distribution algorithms. Artificial Intelligence in Medicine 39(1), 49–63 (2007)
20. Santana, R., Larrañaga, P., Lozano, J.A.: Combining variable neighborhood search and estimation of distribution algorithms in the protein side chain placement problem. Journal of Heuristics (to appear, 2008)
21. Santana, R., Larrañaga, P., Lozano, J.A.: Protein folding in simplified models with estimation of distribution algorithms. IEEE Transactions on Evolutionary Computation (to appear, 2008)
22. Santana, R., Ochoa, A., Soto, M.R.: The mixture of trees factorized distribution algorithm. In: Spector, L., Goodman, E., Wu, A., Langdon, W., Voigt, H., Gen, M., Sen, S., Dorigo, M., Pezeshk, S., Garzon, M., Burke, E. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference GECCO 2001, pp. 543–550. Morgan Kaufmann Publishers, San Francisco (2001)
23. Yanover, C., Weiss, Y.: Approximate inference and protein-folding. In: Becker, S., Thrun, S., Obermayer, K. (eds.) Advances in Neural Information Processing Systems 15, pp. 1457–1464. MIT Press, Cambridge (2003)