Advances in Probabilistic Graphical Models for Optimization and Learning. Applications in Protein Modeling

Dissertation

Roberto Santana
Department of Computer Science and Artificial Intelligence of the
University of the Basque Country
Co-Supervised by : J. A. Lozano and P. Larrañaga

Donostia - San Sebastián, March 2006

Acknowledgments

This dissertation would never have been possible if it were not for the support of my thesis supervisors José A. Lozano and Pedro Larrañaga. Thank you very much for being my mentors, for your encouragement and patience.

I am grateful to Iñaki Inza and my colleagues in the Intelligent System Group: Aritz Pérez, Borja Calvo, Dinora Morales, Guzmán Santafé, José Luis Flores, Josu Galdeano, Ramón Sagarna, Rosa Blanco and Rubén Armañanzas for the excellent working atmosphere, occasional advice and continous support. Also to those that spent some time at the lab and contributed with their comments to enrich the work presented in this dissertion: Frederick Vincent, Robin Höns, Siddartha Shakya and Yvan Saeys. I am particularly grateful to Robin Höns who hosted me during my stay in Bonn. We enjoyed long sessions of friendly dicussions and talks on EDAs. The dissertation benefited as well from the insights and comments of Marta R. Soto. Thanks too to Alex Mendiburu and Rosa Blanco for their collaboration to finish the thesis.

I am grateful to the University of the Basque Country for the financial support under a pre-doctoral grant.

I would like to take this opportunity to mention some people that have significantly constributed at different times of my research: Eunice Ponce de León, Marta R. Soto, Alberto Ochoa, Heinz Mühlenbein, Livang Lozada, and Thilo Mahnig.

Finally, I would like to thank my family and Cuban friends. They have been present all this time. This thesis is also for them.

Contents

List of Figures

List of Tables

List of algorithms

List of Abbreviations

ACO: Ant Colony Optimization
ADF: Additive Decomposed Function
AIC: Aikaike Information Criterion
BDe: Bayesian-Dirichlet Equivalence Metric
BIC: Bayesian Information Criterion
BEDA: Boltzmann Estimation of Distribution Algorithm
BMDA: Bivariate Marginal Distribution Algorithm
BMMF: Best Max-Marginal Fit algorithm
BN: Bayesian Network
BP: Belief Propagation
BOA: Bayesian Optimization Algorithm
cGA: compact Genetic Algorithm
CVM: Cluster Variation Method
DEE: Dead-end elimination
EDA: Estimation of Distribution Algorithm
FDA: Factorized Distribution Algorithm
GA: Genetic Algorithm
GBP: Generalized Belief Propagation
GS: Gibbs Sampling
KLD: Kullback Leibler Divergence
MC: Monte Carlo
MCMC: Markov Chain Monte Carlo
MDL: Minimum Description Length
MN-EDA: Markov Network Estimation of Distribution Algorithm
MT-EDA: Mixture of Trees Factorized Distribution Algorithm
PBIL: Population Based Incremental Learning
UMDA: Univariate Marginal Distribution Algorithm
VNS: Variable Neighborhood Search

Part I

Probabilistic graphical models

1 Introduction

1.1 Preamble

Probabilistic graphical models (43; 127; 234) are an increasingly used knowledge representation tool. Theoretical developments and reports of successful applications of these models in different domains have been extensively reflected in the literature.

The use of probabilistic graphical models in optimization has been enriched with the introduction of estimation of distribution algorithms (125; 161) and optimization algorithms based on inference methods (36; 240). This new area of application poses new challenges and requires to widen the scope for theoretical studies on graphical models. Furthermore, the algorithms conceived for optimization based on graphical models may be directly applied to practical problems from different fields. One of the current application arenas is computational biology.

This dissertation has two main objectives. The first one is to introduce a number of results from the investigation of probabilistic graphical models that use region based decompositions, and to explain how these results may be applied to optimization. The second objective is to show the way in which protein problems can be efficiently solved by using methods based on graphical models which serve not only as problem solvers but also as modeling tools able to provide useful information about the problem domain. Moreover, the thesis intends to illustrate how challenging problems from computational biology can be addressed by the application of optimization algorithms based on simple and more advanced probabilistic models.

1.2 Overview

The dissertation is divided into four parts and comprises eleven chapters.

The first part defines the thesis objectives. It focuses on the analysis of undirected graphical models. The ways in which inference and sampling methods based on undirected graphical models can be applied to optimization are discussed. It comprises this introductory chapter as well as two other chapters.

In this chapter, we have defined the objectives and present an overview of the thesis. Chapter 2 introduces probabilistic graphical models, focusing on those issues that will

be treated in detail later on in the thesis. We review some of the main concepts related to undirected and directed graphical models, approaches to learn graphical models from data, and inference algorithms.

Chapter 3 focuses the analysis on the use of probabilistic graphical models in optimization. The optimization approaches that use probabilistic graphical models relevant to our research are classified.

The second part of the thesis introduces a number of properties of the class of Kikuchi approximation that use clique-based decompositions. An algorithm that learns this approximation from data is introduced and evaluated in different types of approximation problem. Region-based decompositions in optimization methods that use inference techniques are analyzed. The second part comprises three chapters.

In Chapter 4, the Markov properties of the Kikuchi approximation are proved. It is shown that it is possible to decompose the Kikuchi approximation into the product of local Kikuchi approximations defined on a decomposition of the graph. Partial Kikuchi approximations are introduced. Additionally, the chapter clarifies the place of clique-based decompositions in relation to other techniques inspired by methods from statistical physics, and discusses the application of the results introduced in the paper to the conception of Kikuchi approximation learning algorithms.

Chapter 5 presents a score+search algorithm that learns Kikuchi approximations from datasets. The paper proves that the Markov and decomposability properties satisfied by the Kikuchi approximation permit the conception of learning algorithms that can pursue the search of the model by making only local updates on the current approximation. The chapter presents results on the use of the algorithm that learn Kikuchi approximations to approximate probability distributions by means of Bayesian networks and Boltzmann distributions.

Chapter 6 focuses on the use of region based decompositions, propagation and most probable configuration methods in optimization. The chapter presents a generalized estimation of distribution algorithm that illustrates the application of the results on Kikuchi approximations detailed in previous chapters. Additionally, the chapter puts the algorithms used in the third part of the thesis into context.

The third part of the thesis addresses the application of optimization algorithms based on graphical models to problems from computational biology. It starts by reviewing a number of computational protein problems. Different proposals that allow the efficient solution of some of these problems are introduced. A link to the results achieved in the first part of the thesis is presented by showing how the probabilistic models and techniques introduced can be added to obtain solutions good enough for these problems. The third part of the thesis has four chapters.

Chapter 7 reviews part of the biological basis of proteins. The chapter presents a number of problems in protein research and describes how they have been treated by some

approaches from the field of machine learning.

Chapter 8 introduces the application of different variants of estimation of distribution algorithms to the solution of the protein structure problem in simplified models, and proposes their use as a simulation tool for the analysis of the protein folding process. New ideas for the application of EDAs to the bidimensional and tridimensional (2-d and 3-d) simplified protein folding problems are developed.

Chapter 9 presents an algorithm for the solution of the side chain placement problem. The algorithm combines the application of the Goldstein elimination criterion with the univariate marginal distribution algorithm 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. The results obtained show that the algorithm can achieve better structures than those obtained with other state-of-the-art methods.

Chapter 10 presents an application of optimization methods based on graphical models to a problem from the protein design field. An estimation of distribution algorithm that includes a method to generate the most probable configuration of a probability distribution is introduced. The method is applied to find protein sequences with low energy.

Part four consists of only one chapter that presents the conclusions of the thesis.

2 Probabilistic graphical models

2.1 Introduction

Probabilistic graphical models (43; 127; 234) have become common knowledge representation tools capable of efficiently representing and handling independence relationships. They are one of the most recurred machine learning paradigms to specify interactions in complex systems in terms of probabilistic dependencies. In this chapter, we introduce the main concepts related to graphical models that will be used throughout the dissertation.

In the most general case, there are four distinct components of a graphical model (25): the semantics, the structure, the implementation, and the parameters.

There are a variety of semantics, including directed and undirected models, chain graphs and other frameworks. The model structure is related to dependence/independency relationships displayed in a graph: the absence of some links means the existence of certain conditional and/or marginal independence relationships between variables, and the presence of links may represent the existence of direct dependency relationships.

Fixing the structure, there are a number of ways to implement the dependencies between random variables, such as conditional probability tables or ADtrees (8). The quantitative component of the model is a collection of numerical parameters, usually conditional probabilities, which give an idea of the strength of the dependencies.

2.2 Notation

Let X be a random variable. A value of X is denoted x. $\mathbf{X} = (X_1, \ldots, X_n)$ will denote a vector of random variables. We will use $\mathbf{x} = (x_1, \ldots, x_n)$ to denote an assignment to the variables. S will denote a set of indices in $N = \{1, \ldots, n\}$, and $\mathbf{X}_S$ (respectively, $\mathbf{x}_S$) a subset of the variables of $\mathbf{X}$ (respectively, a subset of values of $\mathbf{x}$) determined by the indices in S. The joint generalized probability distribution of $\mathbf{x}$ is represented as $\rho(\mathbf{X} = \mathbf{x})$ or $\rho(\mathbf{x})$. $\rho(\mathbf{x}_S)$ will denote the marginal generalized probability distribution for $\mathbf{X}_S$. We use $\rho(X_i = x_i \mid X_j = x_j)$ or, in a simplified form, $\rho(x_i \mid x_j)$, to denote the conditional generalized probability distribution of X_i given $X_j = x_j$.

If the set of random variables $\mathbf{X}$ is discrete, $\rho(\mathbf{X} = \mathbf{x}) = p(\mathbf{X} = \mathbf{x})$ or $p(\mathbf{x})$. $p(\mathbf{X} = \mathbf{x})$ is known as the joint probability mass function for $\mathbf{X}$. Similarly, $p(X_i = x_i)$ is the marginal

mass probability function of X_i and $p(x_i \mid x_j)$ is the conditional mass probability of X_i given $X_j = x_j$.

If the set of random variables $\mathbf{X}$ is continuous, $\rho(\mathbf{X} = \mathbf{x}) = f(\mathbf{x})_{\mathbf{X}}$ or $f(\mathbf{x})$. $f(\mathbf{X} = \mathbf{x})$ is known as the joint density function of $\mathbf{X}$. Similarly, $f(x_i)_{X_i}$ is the marginal density function of X_i and $f(x_i \mid x_j)$ is the conditional density function of X_i given $X_j = x_j$.

2.3 Graphical models based on undirected graphs

An undirected graph $G = (V, E)$ is defined by a non-empty set of vertices or nodes V, and a set of edges E. An edge between nodes V_i and V_j will be represented by $i \sim j$.

Definition 2.1. *Given an undirected graph $G = (V, E)$, a clique in G is a subset of V for which there exists an edge between every pair of vertices. We reserve the letter C to refer to a clique. C is maximal when it is not contained in any other clique. C is a maximum clique of the graph if it is a clique in $\mathcal{C}$ with the highest number of vertices. The collection of all maximal cliques in G is denoted as $\mathcal{C}$.*

Definition 2.2. *Given an undirected graph $G = (V, E)$, a junction graph constructed from a set of maximal cliques is a graph where each node corresponds to a maximal clique, and there exists an edge between two nodes if their corresponding cliques overlap.*

A cycle in a graph is a vertex sequence $V_1, \ldots, V_n$ where $V_1 = V_n$ but all other pairs are distinct, and $\{V_i, V_{i+1}\} \in E$. A cycle is chordless if all pairs of vertices that are not adjacent in the cycle are not neighbors. An undirected graph is said to be chordal if every cycle of length four or more has a chord. A fundamental property of chordal graphs is that their maximal cliques can be joined to form a tree, called the junction tree, such that any two cliques containing a node α are either adjacent in the junction tree, or connected by a chain made entirely of cliques that contain α. This property is called the running intersection property (127). A junction tree is an acyclic junction graph.

Independence graphs display the probability dependencies that exist between the variables of a given probability distribution. Two vertices are joined in the independence graph if the corresponding variables are not conditionally independent given the rest of the variables. Throughout this thesis we will mainly consider independence graphs defined on undirected graphs. Each variable X_i has an associated vertex V_i in the corresponding graph. To emphasize this relationship, vertices may be named after their variables: $V = \{X_1, \ldots, X_n\}$. When we refer to a variable, or a vertex or node, it will be clear from the context.

Markov random fields

Given an undirected graph $G = (V, E)$ we have the following definitions:

Definition 2.3. *The neighborhood $N(X_i)$ of a vertex $X_i \in \mathbf{X}$ is defined as $N(X_i) = \{X_j : X_j \sim X_i \in E\}$. The set of edges uniquely determines a neighborhood system on the associated graph G.*

Definition 2.4. *The boundary of a set of vertices, $\mathbf{X}_S \subseteq \mathbf{X}$, is the set of vertices in $\mathbf{X} \setminus \mathbf{X}_S$ that neighbors at least one vertex in $\mathbf{X}_S$. The boundary of $\mathbf{X}_S$ is denoted as $bd(\mathbf{X}_S)$.*

Definition 2.5. *The closure of a set of vertices, $\mathbf{X}_S \subseteq \mathbf{X}$, is the set of vertices $cl(\mathbf{X}_S) = \mathbf{X}_S \cup bd(\mathbf{X}_S)$.*

Definition 2.6. *A probability $p(\mathbf{x})$ is called a Markov random field with respect to the neighbor system on a graph G if, $\forall \mathbf{x} \in \mathbf{X}$, $\forall i \in \{1, \ldots, n\}$, $p(x_i|\mathbf{x} \setminus x_i) = p(x_i|bd(x_i))$.*

Definition 2.7. *A probability $p(\mathbf{x})$ on a graph G is called a Gibbs field with respect to the neighborhood system on the associated graph G when it can be represented as follows:*

$$p(\mathbf{x}) = \frac{1}{Z} e^{-H(\mathbf{x})} \tag{2.1}$$

where $H(\mathbf{x}) = \sum_{C \in \mathcal{C}} \Phi_C(\mathbf{x})$ is called the energy function, being $\Phi = \{\Phi_C \in C\}$ the set of clique potentials, one for each of the maximal cliques in G. The value of $\Phi_C(\mathbf{x})$ depends on its local configuration on the clique C. The normalizing constant Z is the corresponding partition function, $Z = \sum_{\mathbf{x}} e^{-H(\mathbf{x})}$.

Probabilistic models based on undirected graphs will be analyzed in detail in chapter 4.

2.4 Graphical models based on directed graphs

A probabilistic graphical model based on a directed graph is a representation of the joint generalized probability density function $\rho(\mathbf{x})$. The representation consists of two components: a structure and a set of local generalized probability distributions. The structure S for $\mathbf{X}$ is a directed acyclic graph (DAG) that represents a set of conditional independence assertions about the variables on $\mathbf{X}$. The structure S for $\mathbf{X}$ represents the assertions that X_i and $\{X_1, \ldots, X_n\} \setminus \mathbf{Pa}_i^S$ are independent given $\mathbf{Pa}_i^S$, $i = 2, \ldots, n$. The set of variables $\mathbf{Pa}_i^S$ are called the parents of X_i. Thus, the factorization is as follows:

$$\rho(x_1, \ldots, x_n) \;=\; \prod_{i=1}^{n} \rho(x_i|\mathbf{pa}_i^S) \tag{2.2}$$

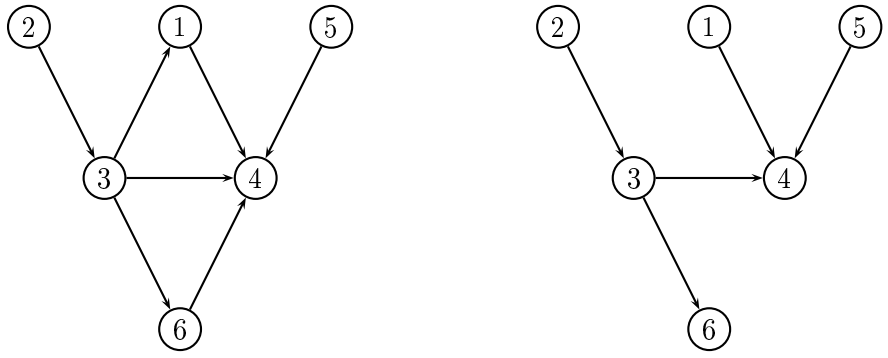


Figure 2.1: Structures of: a) General Bayesian network; b) Polytree.

The local generalized probability distributions associated with the probabilistic graphical model are precisely those in the previous equation. In this representation, it is assumed that, for each variable, the local generalized probability distribution depends on a finite set of parameters $\theta_S = (\theta_1, \ldots, \theta_n)$. Hence, Equation (2.2) could be rewritten as:

$$\rho(\mathbf{x} \mid \theta_S) = \prod_{i=1}^{n} \rho(x_i \mid \mathbf{pa}_i^S, \theta_i) \tag{2.3}$$

2.4.1 Bayesian networks

Bayesian networks are graphical models based on directed acyclic graphs and discrete variables. They have been used for probabilistic inference in domains such as expert systems (61; 126) classification problems (27; 76), and optimization (72; 179).

In a Bayesian network, where variable X_i has r_i possible values, $(x_i^1, \ldots, x_i^{r_i})$, the local distribution $p(x_i \mid \mathbf{pa}_i^{j,S}, \theta_i)$ is an unrestricted discrete distribution:

$$p(x_i^k \mid \mathbf{pa}_i^{j,S}, \theta_i) = \theta_{x_i^k \mid \mathbf{pa}_i^j} \equiv \theta_{ijk} \tag{2.4}$$

where $\mathbf{pa}_i^{1,S}, \ldots, \mathbf{pa}_i^{q_i,S}$ denotes the values of $\mathbf{Pa}_i^S$, the set of parents of X_i in the structure S. q_i is the number of possible different instances of the parent variables of X_i, hence $q_i = \prod_{X_g \in \mathbf{Pa}_i^S} r_g$.

The local parameters are given by $\theta_i = ((\theta_{ijk})_{k=1}^{r_i})_{j=1}^{q_i}$. Parameter θ_{ijk} is the conditional probability of variable X_i being in its k-th state given that the set of parents is in its j-th configuration.

One of the aspects that influences the complexity of the Bayesian network is the connectivity of the model structure. A number of particular Bayesian networks can be defined according to the structure.

Definition 2.8. *A polytree is a Bayesian network that does not allow cycles in the associated undirected graph.*

Definition 2.9. *A tree is a subclass of the polytree class where each vertex in the graph can have not more than one parent in the network.*

Another particular case of a graphical model is the total independent model where none of the nodes have parents (all the variables are independent). Figure 2.1 shows the graphical model structures corresponding to a general Bayesian network and a polytree.

2.5 Algorithms for learning graphical models from data

Graphical models can be learned from data. In Chapter 5, we will analyze the problem of learning a probabilistic model based on an undirected graph from data. Here we briefly review the alternatives that exist when learning a Bayesian network.

There are two main approaches to learning Bayesian networks from data: learning based on detecting conditional independencies by means of independence tests, and score+search algorithms. The problem of learning a general Bayesian network from data is NP-hard (48).

2.5.1 Learning based on detecting conditional independencies

The input of these algorithms are some dependency relationships among subsets of variables of the problem, and their output, the Bayesian network structure. The dependency relationships can be obtained by different means. Independence relationships can be learned from the data by means of independence tests. These type of algorithms differ in the cost of the statistical tests, and in the reliability of the results (124). Two of the methods used for the structure learning of Bayesian networks, and which are based on detecting conditional independencies, are the PC algorithm (208; 209), and the algorithms used to learn polytrees presented in (2).

2.5.2 Learning based on score+search methods

The problem of finding an accurate Bayesian network can be posed as the optimization of a score function over the set of all possible graph structures. The score can be understood

as a measure of the accuracy of the structure to represent the independency relationships of the data. The score usually includes a term to penalize the complexity of the network. Likelihood scores are usually employed together with complexity penalization functions like the Bayesian information criterion (BIC) (198) or the Akaike information criterion (AIC) (6). Bayesian and information theory based scores can also be employed (40). A certain Bayesian scores is the Bayesian-Dirichlet equivalence (BDe), whose metric combines prior knowledge of the problem and statistical data from a given dataset.

Learning a Bayesian network from data as a score+search approach is proven to be an NP-hard problem when the BDe metric is the objective function (48). It is assumed that, when other scores are used, the complexity of the problem is not reduced (27). Therefore, heuristic strategies, among which the most popular are greedy based search heuristics (51), have to be followed to obtain good solutions. A detailed account of the methods used to learn Bayesian networks can be found in (124).

2.6 Inference

Used for inference, a graphical model can provide specific information about the joint probability distribution (e.g. marginal probabilities). Inference can be done in the presence of information that was not initially stored in the graphical model (evidence). When knowledge is represented in terms of the probability dependencies contained in a graphical model, it can be used to infer the probability of an event given that some (possibly incomplete) information is available.

Given a graphical model that encodes the dependencies of the variables of our domain, we would like to find the probabilities of the assignments for a subset of the variables. This can be achieved by marginalization. Marginal probabilities are defined in terms of sums over all the possible states for all the other nodes in the system. For example, if the marginal probability of the last node $p(x_n)$ needs to be computed, this can be achieved using Equation 2.5.

$$p(x_n) = \sum_{x_1} \sum_{x_2} \cdots \sum_{x_{n-1}} p(x_1, x_2, x_3, \ldots, x_n) \tag{2.5}$$

Approximate marginal probabilities are usually called "beliefs". For graphical models with a small number of variables and values, marginalization can be easily done directly, but the number of terms in the sums will grow exponentially with the number of nodes in the model. In these cases, different algorithms can be used to calculate the marginals. We analyze two cases: Algorithms based on sampling (91), and belief propagation (BP) (24; 177; 245).

2.6.1 Inference algorithms based on stochastic sampling

Stochastic sampling algorithms (also called stochastic simulation or Monte Carlo algorithms) (23; 140) are a family of algorithms used to approximate a distribution $p(\mathbf{x})$ and to estimate expectations derived from $p(\mathbf{x})$. They include methods that only generate independent samples from the desired distribution (e.g. rejection sampling) and other methods where this is not necessarily the case, like Markov chain Monte Carlo methods (MCMC).

MCMC methods comprise different algorithms able to devise an ergodic Markov chain such that the steady-state distribution of the chain is $p(\mathbf{x})$. To approximate the probability, the chain can be started at any point $\mathbf{x}$, and, after a sufficiently long time (e.g. k steps), take each state of the chain as a sample of the desired probability. The MCMC literature refers to the initial time the Markov chain is run until it is assumed to be as close enough to stationarity as the burn-in period.

Classical MCMC algorithms can determine only an approximation of $p(\mathbf{x})$ that depends on the burn-in period. In general, it is a difficult to estimate the burn-in period a priori in such a wayt that $d(p(\mathbf{x}) - \sigma(\mathbf{x})) < \epsilon$, where $\sigma(\mathbf{x})$ is the approximation of $p(\mathbf{x})$, and ϵ is the acceptance error.

Inference algorithms based on stochastic sampling generate a set of points simulating from the graphical model. From this set, they estimate the beliefs of the desired variables. After a sufficient amount of points have been generated, they are taken as samples from the original distribution, and used to find the statistics desired. Probabilistic logic sampling (PLS) (91) was the first proposed sampling algorithm for Bayesian networks. It starts from an order imposed to the variables according to the Bayesian network structure. Each variable is sampled given the values assigned to its ascendants in the order.

2.6.2 Inference algorithms based on propagation methods

Traditional propagation methods are commonly used for probabilistic reasoning based on graphical models. They proceed by sending flows of messages through the nodes of the graphical model. Belief propagation distributes information across the whole structure.

Pearl (177) has shown that belief propagation gives the exact marginal probabilities for all the nodes in any singly-connected graphical models. Usually, belief propagation is defined on junction trees that can be obtained directly from undirected graphs. For Bayesian networks, they can be obtained by means of a process called compilation that includes the moralization and triangulation of the graph.

As the belief propagation algorithm does not make reference to the topology of the graph that it is running on, it can be used in graphs with loops too. However, in these cases, the messages may circulate indefinitely around the loops and the process may not converge

to a stable equilibrium (177). There are examples of successful applications of BP in graphs with loops (245).

BP can also be used to calculate the most probable configuration of the graphical model (54). This process is also known as *belief revision* (177), or *the most probable explanation problem* (213).

3 Probabilistic graphical models in optimization

3.1 Introduction

In this chapter, we review a number of approaches for the application of probabilistic graphical models in optimization. The chapter lays the foundations for the proposals presented in Chapters 8, 9 and 10.

Given a function $f : \Omega_{\mathbf{X}} \rightarrow \mathcal{R}$, the optimization problem is to find $\mathbf{x}^{max}$ with:

$$f(\mathbf{x}^{max}) = max_{\mathbf{x}} f(\mathbf{x}) = f_{max} \tag{3.1}$$

This maximum is not necessarily unique. The function will be called the *objective function* or *fitness function*. We will constrain the analysis to functions defined on discrete variables.

Regarding the way the search space is sampled by general heuristic methods, there are two main types of approaches. *Single point search methods* inspect one point at a time whereas *population based methods* use a set of points to conduct the search. There are also hybrid approaches that combine techniques of both types.

Among the most used single point search heuristics are tabu search (79), simulated annealing (109) and variable neighborhood search (85). Population based algorithms include ant colony optimization (ACO) (67), evolution algorithms like genetic programming (115), evolution strategies (199), genetic algorithms (GAs) (80; 94), and immune algorithms (73).

Wolpert and Macready (235) have proven that all problem solvers have the same average behavior if all the possible search spaces are considered. In this sense, none of them is better than random search. Other authors (68) state that more restricted scenarios, where complexity or difficulty is limited in some sense, fit better to real life optimization than the scenario described in (235). For these restricted scenarios, it is possible to demonstrate the convenience of using one class of optimization techniques over the others.

The use of inference in optimization begins assuming that, by using probabilistic models, some useful information of the search space can be learned from a set of solutions already

inspected or from the problem structure. This information can be used to conduct a more effective search.

Our analysis will focus on the class of optimization methods that use probabilistic graphical models to organize the search. In this class, we constrain the analysis to algorithms of two types:

1. Estimation of distribution algorithms.
2. Optimization methods based on inference.

3.2 Estimation of distribution algorithms

Estimation of distribution algorithms (EDAs) (30; 125; 161; 178) are evolutionary algorithms that work with a set (or population) of points. Initially, a random sample of points is generated. These points are evaluated using the objective function, and a subset of points is selected based on this evaluation. Hence, points with better function values have a higher chance to be selected. Then a probabilistic model of the selected solutions is built, and a new set of points is sampled from the model. The process is iterated until the optimum has been found or another termination criterion is fulfilled.

Algorithm 3.1: Estimation of distribution algorithm

```
1  Set t ⇐ 0. Generate M points randomly.
2  do {
3      Evaluate the points using the fitness function.
4      Select a set S of N ≤ M points according to a selection method.
5      Calculate a probabilistic model of S.
6      Generate M new points sampling from the distribution represented
       in the model.
7      t ⇐ t + 1
8  } until Termination criteria are met.
```

The general scheme of the EDA approach is shown in Algorithm 3.1. There are a number of selection methods that can be used. In the literature, truncation, Boltzmann, and tournament selection are commonly used with EDAs.

One essential assumption of these algorithms is that it is possible to build a probabilistic model of the search space that can be used to guide the search for the optimum. A key characteristic and crucial step of EDAs is the construction of this probabilistic model. If there is available information about the function (e.g. variable dependencies), this can be exploited by including parametrical and/or structural prior information in the model.

Otherwise, the model is learned exclusively using the selected population. Several probabilistic models with different expressive power and complexity can be applied. 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 the classification according to the complexity of the models used to capture the interdependencies between the variables (122). Regarding the way learning is done in the probability model, EDAs can be divided into two classes. One class groups the algorithms that do a parametric learning of the probabilities, and the other one comprises those algorithms where structural learning of the model is also done. Parametric and structural learning are also known as model fitting and model selection, respectively. Population based incremental learning (PBIL) (15), compact GA (cGA) (88), the univariate marginal distribution algorithm (UMDA) (161), and the factorized distribution algorithm that uses a fixed model of the interactions in all the generations (FDA) (160) all belong to the first class of algorithms. Likewise, the mutual information maximization for input clustering algorithm (MIMIC) (62), the extended compact GA (EcGA) (87) and EDAs that use Bayesian and Gaussian networks (32; 72; 158; 167; 168; 179; 178) belong to the second class. The success of EDAs in the solution of different practical problems has been documented in the literature (138).

As an example, we present the pseudocodes of UMDA (Algorithm 3.2) and Tree-EDA (Algorithm 3.3), which are used in this thesis.

3.2.1 Univariate marginal distribution algorithm

The univariate model used by UMDA 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:

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i) \tag{3.2}$$

Algorithm 3.2 shows the steps of the UMDA. In Algorithm 3.2, $p_i^s(x_i, t)$ is the marginal probability corresponding to value x_i of variable X_i calculated from the selected population at generation t.

Theoretical results derived from the UMDA (161) expose its relationship with GAs, particularly with GAs that use uniform crossover. Mühlenbein and Mahnig (157) have investigated some of the issues that explain the success of UMDA in the optimization of a wide class of functions. Other theoretical results have been obtained for UMDA (82; 249) and other EDAs based on univariate models (81).

Algorithm 3.2: Pseudocode for UMDA

1 Set $t \Leftarrow 0$. Generate M points randomly.
2 **do** {
3 Select a set S of $N \leq M$ points according to a selection method.
4 Compute the univariate marginal frequencies $p_i^s(x_i, t)$ of S.
5 Generate M new points according to the distribution $p(\mathbf{x}, t+1) = \prod_{i=1}^{n} p_i^s(x_i, t)$.
6 $t \Leftarrow t + 1$
7 } **until** Termination criteria are met.

3.2.2 Tree based estimation of distribution algorithms

The tree based estimation of distribution algorithm (Tree-EDA) (191) uses a factorization that is based on a forest. Although other methods can also be employed (180; 207), the factorization is constructed using the algorithm introduced in (49) that calculates the maximum weight spanning tree from the matrix of mutual information between pair of variables. Additionally, a threshold for the mutual information values is used when calculating the maximum weight spanning tree, to allow for disconnected components in the graphical structure.

Algorithm 3.3: Pseudocode for Tree-EDA

1 Set $t \Leftarrow 0$. Generate M points randomly.
2 **do** {
3 Select a set S of $N \leq M$ points according to a selection method.
4 Compute the univariate and bivariate marginal frequencies $p_i^s(x_i, t)$ and $p_{i,j}^s(x_i, x_j, t)$ of S.
5 Calculate the mutual information and learn the tree structure.
6 Generate M new points from the tree.
7 $t \Leftarrow t + 1$
8 } **until** Termination criteria are met.

3.2.3 Recent theoretical developments in EDAs

Initial applications of probabilistic models in EDAs were mainly constrained to the transplantation of methods for learning and sampling probabilistic models to the context of population based optimization. Recently, emphasis has been given to the investigation of other important issues, e.g. analyzing the interactions between EDA component and

elucidating how a given optimization problem determines the nature and number of dependencies that arise in EDAs.

Some of the current research trends are fitness function approximation with probabilistic models (197; 201; 207), use of entropy based methods for theoretical analysis and improvement of EDAs (166; 169), analysis of the role of selection in the arousal of dependencies (77; 193), introduction of alternative sampling methods (95; 190; 207), and development of alternative probabilistic models to represent dependencies in EDAs (33; 148; 190; 219). The last two research trends, which are very related to each other, are particularly relevant to our work and will be treated in this thesis.

An example of alternative sampling methods in the context of EDAs is the use of the algorithms to calculate the most probable configurations. In (207), an algorithm that calculates the most probable configurations in the context of optimization by EDAs was introduced. The algorithm was applied to obtain the most probable configurations of the univariate marginal model, and models based on trees and polytrees. For the univariate model no BP is needed. For trees and polytrees BP is guaranteed to converge to the MAP solution. The results presented show that a UMDA that combines PLS with the computation of the most probable configuration was more efficient in terms of function evaluations than the UMDA that only uses PLS.

Another sampling procedure introduced together with region based decompositions for EDAs (see Chapter 4) is Gibbs sampling, which is applied to sample the Kikuchi approximation constructed from a clique-based decomposition of an independence graph (190). Clique-based decompositions are a particular type of region decompositions. Clique-based decompositions can be obtained from previous information about the problem or can be learned from data.

A common problem of EDAs that use sampling based on factorizations or Gibbs sampling is that the most probable configurations have an exponentially small probability. In this case, Monte Carlo sampling is not a favorable technique because, to hit the optimum, a large number of configurations needs to be visited before. In (95), region-based approximations were used to estimate and sample the probability in the context of EDAs. The local beliefs of a region graph are calculated using generalized belief propation (GBP) (245). Then, a factorization is used to sample promising points in the search space. In contrast to the work presented in (190), the sampling method employed avoids the use of the more expensive Gibbs sampling algorithm. Additionally, the important role of the inverse temperature β as a useful parameter to influence the search is analyzed.

3.3 Optimization methods based on inference

In contrast to EDAs, optimization methods based on inference (240; 241) are not iterative methods, in the sense that the probability model is constructed only once. There are no

subsequent generations of points. Instead, the idea is to build a probabilistic model from the given function in such a way that the most probable configuration using the model corresponds to the optimum of the problem. Exact or approximate inference can then be employed to find the most probable configuration of the model. When the original independence graph is chordal, a junction tree can be constructed (127). For these models, the most probable configuration can be calculated using dynamic programming (177). This technique was then extended to calculate the M most probable configurations (164; 242).

Although the number of published papers devoted to the use of optimization methods based on inference appears to be smaller than those devoted to EDAs, there are results on the application of approximate inference algorithms in optimization problems coming from the bioinformatics field (240; 242). In (240), the best max-marginal fit algorithm (BMMF) to calculate the most probable configurations, based on the use of loopy belief propagation is presented. BMMF is based on a dynamic programming method computationally more efficient than Nilsson's method (164). For the problems investigated, BMMF was able to find better most probable configurations than those obtained by Gibbs sampling and another method based on a greedy optimization of the posterior probability.

Propagation algorithms have also been employed to find the optimal solutions of constraint problems. In (35; 36), warning and survey propagation is introduced to solve the satisfiability problem (SAT). Warning and survey propagation belong to a new type of propagation algorithms that intend to find satisfiable assignments to a set of clauses. The algorithm uses factor graphs (119) to represent the structure of the problem and organizes the passing of messages from variables nodes to factor nodes and vice-versa. Although these message passing algorithms do not use region based decompositions, their purpose is also to find optimal solutions. The relationship between warning and survey propagation algorithms and sum-propagation algorithms is discussed in (35).

Part II

Region based decompositions

4 Kikuchi approximation of probability distributions

4.1 Introduction

Belief propagation (3; 177) is a well-known technique used in statistical inference to obtain a posteriori marginal probabilities in graphical models. Generalized belief propagation enables the class of models where these inference algorithms can be applied to be extended. Recent work on generalized belief propagation methods (245) have revealed the applicability that results achieved in the statistical physics domain, in the approximation of energy and entropy measures, have in the machine learning domain. One of the contributions from the field of statistical physics to inference algorithms comes from the use of region-based decompositions, like the Bethe (24) and Kikuchi (108; 154) approximations, in the context of generalized belief propagation.

Basically, a region-based decomposition of a function can be seen as a function defined on the variables associated to the vertices of a graph. The global function is formed by the composition of local subfunctions defined in those variables grouped in each of the regions. For instance, in the free energy approximation in physics, regions serve as the basic units to define the local energies, which are combined to give the global free energy function. Region-based decompositions can be used for the approximation of other measures. In this chapter, we use them to calculate suitable approximations of probability distributions. In this context, an essential question is how to determine a convenient region-based decomposition that maximizes the accuracy of the approximation. As we will see throughout the chapter, there are algorithms that can calculate these decompositions.

One particular case of such algorithms is the cluster variation method (CVM) (108; 154), originally introduced to obtain Kikuchi approximations of free energy in statistical physics. Starting from an initial set of regions defined in a graph, the CVM determines a way to obtain a whole set of regions where the free energy is decomposed. The CVM does not specify any particular choice for the initial regions. Nevertheless, the Kikuchi approximation clearly depends on this choice.

Some approaches that try to cope with the problem of selecting an appropriate set of initial regions have been published. This interest highlights the relevance of this prob-

lem for the field. The methods introduced have been proposed in the context of belief propagation algorithms (232), and are also related to work done on structural mean field methods (237; 238). The clique-based decomposition of the graph has been introduced in (190) as a particular way to select the initial regions for the CVM. The method was used for optimization by means of EDAs that employ Kikuchi approximations of the probability.

We focus on the clique-based decomposition as the method to select the initial regions of the Kikuchi approximation which is, in this case, an unnormalized approximation of the probability distribution. Therefore, this approach departs from common used schemes that look for consistent probability distribution approximations. There are a number of reasons that make this approach worth of consideration. First, in several situations, the computation of normalized distributions is infeasible. The number of terms to be considered in the partition function grows exponentially with the number of variables. Second, in some cases where belief propagation can be employed, there is not guarantee the method will converge. Third, recent applications of region-based decompositions (100; 190) show that they can be used not only for the common tasks of inference but can also be applied for optimization and classification. In these contexts, the decompositions will be used for tasks such as sampling that can be accomplished using unnormalized factorizations. Finally, even in cases in which normalized approximations are feasible, it is an important and unexplored area to investigate to what extent the unnormalized approximation is useful, and which properties it satisfies.

The approach followed in this chapter is to study some of the properties satisfied by clique-based Kikuchi approximations. The properties proven in this thesis are useful to achieve goals such as evaluating the quality of the approximation, the design of algorithms to learn Kikuchi approximations from data, and the design of sampling algorithms. Our work can be seen in the more general context of setting the theoretical basis for the application of unnormalized Kikuchi approximations as a practical approximation method in machine learning. In order to achieve this purpose, we investigate Markov and decomposability properties of the Kikuchi approximation constructed from clique-based decompositions of the graphs. The introduction of the concept of partial Kikuchi approximations allows to combine exact marginal with Kikuchi approximations of some of the irreducible components of the graph.

Additionally, we pay attention to recent developments in the application of region-based decompositions in machine learning. Our analysis intends to throw some light on the way these approximations are being conceived and applied in the field. We clarify the place of clique-based decompositions in relation to other techniques inspired by methods from statistical physics.

The chapter is organized as follows. Section 4.2, introduces clique-based decompositions and the Kikuchi approximation of the probability distribution. Section 4.3 proves that

the clique-based Kikuchi approximation satisfies the local, pairwise and global Markov properties with respect to the independence graph. Section 4.4 shows that the Kikuchi approximation can be factorized into the irreducible components of the graph. In Section 4.5, we argue that our work is part of a current research trend that benefits from the cross-fertilization between machine learning and statistical physics. Section 4.6 presents the conclusions of the chapter.

4.2 Kikuchi approximation: definitions

Kikuchi approximations of the energy (108) are region-based decompositions of the energy that satisfy certain constraints. The *Kikuchi approximation of a probability distribution from a clique-based decomposition of an independence graph* (190) is a particular type of factorization that uses marginal distributions. The marginals in the factorization are completely determined by the independence graph. Given this graph, the clique-based decomposition is formed by the maximal cliques of the graphs and their intersections. All these cliques are called regions.

Given a probability distribution $p(\mathbf{x})$, its independence graph $G = (V, E)$ associates one vertex with every variable of $\mathbf{X}$, and where two vertices are connected if the corresponding variables are conditionally dependent given the rest of the variables.

We define a region R of the independence graph $G = (V, E)$ of a probability distribution $p(\mathbf{x})$ as a subset of V. A graph region-based decomposition $(\mathcal{R}, U)$, is a set of regions $\mathcal{R}$ that covers V, and an associated set of *overcounting numbers* U which is formed by assigning one overcounting number c_R for each $R \in \mathcal{R}$. c_R will always be an integer, and might be zero or negative for some R. There are different methods that find region-based decompositions (4; 24; 108; 243), among them the CVM that learns Kikuchi approximations. In the CVM, $\mathcal{R}$ is formed recursively by an initial set of regions $\mathcal{R}_0$ such that each node is in at least one region of $\mathcal{R}_0$, and any other region in $\mathcal{R}$ is the intersection of one or more of the regions in $\mathcal{R}$. The set of regions $\mathcal{R}$ is closed under the intersection operation, and can be ordered as a partially ordered set (154).

To be valid, a decomposition must satisfy a number of constraints that relate $\mathcal{R}$ and U. Inspired by the work by (244), we call this sub-problem that of *finding a valid region-based decomposition of a graph*. A set of regions $\mathcal{R}$ and overcounting numbers U give a valid region-based graph decomposition (244) when for every variable X_i:

$$\sum_{\substack{R \in \mathcal{R} \\ X_i \subseteq \mathbf{X}_R}} c_R = 1 \tag{4.1}$$

Equation (4.1) states that, for any variable X_i, the sum of the overcounting numbers of regions that contain X_i is 1. Equation (4.2) can be obtained extending the previous constraint to every subset of variables $\mathbf{X}_S$ the following way:

$$\sum_{\substack{R\in\mathcal{R}\\ \mathbf{X}_S\subseteq\mathbf{X}_R}} c_R = 1 \tag{4.2}$$

where the sum of the overcounting numbers of regions that contain $\mathbf{X}_S$ is also 1. In (173), conditions represented by equations (4.1) and (4.2) are called balanced and totally balanced conditions, respectively.

We will apply the CVM making a particular choice of the initial regions. We will form set $\mathcal{R}_0$ by taking one region for each maximal clique in G. As a result, all the regions $R \in \mathcal{R}$ will be cliques because they are the intersection of two or more cliques. We call this type of region-based decomposition of undirected graphs a *clique-based decomposition*.

We define the Kikuchi approximation of the probability distribution $p(\mathbf{x})$ associated with a clique-based decomposition, $k(\mathbf{x})$, as:

$$k(\mathbf{x}) = \prod_{R\in\mathcal{R}} p(\mathbf{x}_R)^{c_R} \tag{4.3}$$

where $\mathcal{R}$ comes from a clique-based decomposition and the overcounting numbers c_R are constrained to be calculated using the following recursive formula:

$$c_R = 1 - \sum_{\substack{S\in\mathcal{R}\\ R\subset S}} c_S \tag{4.4}$$

where c_S is the overcounting number of any region S in $\mathcal{R}$ such that S is a superset of R. c_R values corresponding to the initial maximal cliques are equal to 1. If c_R is different from zero, the region is included in the clique-based decomposition. In (173) is proven that any collection of regions that is closed under intersection, and where overcounting number are calculated following 4.4 fulfills the balanced and totally balanced conditions (173).

From now on, when we refer to a Kikuchi approximation, we imply a Kikuchi approximation of the probability distribution obtained from a clique-based decomposition. Example 4.1 describes the Kikuchi approximation of the probability calculated a given graph.

Example 4.1. This example shows the Kikuchi approximation (Equation (4.1)) corresponding to the independence graph shown in Figure 4.1.

$$k(\mathbf{x}) = \frac{p(x_A,x_T)p(x_T,x_E)p(x_E,x_X)p(x_E,x_L)p(x_L,x_S)p(x_S,x_B)p(x_B,x_D)p(x_E,x_D)}{p(x_T)p(x_E)^3p(x_L)p(x_S)p(x_B)p(x_D)} \tag{4.5}$$

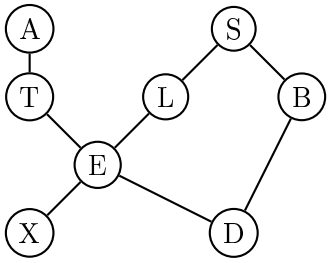


Figure 4.1: Undirected graph associated with the `Asia` network.

The factors of the decomposition correspond to the eight maximal cliques of the graph and their overlappings. The factor corresponding to variable X_E has an overcounting number 3 because it is included in four original cliques.

4.3 Markov properties of the Kikuchi approximation

Graphical models associated with probability distributions display a number of conditional and marginal independence properties that are stated by the Markov properties defined on the graph. These independence properties can be used for more efficient storing and sampling of probability distributions. Therefore, it is important to investigate which information about the properties of the Kikuchi approximation can be extracted from the graphical model where it is defined. In this section, we show that certain independence properties of the Kikuchi approximation can be deduced from the graph structure.

Firstly, we prove that the Kikuchi approximation satisfies the local, pairwise, and global Markov properties with respect to the independence graph.

4.3.1 Notation

We use the notation introduced in Section 2.2. The marginal and conditional functions of the Kikuchi approximation are defined as:

$$k(\mathbf{x}_S) = \sum_{\mathbf{x}'|\mathbf{x}'_S=\mathbf{x}_S} k(\mathbf{x}') \tag{4.6}$$

$$k(\mathbf{x}_A \mid \mathbf{x}_B) = \frac{k(\mathbf{x}_{\{A,B\}})}{k(\mathbf{x}_B)} \tag{4.7}$$

where $\{A, B\}$ is a simplified notation for $\{A \cup B\}$. Notice that, as we work with a positive probability distribution $p(\mathbf{x})$, marginal and conditional probability distributions are also positive. As $k(\mathbf{x})$ is not necessarily a probability distribution, neither are $k(\mathbf{x}_S)$ and $k(\mathbf{x}_A \mid \mathbf{x}_B)$. Nevertheless, they can respectively be used as approximations of $p(\mathbf{x}_S)$ and $p(\mathbf{x}_A \mid \mathbf{x}_B)$.

4.3.2 Markov properties

The Markov properties of a probability distribution $p(\mathbf{x})$ given its independence graph G are:

1. Pairwise Markov property: For all non-adjacent vertices X_i and X_j in G,

$$p(x_i, x_j \mid \mathbf{x} \setminus (x_i, x_j)) = p(x_i \mid \mathbf{x} \setminus (x_i, x_j))p(x_j \mid \mathbf{x} \setminus (x_i, x_j))$$

2. Local Markov property: For every vertex X_i in G,

$$p(x_i, \mathbf{x} \setminus cl(x_i) \mid bd(x_i)) = p(x_i|bd(x_i))p(\mathbf{x} \setminus cl(x_i)|bd(x_i))$$

3. Global Markov property: For all disjoint subsets $\mathbf{X}_A$, $\mathbf{X}_B$, and $\mathbf{X}_C$, whenever $\mathbf{X}_B$ and $\mathbf{X}_C$ are separated by $\mathbf{X}_A$ in the graph, in the sense that all paths from $\mathbf{X}_B$ to $\mathbf{X}_C$ go through $\mathbf{X}_A$, then:

$$p(\mathbf{x}_B, \mathbf{x}_C \mid \mathbf{x}_A) = p(\mathbf{x}_B \mid \mathbf{x}_A)p(\mathbf{x}_C \mid \mathbf{x}_A)$$

To prove that the Markov properties are satisfied by the Kikuchi approximation, we start from the fact that the Kikuchi approximation from Equation (4.3) is defined on the maximal cliques of the graph and can be treated as the unnormalized version of the probability distribution corresponding to the undirected graphical model.

Let us call the normalized distribution $q(\mathbf{x}) = \frac{k(\mathbf{x})}{Z_k}$, where Z_k is the normalizing constant calculated for $k(\mathbf{x})$. The conditional function of the Kikuchi approximation can be written as:

$$k(\mathbf{x}_A \mid \mathbf{x}_B) = \frac{k(\mathbf{x}_A, \mathbf{x}_B)}{k(\mathbf{x}_A)} = \frac{q(\mathbf{x}_A, \mathbf{x}_B)}{q(\mathbf{x}_A)} \tag{4.8}$$

The previous equality is given by the fact that $q(\mathbf{x}_A) = \frac{k(\mathbf{x}_A)}{Z_k}$ and that normalization constants cancel out each other. As a consequence, the proof of the following theorems is straightforward.

Theorem 4.1. *Given a Kikuchi approximation $k(\mathbf{x})$ defined on a graph G, and a set of variables $\mathbf{X}_A$,*

$$k(\mathbf{x}_A \mid \mathbf{x} \setminus \mathbf{x}_A) = k(\mathbf{x}_A \mid bd(\mathbf{x}_A))$$

Theorem 4.2 (Local Markov property). *Given a Kikuchi approximation $k(\mathbf{x})$ defined on a graph G, and a variable X_i,*

$$k(x_i, \mathbf{x} \setminus cl(x_i) \mid bd(x_i)) = k(x_i|bd(x_i))k(\mathbf{x} \setminus cl(x_i)|bd(x_i))$$

Theorem 4.3 (Pairwise Markov property). *Given a Kikuchi approximation $k(\mathbf{x})$ defined on a graph G, and two variables X_i and X_j, if the corresponding vertices are not joined in G:*

$$k(x_i, x_j \mid \mathbf{x} \setminus (x_i, x_j)) = k(x_i \mid \mathbf{x} \setminus (x_i, x_j))k(x_j \mid \mathbf{x} \setminus (x_i, x_j))$$

Theorem 4.4 (Global Markov property). *Given a Kikuchi approximation $k(\mathbf{x})$ defined on a graph G, for all disjoint subsets $\mathbf{X}_A$, $\mathbf{X}_B$, and $\mathbf{X}_C$, whenever $\mathbf{X}_B$ and $\mathbf{X}_C$ are separated by $\mathbf{X}_A$ in the graph, then:*

$$k(\mathbf{x}_B, \mathbf{x}_C \mid \mathbf{x}_A) = k(\mathbf{x}_B \mid \mathbf{x}_A)k(\mathbf{x}_C \mid \mathbf{x}_A)$$

Now we study the decomposition property of the Kikuchi approximation. Given any region A, we introduce $K(\mathbf{x}, A)$ and $\bar{K}(\mathbf{x}, A)$ for notational convenience, to represent $k(\mathbf{x})$ more concisely. If $K(\mathbf{x}, A) = \prod_{\substack{R \in \mathcal{R} \\ \mathbf{x}_R \cap \mathbf{x}_A \neq \emptyset}} p(\mathbf{x}_R)^{c_R}$, and $\bar{K}(\mathbf{x}, A) = \prod_{\substack{R \in \mathcal{R} \\ \mathbf{x}_R \cap \mathbf{x}_A = \emptyset}} p(\mathbf{x}_R)^{c_R}$, then $k(\mathbf{x})$ can be expressed in the following way:

$$\begin{aligned} k(\mathbf{x}) &= \prod_{\substack{R \in \mathcal{R} \\ \mathbf{x}_R \cap \mathbf{x}_A \neq \emptyset}} p(\mathbf{x}_R)^{c_R} \prod_{\substack{R \in \mathcal{R} \\ \mathbf{x}_R \cap \mathbf{x}_A = \emptyset}} p(\mathbf{x}_R)^{c_R} \\ &= K(\mathbf{x}, A)\bar{K}(\mathbf{x}, A) \end{aligned} \tag{4.9}$$

The non-standard notation $\sim \{\mathbf{x}_S\}$ will be used to represent the summation over all variables except $\mathbf{X}_S$, when $\mathbf{X}_S = \mathbf{x}_S$, obtaining:

$$p(\mathbf{x}_S) = \sum_{\sim\{\mathbf{x}_S\}} p(\mathbf{x}') = \sum_{\mathbf{x}' | \mathbf{x}'_S = \mathbf{x}_S} p(\mathbf{x}') \tag{4.10}$$

Notice the use of this notation in equation (4.11), which shows two different ways to calculate the marginal probabilities of $bd(\mathbf{X}_A)$.

$$\sum_{\sim cl(\mathbf{x}_A)} \sum_{\sim\{\mathbf{x}\setminus\mathbf{x}_A\}} p(\mathbf{x}') = \sum_{\sim bd(\mathbf{x}_A)} p(\mathbf{x}') \tag{4.11}$$

Equation (4.11), together with implications (4.12) and (4.13) below, are used in our proofs.

$$\forall R,\, \mathbf{X}_R \supseteq X_i \subset \mathbf{X}_A \Rightarrow \mathbf{X}_R \subseteq cl(\mathbf{X}_A) \tag{4.12}$$
$$\forall R,\, \mathbf{X}_R \not\supseteq X_i \subset \mathbf{X}_A \Rightarrow \mathbf{X}_R \subseteq \mathbf{X} \setminus \mathbf{X}_A \tag{4.13}$$

Implication (4.12) derives from the fact that, in the clique-based decomposition, any region is a clique. Therefore, any set of variables in the same region as $\mathbf{X}_A$ belongs to its closure. On the other hand, implication (4.13) represents the fact that, if none of the variables that are in region $\mathbf{X}_R$ belongs to $\mathbf{X}_A$, then the whole region $\mathbf{X}_R$ belongs to $\mathbf{X} \setminus \mathbf{X}_A$.

In what follows, $k_A\mathbf{x}$ will refer to the Kikuchi approximation constructed from the subgraph that includes vertices and edges that contain variables in $\mathbf{X}_A$.

Theorem 4.5 (Kikuchi decomposition property). *Given a Kikuchi approximation $k(\mathbf{x})$ defined on a graph G, such that $\mathbf{X} = \mathbf{X}_A \cup \mathbf{X}_B \cup \mathbf{X}_C$, and $\mathbf{X}_A$ is a separator of $\mathbf{X}_B$ and $\mathbf{X}_C$, then:*

$$k(\mathbf{x}) = \frac{k_{AB}(\mathbf{x}_A, \mathbf{x}_B) k_{AC}(\mathbf{x}_A, \mathbf{x}_C)}{k_A(\mathbf{x}_A)} \tag{4.14}$$

Proof.

From the global Markov property,

$$k(\mathbf{x}) = \frac{k(\mathbf{x}_A, \mathbf{x}_B) k(\mathbf{x}_A, \mathbf{x}_C)}{k(\mathbf{x}_A)}$$

Additionally, as $\mathbf{X}_A$, $\mathbf{X}_B$, and $\mathbf{X}_C$ are a partition of $\mathbf{X}$, the following equations are verified:

$$\begin{aligned} \mathbf{X}_R \cap \mathbf{X}_C = \emptyset \Rightarrow & \mathbf{X}_R \subseteq \mathbf{X}_{AB} \\ \mathbf{X}_R \cap \mathbf{X}_B = \emptyset \Rightarrow & \mathbf{X}_R \subseteq \mathbf{X}_{AC} \\ \mathbf{X}_R \cap \mathbf{X}_{BC} = \emptyset \Rightarrow & \mathbf{X}_R \subseteq \mathbf{X}_A \end{aligned}$$

Finally, from the definitions of $K(\mathbf{x}, A)$ and $\bar{K}(\mathbf{x}, A)$ (4.9) follows:

$$\bar{K}(\mathbf{x}, BC) = \bar{K}(\mathbf{x}, C) \bar{K}(\mathbf{x}, B) \tag{4.15}$$

Equation (4.15) is used in order to prove the result of Equation (4.14). Decomposing $k(\mathbf{x})$ by means of the global Markov property, we obtain:

$$
\begin{aligned}
k(\mathbf{x}) = \frac{k(\mathbf{x}_A, \mathbf{x}_B)k(\mathbf{x}_A, \mathbf{x}_C)}{k(\mathbf{x}_A)} &= \frac{\sum_{\mathbf{x}'_C} K(\mathbf{x}, C)\bar{K}(\mathbf{x}', C)\sum_{\mathbf{x}'_B} K(\mathbf{x}', B)\bar{K}(\mathbf{x}', B)}{\sum_{\mathbf{x}'_{BC}} K(\mathbf{x}', BC)\bar{K}(\mathbf{x}', BC)} \\
&= \frac{\bar{K}(\mathbf{x}, C)\bar{K}(\mathbf{x}, B)\sum_{\mathbf{x}'_C} K(\mathbf{x}', C)\sum_{\mathbf{x}'_B} K(\mathbf{x}', B)}{\bar{K}(\mathbf{x}, BC)\sum_{\mathbf{x}'_{BC}} K(\mathbf{x}', BC)} \\
&= \frac{\sum_{\mathbf{x}'_C} K(\mathbf{x}', C)\sum_{\mathbf{x}'_B} K(\mathbf{x}', B)}{\sum_{\mathbf{x}'_{BC}} K(\mathbf{x}', BC)} \\
&= \frac{\prod_{R\in\mathcal{R}} \mathbf{x}_R \subseteq \mathbf{x}_{A\cup B} p(\mathbf{x}_R)^{c_R} \prod_{R\in\mathcal{R}} \mathbf{x}_R \subseteq \mathbf{x}_{A\cup C} p(\mathbf{x}_R)^{c_R}}{\prod_{R\in\mathcal{R}} \mathbf{x}_R | \mathbf{x}_R \subseteq \mathbf{x}_A p(\mathbf{x}_R)^{c_R}} \\
&= \frac{K_{AB}(\mathbf{x}, (A, B))K_{AC}(\mathbf{x}, (A, C))}{K_A(\mathbf{x}, A)} \\
&= \frac{K_{AB}(\mathbf{x}_A, \mathbf{x}_B)K_{AC}(\mathbf{x}_A, \mathbf{x}_C)}{K_A(\mathbf{x}_A)} \qquad (4.16)
\end{aligned}
$$

□

To summarize the results proven in this section, we have shown that some of the properties of the independence graph are translated into the Kikuchi approximation. Pairwise, local, and global Markov properties are fulfilled. We have proven that it is possible to decompose the Kikuchi approximation in the product of local Kikuchi approximations defined from a decomposition of the graph.

4.4 Decomposability of the Kikuchi approximation

Decomposability is essential to handle feasible approximations of a probability. In this section, we show how the Kikuchi decomposition property will permit the definition of Kikuchi approximations in which each factor itself is a Kikuchi approximation corresponding to a subgraph of the original independence graph. We go one step further and define the class of partial Kikuchi approximations, in which some of the Kikuchi approximation components correspond to exact marginal probability distributions. First of all, we introduce a number of definitions, propositions and theorems, borrowed from (234), that lead to a factorization of the Kikuchi approximation based on the irreducible components of the independence graph.

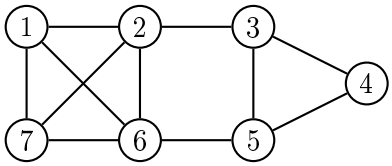


Figure 4.2: Independence graph with one incomplete maximal irreducible component and two complete ones.

4.4.1 Definitions

Definition 4.6 (Whittaker:1991, pp. 381). *There exists a decomposition of the random vector* $\mathbf{X}$ *with respect to a probability distribution* $p(\mathbf{x})$ *or equivalently,* $\mathbf{X}$ *is reducible, if and only if there exists a partition* $\mathbf{X}$ *into* $(\mathbf{X}_A, \mathbf{X}_B, \mathbf{X}_C)$ *such that:*

1. $p(\mathbf{x}_B, \mathbf{x}_C|\mathbf{x}_A) = p(\mathbf{x}_B|\mathbf{x}_A)p(\mathbf{x}_C|\mathbf{x}_A)$ *and neither* B *nor* C *are empty.*
2. *The subgraph on* A*, in the independence graph of* $\mathbf{X}$*, is complete.*

If so, the components of $\mathbf{X}$ *are* $\mathbf{X}_{AB} = (\mathbf{X}_A, \mathbf{X}_B)$ *and* $\mathbf{X}_{AC} = (\mathbf{X}_A, \mathbf{X}_C)$*. If such a decomposition does not exist,* $\mathbf{X}$ *is said to be irreducible.*

Example 4.2. Consider the graph in Figure 4.2 and the partition defined by sets $A = \{2, 6\}$, $B = \{1, 7\}$ and $C = \{3, 4, 5\}$. Then, $\mathbf{X}_{AB}$ and $\mathbf{X}_{AC}$ are the components that form a decomposition of the graph. $\mathbf{X}_{AB}$ is an irreducible component, and $\mathbf{X}_{AC}$ is a reducible one because it can be decomposed according to definition 4.6.

Definition 4.7. *The random vectors* $\mathbf{X}_{D_1}, \mathbf{X}_{D_2}, \ldots, \mathbf{X}_{D_m}$ *are the maximal irreducible components of* $\mathbf{X}$ *if and only if:*

1. *Each vector* $\mathbf{X}_{D_i}$ *is an irreducible component of* $\mathbf{X}$*.*
2. *No subset* D_i *is a proper subset of any other,* D_j*.*
3. $\mathbf{X} = \mathbf{X}_{D_1} \cup \mathbf{X}_{D_2} \cup \cdots \cup \mathbf{X}_{D_m}$

An irreducible component is said to be complete if it is a clique. Otherwise, it is called an incomplete irreducible component.

Example 4.3. $\mathbf{X}_{1,2,6,7}$, $\mathbf{X}_{2,3,5,6}$, and $\mathbf{X}_{3,4,5}$ are the maximal irreducible components of the graph shown in Figure 4.2. $\mathbf{X}_{1,2,6,7}$ and $\mathbf{X}_{3,4,5}$ are complete irreducible components. $\mathbf{X}_{2,3,5,6}$ is an irreducible component that is not complete.

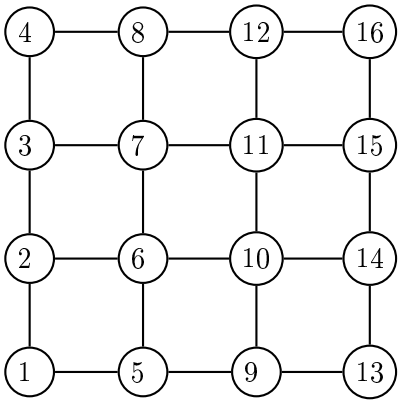


Figure 4.3: The grid lattice for the Ising and sping glass models is an irreducible incomplete component.

Proposition 4.8 (Whittaker:1991, pp. 385). *The maximal irreducible components of* $\mathbf{X}$ *corresponding to the subsets* $\{\mathbf{X}_{D_1}, \mathbf{X}_{D_2}, \ldots, \mathbf{X}_{D_m}\}$ *are unique, and the density function of* $\mathbf{X}$ *factorizes uniquely into* $f(\mathbf{x}) = \frac{f_{\mathbf{x}_{D_1}} f_{\mathbf{x}_{D_2}} \cdots f_{\mathbf{x}_{D_m}}}{g}$*, where function* g *is a product of marginal density functions,* $g = \prod f_{\mathbf{x}_A}$*, in which each subset* $\mathbf{X}_A$ *is an intersection of irreducible components, and it is complete.* $f(\mathbf{x})$ *is called an irreducible component factorization.*

Definition 4.9 (Whittaker:1991, pp. 389). *An* n*-dimensional random vector* $\mathbf{X}$*, or its density function, is decomposable if and only if there exists a sequence of decompositions to complete irreducible components.*

Independence graphs can be decomposed into irreducible components. The problem is known as decomposition by clique separators (214) or maximal prime subgraph decomposition (170) and it may be solved by certain algorithms (214). Maximal prime subgraph decompositions have been proposed in Bayesian networks as a computational structure for lazy propagation (170). However, in the case of Bayesian networks, this type of decomposition has also been criticized as a very limited representation of the independence relationships of this class of models (41).

One important remark is that irreducible incomplete components can be very large and, in fact, a graph can be formed by a unique irreducible component (e.g. the graph shown in Figure 4.3).

4.4.2 Decomposability of the Kikuchi approximation

We study how to extend the results achieved for the factorization of distributions to the Kikuchi approximation. We show that the Kikuchi approximation can be decomposed into the Kikuchi approximations of the irreducible components of the original graph. This result provides useful insight into ways to measure the accuracy of the Kikuchi approximation. It also allows the definition of the partial Kikuchi approximation which combines exact marginal with Kikuchi approximations of some of the irreducible components.

Theorem 4.10. *Given the independence graph G of $\mathbf{X}$, and the Kikuchi approximation $k(\mathbf{x})$ defined on G, if there exists a partition $\mathbf{X}$ into $(\mathbf{X}_A, \mathbf{X}_B, \mathbf{X}_C)$ such that the components of $\mathbf{X}$ are $\mathbf{X}_{AB} = (\mathbf{X}_A, \mathbf{X}_B)$ and $\mathbf{X}_{AC} = (\mathbf{X}_A, \mathbf{X}_C)$, then:*

$$k(\mathbf{x}) = \frac{k_{AB}(\mathbf{x}_{AB})k_{AC}(\mathbf{x}_{AC})}{k_A(\mathbf{x}_A)} \tag{4.17}$$

Proof.

This is a particular case of Theorem 4.5 when the separator $\mathbf{X}_A$ is a clique. □

Theorem 4.11. *Given the independence graph G, and a factorization*

$$p(\mathbf{x}) = \frac{p_{\mathbf{x}_{D_1}} p_{\mathbf{x}_{D_2}} \cdots p_{\mathbf{x}_{D_m}}}{\prod_{i=1}^{m-1} p_{\mathbf{x}_{d_i}}}$$

in which $\mathbf{x}_{d_1}, \mathbf{x}_{d_2}, \ldots, \mathbf{x}_{d_{m-1}}$ is the (possibly empty) set of complete irreducible components, resulting from the intersection of the irreducible components $\mathbf{x}_{D_1}, \mathbf{x}_{D_2}, \ldots, \mathbf{x}_{D_m}$, then, the Kikuchi approximation $k(\mathbf{x})$ defined on G can be decomposed as:

$$k(\mathbf{x}) = \frac{k_{D_1}(\mathbf{x}_{D_1})k_{D_2}(\mathbf{x}_{D_2}) \cdots k_{D_m}(\mathbf{x}_{D_m})}{\prod_{i=1}^{m-1} p_{\mathbf{x}_{d_i}}}$$

Proof.

The proof will be done by induction on the number of components, and using Theorem 4.10.

If $m = 1$, there is only one component $\mathbf{X}_{D_1}$ and no separators. In this case, $k(\mathbf{x}) = k(\mathbf{x}_{D_1})$.

Let us suppose that, for the component $\mathbf{X}_{(D_1, D_2, \ldots, D_{i-1})}$, the theorem holds, i.e.

$$k(\mathbf{x}_{(D_1, D_2, \ldots, D_{i-1})}) = \frac{k_{D_1}(\mathbf{x}_{D_1})k_{D_2}(\mathbf{x}_{D_2}) \cdots k_{D_{i-1}}(\mathbf{x}_{D_{i-1}})}{\prod_{j=1}^{i-2} p_{\mathbf{x}_{d_j}}}$$

Now we prove that, for $\mathbf{X}_{(D_1,D_2,\ldots,D_i)}$, the theorem is satisfied.

Let $A = d_{i-1}$, $B = \{D_1, D_2, \ldots, D_{i-1}\}$, and $C = D_i$. $(\mathbf{X}_A, \mathbf{X}_B, \mathbf{X}_C)$ is a partition of $\mathbf{X}_{(D_1,D_2,\ldots,D_i)}$ that satisfies the conditions of theorem 4.10. Therefore,

$$\begin{aligned}
&k(\mathbf{x}_{(D_1,D_2,\ldots,D_i)}) \\
&= \frac{k(\mathbf{x}_{(D_1,D_2,\ldots,D_{i-1})})k(\mathbf{x}_{D_i})}{k(\mathbf{x}_{d_i})} \\
&= \frac{k_{D_1}(\mathbf{x}_{D_1})k_{D_2}(\mathbf{x}_{D_2})\cdots k_{D_{i-1}}(\mathbf{x}_{D_{i-1}})}{\prod_{j=1}^{i-2} p_{\mathbf{x}_{d_j}}} \frac{k(\mathbf{x}_{D_i})}{k(\mathbf{x}_{d_i})} \\
&= \frac{k_{D_1}(\mathbf{x}_{D_1})k_{D_2}(\mathbf{x}_{D_2})\cdots k_{D_i}(\mathbf{x}_{D_i})}{\prod_{j=1}^{i-1} p_{\mathbf{x}_{d_j}}}
\end{aligned}$$

For all i, $\mathbf{X}_{d_i}$ is a complete irreducible component (i.e. clique). Thus, $k_{d_i}(\mathbf{x}_{d_i}) = p(\mathbf{x}_{d_i})$.

We have reached the decomposition of the Kikuchi approximation of $\mathbf{X}_{(D_1,D_2,\ldots,D_i)}$. To complete the proof, notice that $\mathbf{X} = \mathbf{X}_{(D_1,D_2,\ldots,D_m)}$. □

Example 4.4. We analyze the independence graph shown in Figure 4.2. The Kikuchi approximation of this graph is:

$$k(\mathbf{x}) = \frac{p(x_1, x_2, x_6, x_7)p(x_2, x_3)p(x_5, x_6)p(x_3, x_4, x_5)}{p(x_2)p(x_3)p(x_5)p(x_6)} \tag{4.18}$$

The factorization of $k(\mathbf{x})$ based on the irreducible components of the graph is:

$$k(\mathbf{x}) = \frac{k(x_1, x_2, x_7, x_6)k(x_2, x_3, x_5, x_6)k(x_3, x_4, x_5)}{k(x_2, x_6)k(x_3, x_5)} \tag{4.19}$$

The Kikuchi approximation of the incomplete irreducible component (X_2, X_3, X_5, X_6) is:

$$k_{(2,3,5,6)}(x_2, x_3, x_5, x_6) = \frac{p(x_2, x_3)p(x_3, x_5)p(x_5, x_6)p(x_2, x_6)}{p(x_2)p(x_3)p(x_5)p(x_6)} \tag{4.20}$$

Substituting Equation (4.20) in (4.19), and considering that the other factors are calculated from complete irreducible components, and that, therefore, the Kikuchi marginals coincide with the probability marginals, we obtain the original Kikuchi approximation shown in Equation (4.18).

Now we highlight another aspect related to the Kikuchi approximation of an independence graph: by identifying the regions of the independence graph where the Kikuchi approximation is localized (the incomplete components), we can estimate to what extent the Kikuchi approximation is used for the approximation of the distribution associated with a given graph. Therefore, we can obtain a measure of the approximation accuracy. Many irreducible components will imply more components approximated with the Kikuchi approximation.

Furthermore, we can constrain the use of the Kikuchi approximation to certain areas of the graph.

Given the independence graph G of $\mathbf{X}$, a partial Kikuchi approximation of the probability is one where only a subset of all the irreducible incomplete components are approximated by the corresponding Kikuchi approximation of the components. The rest of the components are calculated exactly.

The partial Kikuchi approximation admits the existence of components that are calculated exactly, i.e. they can be triangulated the same way methods for doing inference in graphical models usually do.

Let us suppose that the number of incomplete irreducible components in G is t. The number of partial Kikuchi approximation is 2^t-2, including the complete Kikuchi approximation. Hence, the total number of partial Kikuchi approximations is 2^t-1. Rationales for selecting one partial Kikuchi approximation rather than others might be related to the size of the induced triangulated clique (e.g. 4-sized cliques could be assumed to be triangulated). Another criterion can be the cardinality of the variables that are in the incomplete irreducible component (e.g. when the cardinality of the variables involved is high, the Kikuchi approximation is recommendable to diminish the number of parameters needed to approximate the model).

4.5 Related work on Kikuchi approximations

We have presented a number of properties fulfilled by the Kikuchi approximation constructed from clique-based decompositions. Now we relate these results to current research on similar topics.

4.5.1 Region based approximations

Kikuchi approximations are an example of a panoply of methods that deal with the problem of approximating a measure (i.e. energy, entrogy, probability) using graph-based decompositions. Initial applications of Bethe and Kikuchi approximations were constrained to the field of statistical physics (108; 153; 154). The purpose of finding a

way to decompose the otherwise difficult to handle free energy of a system led to the use of these approximations in physics. The idea was developed later by (245) in the context of generalized belief propagation. This contribution expanded the scope of application of belief propagation, which has been traditionally used in tasks like obtaining a posteriori marginal probabilities in graphical models (177), computing the most probable global states or system configurations (164), and improving the efficiency of iterative proportional fitting (IPF) (11; 216).

4.5.2 Generalized Kirkwood superposition

There is another path that leads to the notion of Kikuchi approximation treated in this chapter. The Kirkwood superposition (110) is an approximation for the three-body distribution of liquids introduced in liquid-state statistical mechanics. Since this approximation has been recently applied in the machine learning community (100), we elaborate on its relationship with the clique-based decomposition and the Kikuchi approximation.

The essence of the original Kirkwood superposition approximation is that all possible correlations in a system are expressed by binary relations. Although the approach has been widely applied in the theory of liquids, its implications and range of applications are controversial (see (84) for an extensive review on the subject). More relevant to our research is the derivation of the generalized Kirkwood superposition for the expansion of the information entropy in terms of correlation functions that has been proposed in (10).

In simple terms, this derivation proposes an approximation of the information entropy that includes successively higher-order correlations in a systematic fashion. The approach is extended by (141) to calculate the highest order mutual information $MI(\mathbf{X})$:

$$MI(X_1,\ldots,X_n) = (-1)^n \sum_{x_1,\ldots,x_n} p(x_1,\ldots,x_n) ln \frac{p(x_1,\ldots,x_n)}{\bar{p}(x_1,\ldots,x_n)} \tag{4.21}$$

where $\bar{p}(\mathbf{x})$ is the generalized Kirkwood superposition defined as:

$$\bar{p}(x_1,\ldots,x_n) = \prod_{k=1}^{n-1} (-1)^{k+1} \prod_{i_1<\ldots<i_{n-k}} p(x_1,\ldots,x_{n-k}) \tag{4.22}$$

where $\prod_{i_1<\ldots<i_{n-k}}$ runs over all possible combinations $\{i_1,\ldots,i_{n-k}\} \subseteq \{1,\ldots,n\}$ and $i_1 < i_2 < \cdots < i_{n-k}$.

There is a clear relationship between the generalized Kirkwood superposition and the Kikuchi approximation constructed from a clique-based decomposition. The Kirkwood superposition corresponds to a situation in which a complete graph is considered and, instead of choosing the single maximal clique of size n, all cliques of size $n-1$ are chosen

as the initial regions. Nevertheless, the higher-order contributions can be neglected from Equation (4.22), obtaining approximations that may yield good results for weakly correlated examples (141).

4.5.3 Research trends in the application of region-based decompositions

Concerning the field of machine learning, we identify two main current research trends in the application of region-based decompositions.

1. The use of region-based decompositions to design and improve inference methods, particularly, GBP algorithms.
2. The use of region-based decompositions to find approximate factorizations of probability distributions based on marginal probability distributions.

The first research trend (5; 56; 92; 147; 173; 215; 225; 226; 245; 246) includes work on the determination of efficient message passing schemes in BP, bounds on the accuracy of the inferred marginals, and conditions of convergence for the propagation algorithms. The second one (95; 100; 101; 190) studies the conception of measures to evaluate the accuracy of the learned approximations, algorithms to learn and sample these approximations from data, and the identification of significant interactions in data.

One common problem of both lines of research is the selection of the initial regions upon which the approximations are based. Some recent work on belief propagation algorithms addresses this problem using graph partition strategies (237; 238), sequential methods (232), and other approaches (173; 245).

In (101), the use of Kikuchi approximations for supervised classification has been proposed. A region-based decomposition learning algorithm is introduced with this objective. Region-based decompositions based on 2-way and 3-way interactions are tested. Higher order interactions are not considered.

In opposition to the GBP approach, clique-based decompositions are a way to automatically determine the initial regions of the graph that can be used to construct region-based decompositions. The local Markov property of the Kikuchi approximation was useful to define algorithms to learn the approximations from data (190) and sample the approximation obtained. However, results presented in (190) did not provide any measure to evaluate the accuracy of the approximation. Furthermore, it was not clear whether Kikuchi approximations could be applied locally in the probability distribution approximations. The properties presented in this thesis can be used (see Chapter 5) to define decomposable accuracy measures that can help to create more sophisticated methods for learning the approximations.

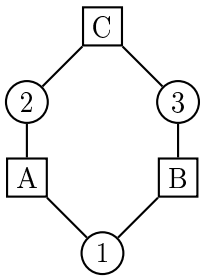


Figure 4.4: An example of a factor graph.

Another difference between the belief propagation approach and the learning approach is the type of graph in which the construction of the Kikuchi approximation is based, and its interpretation. In the original work of (243), the Kikuchi approximation was calculated using pairwise or higher-order Markov random fields (MRFs). Regions in the graph comprised the variables and the sets in which the potential functions were defined. Recent work (244) focuses on models defined on factor graphs (119). Factor graphs have variable and factor nodes. There is a variable node for each variable of the problem and a factor node for each node, with an edge connecting variable node i to factor node a if X_i is an argument of f_a.

In factor graphs, functions can represent some sort of interaction among their argument variables, but there is no requirement concerning the type and strength of these interactions. The validity condition of the region-based decomposition used by (244) establishes that every variable and factor node is counted once in the approximation (using the c_R values in the sum) but, apart from this requirement, the choice of the initial regions is arbitrary. There are other different ways to represent the region graph decompositions, which include Hasse diagrams (173), region graphs (244), and hypergraphs (226).

The work presented in this thesis can be extended to include different graph representations. The definition of the Kikuchi approximation given in Section 4.2 is based on an independence graph that is interpreted as a graphical model. However, the Kikuchi approximation constructed from a clique-based decomposition can be defined not only on undirected graphical models but also on a larger class of models equivalent to hierarchical log-linear models. These models can be graphically represented using factor graphs.

In the factor graph representation of a clique-based decomposition each maximal clique will have an associated factor. As in the case of hierarchical models, it is assumed that the existence of a clique means that all lower interactions are covered by the model. The

factor graph coincides with the maximal representation of the hierarchical models.

Example 4.5. Consider a probabilistic model with factors $\mathbf{X}_{12}$, $\mathbf{X}_{23}$ and $\mathbf{X}_{13}$. The Kikuchi approximation of this model would be

$$k(\mathbf{x}) = \frac{p(x_1, x_2)p(x_2, x_3)p(x_1, x_3)}{p(x_1)p(x_2)p(x_3)}$$

This approximation cannot be recovered from any undirected graph because the original model does not correspond to a log-linear graphical model (234). Figure 4.4 shows the factor graph representation of this model.

4.6 Conclusions

In this chapter, we have investigated a number of properties satisfied by the Kikuchi approximation constructed from the clique-based decomposition. We have shown that the Kikuchi approximation satisfies the local, pairwise, and global Markov properties. These results lay the foundations for the further development of algorithms that use Kikuchi approximations.

We have proposed a decomposition of the Kikuchi approximation according to the irreducible components of the graph. From this decomposition, we have introduced the notion of partial Kikuchi approximations. As a consequence of this result, an initial measure of the complexity of the Kikuchi approximation can be given, based on the number of irreducible components and their complexity (number of nodes and factors involved in the factorization). The results achieved indicate a way to investigate the accuracy of the Kikuchi and partial Kikuchi approximations.

5 Learning Kikuchi approximations from data

5.1 Introduction

An open problem related with region-based decompositions is how to select a set of initial regions that permits an accurate model-based approximation of the probability distribution. Some recent work addresses this problem using graph partition strategies (238) and sequential approaches (232).

A different approach has been taken in (101) in the context of supervised classification. A region-based decomposition learning algorithm is introduced to learn this type of decomposition from data. The algorithm is based on the use of goodness of fit testing. Therefore, a metric to evaluate the accuracy of the Kikuchi approximations is not provided. On the other hand, only region-based decompositions that use 2-way and 3-way interactions were tested. Higher order interactions were not considered. Nevertheless, the results obtained, when compared with traditional classification methods, showed the convenience of using Kikuchi approximations to solve supervised classification problems.

Our aim in this chapter is to study a related problem. Given a dataset, we investigate how to learn an accurate factorization of the probability distribution of the data. This factorization is calculated from a learned region-based decomposition of an underlying graphical model representing the dependencies in the data. We constrain our analysis to clique-based decompositions. Previous work on clique-based Kikuchi approximations was constrained to the study of probability distributions that arise in function optimization. In (190), an algorithm for learning clique-based decomposition from data was introduced. The algorithm learns the model by means of independence tests.

There are certain drawbacks behind learning clique-based decompositions from data using independence tests. Firstly, independence tests can be computationally costly and unreliable. Therefore, the use of higher order tests is impractical. Secondly, when the independence graph is very dense, the dimension of the cliques will increase beyond a feasible limit. Lastly, a measure of the learned approximation accuracy cannot be achieved using this type of approach.

In this chapter, we introduce a method to learn Kikuchi approximations from data by means of a score+search approach. The chapter is organized as follows: Section 5.2

begins describing the two most common approaches used for model learning. As we focus on score+search methods, a measure of the Kikuchi approximation accuracy is introduced. Section 5.3 presents an important property of the Kikuchi approximation that permits the decomposition of accuracy measures according to the graph structure. Section 5.4 introduces local and global search techniques that can be employed to find Kikuchi approximations that optimize the score. Section 5.5 reviews possible applications of the learning algorithm. Section 5.6 presents a number of experiments from different domains where the behavior of the learning algorithm is investigated and its performance validated in the approximation of distributions generated from Bayesian networks and Boltzmann distributions. Finally, Section 5.7 presents the conclusions of the chapter together with a number of lines for future research.

5.2 Learning approaches

Model search methods used in graphical models can be classified into two main groups according to the nature of the modeling: detecting conditional independencies versus score+search methods (90). Both approaches can be tested when learning Kikuchi approximations.

In (190), an algorithm for learning Kikuchi approximations by independence tests is presented. Its pseudocode is shown in Algorithm 5.1. To learn the undirected graph, the methodology proposed by Spirtes et al. (208) has been followed. The idea is to start from a complete undirected graph, and then try to remove edges by testing the conditional independence between the linked nodes, using conditioning sets as small as possible. After completing all the tests, the resulting graph is formed by all the remaining edges.

One limitation of this method is that, as the order of tests increases, their reliability decreases. Therefore, the use of higher order tests is not practical. Another drawback of Algorithm 5.1 is that, when the independence graph is very dense, the dimension of the cliques will increase beyond a feasible limit. An alternative to solve this problem is, at one step previous to the calculation of the cliques, to make the graph sparser. Nonetheless, the refinement step is only a partial solution to deal with this problem. See (190) for other limitations of this approach based on independence tests.

The reasons given above explain why the learning algorithm based on independence tests is an infeasible approach for general situations. This fact points to the need to explore the score+search approach for learning Kikuchi approximations from data. One necessary step is the definition of the score function.

Algorithm 5.1: Learning Kikuchi approximations based on independence tests

1. Learn an independence graph G from the data by using independence tests.
2. If necessary, refine the graph.
3. Find the set $\mathcal{C}$ of all the maximal cliques of G.
4. Construct a clique-based decomposition of the graph.
5. Find the marginal probabilities for the regions of the decomposition.

5.2.1 Accuracy of the Kikuchi approximation

An important issue is how to measure the accuracy of the Kikuchi approximation. First candidates are the accuracy measures used to compare probability distributions (e.g. the Kullback-Leibler divergence). In fact, the Kullback-Leibler divergence has been used in statistical physics to find good Bethe and Kikuchi approximations of the free energy (245).

The Kullback-Leibler divergence (KLD) (53) between the probability distribution $p(\mathbf{x})$ and the probability distribution $q(\mathbf{x})$ is defined as:

$$D(p||q) = \sum_{\mathbf{x}} p(\mathbf{x}) log \frac{p(\mathbf{x})}{q(\mathbf{x})} \tag{5.1}$$

We call the general Kikuchi approximation $k(\mathbf{x})$, and its normalized version $\tilde{p}(\mathbf{x})$,

$$\tilde{p}(\mathbf{x}) = \frac{k(\mathbf{x})}{\sum_{\mathbf{x}'} k(\mathbf{x}')} = \frac{k(\mathbf{x})}{Z_k}$$

We analyze the relationship between the divergence of general and normalized Kikuchi approximations with respect to the original probability distribution $p(\mathbf{x})$. Then, for the general Kikuchi approximation, equation (5.1) becomes:

$$D(p||k) = \sum_{\mathbf{x}} p(\mathbf{x}) log \frac{p(\mathbf{x})}{k(\mathbf{x})} \tag{5.2}$$

For $\tilde{p}(\mathbf{x})$,

$$D(p||\tilde{p}) = \sum_{\mathbf{x}} p(\mathbf{x}) log \frac{p(\mathbf{x})}{\tilde{p}(\mathbf{x})} \tag{5.3}$$

Since, in general, the Kikuchi approximation is not a probability distribution, properties proven for the Kullback-Leibler divergence are not sure to be satisfied when the Kikuchi

approximation is used instead of a probability distribution. For instance, it is not guaranteed that $D(p||k)$ will be always equal to or higher than zero. Nevertheless, it is always true that, if $k(\mathbf{x}) = p(\mathbf{x})$ for all $\mathbf{x}$, then $D(p||k) = 0$. Equation (5.2) can be seen as a relaxation of the Kullback-Leibler divergence.

We analyze $D(p||k)$ starting with its relationship with $D(p||\tilde{p})$.

$$\begin{aligned} D(p||k) &= \sum_{\mathbf{x}} p(\mathbf{x}) log \frac{p(\mathbf{x})}{k(\mathbf{x})} \\ &= \sum_{\mathbf{x}} p(\mathbf{x}) log \frac{p(\mathbf{x}) Z_k}{k(\mathbf{x})} - \sum_{\mathbf{x}} p(\mathbf{x}) log Z_k \\ &= D(p||\tilde{p}) - \sum_{\mathbf{x}} p(\mathbf{x}) log Z_k \\ &= D(p||\tilde{p}) - log Z_k \end{aligned} \tag{5.4}$$

If and only if $Z_k = 1$, then $D(p||\tilde{p}) = D(p||k)$. Otherwise, the sign of $logZ_k$ determines whether $D(p||\tilde{p}) < D(p||k)$ or $D(p||\tilde{p}) > D(p||k)$. At least apparently, $logZ_k > D(p||\tilde{p})$ is possible. In this case, $D(p||k)$ will be negative, which is not a desired property.

Let us consider a 'pathological' case where $k(\mathbf{x}) = \alpha p(\mathbf{x})$, $\forall \mathbf{x}$. In this situation, $Z_k = \sum_{\mathbf{x}} \alpha p(\mathbf{x}) = \alpha$, so $\tilde{p}(\mathbf{x}) = p(\mathbf{x})$. Hence, $D(p||\tilde{p}) = 0$, however $D(p||k) = -log\alpha$. Thus, it can be arbitrarily set. Nevertheless, the Kikuchi approximation $k(\mathbf{x})$ is not arbitrarily determined. It is calculated from the marginals of $p(\mathbf{x})$. Therefore, $k(\mathbf{x}) = \alpha p(\mathbf{x})$ seems to be very unlikely.

$D(p||k)$ and $D(p||\tilde{p})$ will be used as two alternative measures to orient the search. Obviously, the calculation of $D(p||\tilde{p})$ implies an exponential number of numerical operations. Nevertheless, $D(p||\tilde{p})$ is a more accurate measure of the approximation, because it refers to a divergence between probability distributions.

5.2.2 Search strategies

Heuristic algorithms used for model searching in the context of score+search methods can be roughly divided into algorithms that change a single link between the variables at each step, and algorithms that change more links of the graph. Single link lookahead search is usually employed with greedy or simulated annealing heuristics (122).

Although the interplay between scoring metrics and the search process has not been analyzed in detail, it is known that there is a class of probability models that cannot be learned by single link lookahead search procedures (e.g. pseudo-independent models (236)). Global search strategies are required for these situations. This type of strategies

can be designed for Kikuchi approximations. Nevertheless, in this chapter we concentrate on the study of local search techniques.

5.3 Finding Kikuchi approximations by edge addition

In this section, we derive a number of results related to the Kikuchi approximation. These results allow the conception of an algorithm to learn the Kikuchi approximation from data. An edge addition Kikuchi approximation learning algorithm is presented.

The following results will lay the foundations for the conception of a Kikuchi learning algorithm based on local modifications on the independence graph.

The local Markov property (4.2) determines that any subset of variables is independent of the rest given their neighborhood. Hence,

$$k(\mathbf{x}_A \mid \mathbf{x} \setminus \mathbf{x}_A) = k(\mathbf{x}_A \mid bd(\mathbf{x}_A)) \tag{5.5}$$

Now we prove that the computation of $k(\mathbf{x}_A)$ involves only cliques that contain variables in $\mathbf{X}_A$.

Theorem 5.1. *Given a Kikuchi approximation $k(\mathbf{x})$ defined on a graph G, and a set of variables $\mathbf{X}_A$, then:*

$$k(\mathbf{x}_A \mid bd(\mathbf{x}_A)) = \frac{K(\mathbf{x}, A)}{\sum_{\sim\{\mathbf{x}\setminus\mathbf{x}_A\}} K(\mathbf{x}, A)} \tag{5.6}$$

Proof.

$$\begin{aligned}
&k(\mathbf{x}_A \mid bd(\mathbf{x}_A)) \\
&= \frac{k(cl(\mathbf{x}_A))}{k(bd(\mathbf{x}_A))} \\
&= \frac{\sum_{\sim\{cl(\mathbf{x}_A)\}} K(\mathbf{x}, A)\bar{K}(\mathbf{x}', A)}{\sum_{\sim\{\mathbf{x}\setminus\mathbf{x}_A\}} \sum_{\sim\{cl(\mathbf{x}_A)\}} K(\mathbf{x}', A)\bar{K}(\mathbf{x}', A)} \\
&= \frac{K(\mathbf{x}, A)}{\sum_{\sim\{\mathbf{x}\setminus\mathbf{x}_A\}} K(\mathbf{x}', A)} \frac{\sum_{\sim\{cl(\mathbf{x}_A)\}} \bar{K}(\mathbf{x}', A)}{\sum_{\sim\{cl(\mathbf{x}_A)\}} \bar{K}(\mathbf{x}', A)} \\
&= \frac{K(\mathbf{x}, A)}{\sum_{\sim\{\mathbf{x}\setminus\mathbf{x}_A\}} K(\mathbf{x}', A)}
\end{aligned}$$

□

An important question is whether it is possible to construct a Kikuchi approximation that locally optimizes a predefined score by making local changes to the graph. We now show that this can, indeed, be achieved.

Theorem 5.2. *Given two Kikuchi approximations $k_1(\mathbf{x})$ and $k_2(\mathbf{x})$, such that their respective independence graphs are identical except perhaps in some of the edges that join vertices in $\mathbf{X}_A \subset \mathbf{X}$, then $\forall \mathbf{x}$*

$$k_1(\mathbf{x}) = k_2(\mathbf{x})\delta(cl(\mathbf{X}_A)) \tag{5.7}$$

where δ is a function that only depends on $cl(\mathbf{X}_A)$.

Proof.

Using the local Markov property, a simplified expression of $k(\mathbf{x})$ can be found:

$$\begin{aligned} k(\mathbf{x}) &= k(\mathbf{x}_A, \mathbf{x} \setminus cl(\mathbf{x}_A) \mid bd(\mathbf{x}_A))k(bd(\mathbf{x}_A)) \\ &= k(\mathbf{x}_A \mid bd(\mathbf{x}_A))k(\mathbf{x} \setminus cl(\mathbf{x}_A) \mid bd(\mathbf{x}_A))k(bd(\mathbf{x}_A)) \\ &= k(\mathbf{x}_A \mid bd(\mathbf{x}_A))k(\mathbf{x} \setminus cl(\mathbf{x}_A), bd(\mathbf{x}_A)) \\ &= k(\mathbf{x}_A \mid bd(\mathbf{x}_A))k(\mathbf{x} \setminus \mathbf{x}_A) \end{aligned}$$

Noticing that

$$\begin{aligned} k(\mathbf{x} \setminus \mathbf{x}_A) &= \sum_{\sim\{\mathbf{x}\setminus\mathbf{x}_A\}} K(\mathbf{x}, A) \cdot \bar{K}(\mathbf{x}, A) \\ &= \bar{K}(\mathbf{x}, A) \sum_{\sim\{\mathbf{x}\setminus\mathbf{x}_A\}} K(\mathbf{x}', A) \end{aligned} \tag{5.8}$$

and substituting $k(\mathbf{x}_A \mid bd(\mathbf{x}_A))$ by the expression found in Theorem 4.1, we obtain:

$$\begin{aligned} \frac{k_1(\mathbf{x})}{k_2(\mathbf{x})} &= \frac{k_1(\mathbf{x}_A \mid bd(\mathbf{x}_A))k_1(\mathbf{x} \setminus \mathbf{x}_A)}{k_2(\mathbf{x}_A \mid bd(\mathbf{x}_A))k_2(\mathbf{x} \setminus \mathbf{x}_A)} \\ &= \frac{K_1(\mathbf{x}, A)\sum_{\sim\mathbf{x}\setminus\mathbf{x}_A} K_2(\mathbf{x}, A)}{K_2(\mathbf{x}, A)\sum_{\sim\mathbf{x}\setminus\mathbf{x}_A} K_1(\mathbf{x}, A)} \frac{\bar{K}_1(\mathbf{x}, A)\sum_{\sim\{\mathbf{x}\setminus\mathbf{x}_A\}} K_1(\mathbf{x}, A)}{\bar{K}_2(\mathbf{x}, A)\sum_{\sim\{\mathbf{x}\setminus\mathbf{x}_A\}} K_2(\mathbf{x}, A)} \\ &= \frac{K_1(\mathbf{x}, A)}{K_2(\mathbf{x}, A)} \\ &= \delta(cl(\mathbf{x}_A)) \end{aligned} \tag{5.9}$$

The equality between $\bar{K_1}(\mathbf{x}, A) = \bar{K_2}(\mathbf{x}, A)$ holds because the independence graphs of $k_1(\mathbf{x})$ and $k_2(\mathbf{x})$ are identical for $\mathbf{X} \setminus cl(\mathbf{X}_A)$. Therefore, they share the same clique-based decompositions in these subgraphs, and the products or their marginals are equal. □

We are interested in evaluating the gain in accuracy given the addition of edges between variables in $\mathbf{X}_A$.

Corollary 5.3. *Given the same assumptions required for Theorem 5.2,*

$$D(p||k_2) - D(p||k_1) = \sum_{\mathbf{x}} p(\mathbf{x}) log \left(\frac{K_1(\mathbf{x}, A)}{K_2(\mathbf{x}, A)} \right) \tag{5.10}$$

Proof.

$$\begin{aligned} D(p||k_2) - D(p||k_1) &= \sum_{\mathbf{x}} p(\mathbf{x}) log \left(\frac{p(\mathbf{x})}{k_2(\mathbf{x})} \right) - \sum_{\mathbf{x}} p(\mathbf{x}) log \left(\frac{p(\mathbf{x})}{k_1(\mathbf{x})} \right) \\ &= \sum_{\mathbf{x}} p(\mathbf{x}) log(k_1(\mathbf{x})) - \sum_{\mathbf{x}} p(\mathbf{x}) log(k_2(\mathbf{x})) \\ &= \sum_{\mathbf{x}} p(\mathbf{x}) log \left(\frac{k_1(\mathbf{x})}{k_2(\mathbf{x})} \right) \\ &= \sum_{\mathbf{x}} p(\mathbf{x}) log \left(\frac{K_1(\mathbf{x}, A)}{K_2(\mathbf{x}, A)} \right) \end{aligned} \tag{5.11}$$

□

An important particular case of Theorem 5.2 and Corollary 5.3 is when $\mathbf{X}_A$ comprises only two vertices, and the change from $k_1(\mathbf{x})$ to $k_2(\mathbf{x})$ is due to the addition of edge $X_i \sim X_j$. To evaluate the gain due to the addition of this edge, it is enough to consider $cl(X_i, X_j)$. Therefore, the change in divergence can be locally computed.

Example 5.1. Let us consider the Kikuchi approximation shown in (5.12). This approximation corresponds to the independence graph shown in Figure 5.1.

$$k(\mathbf{x}) = \frac{p(x_1, x_2, x_3)p(x_3, x_4, x_5)p(x_1, x_6, x_7)p(x_5, x_6, x_8)p(x_9)}{p(x_1)p(x_3)p(x_5)p(x_6)} \tag{5.12}$$

Notice that there are five maximal cliques, as well as their overlappings. Now we consider the effect that the addition of two different edges to the independence graph has on the new Kikuchi approximation.

Equation (5.13) shows the new Kikuchi approximation obtained after adding edge $X_2 \sim X_4$. The equation has been factorized into two main terms. The first term is formed by

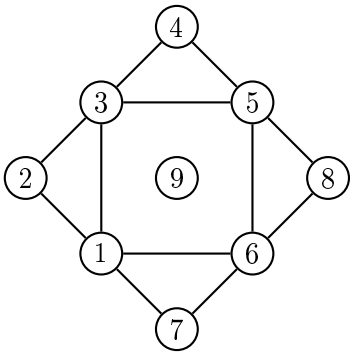


Figure 5.1: Example of an independence graph.

cliques that contain variables in $cl(X_2, X_4)$. The second term is formed by the ones that only contain variables in $(\mathbf{X} \setminus cl(X_2, X_4)$.

$$\begin{aligned} & k_1(\mathbf{x}) \\ & = \frac{p(x_1, x_2, x_3)p(x_3, x_4, x_5)p(x_2, x_3, x_4)p(x_1, x_6, x_7)p(x_5, x_6, x_8)p(x_9)}{p(x_2, x_3)p(x_3, x_4)p(x_1)p(x_3)p(x_5)} \\ & \cdot \frac{1}{p(x_6)} \end{aligned} \tag{5.13}$$

Equation (5.14) shows the new Kikuchi approximation obtained after adding the edge $X_8 \sim X_9$ to the graph in Figure 5.1. In this case, a size two clique of is added to the factorization.

$$\begin{aligned} k_2(\mathbf{x}) = & \frac{p(x_3, x_4, x_5)p(x_2, x_3, x_4)p(x_1, x_6, x_7)p(x_5, x_6, x_8)p(x_8, x_9)}{p(x_5)p(x_6)p(x_8)} \\ & \cdot \frac{p(x_1, x_2, x_3)}{p(x_1)p(x_3)} \end{aligned} \tag{5.14}$$

From the comparison between the Kikuchi approximations $k_1(\mathbf{x})$ and $k_2(\mathbf{x})$ obtained in Example 5.1, and the original Kikuchi approximation $k(\mathbf{x})$, it can be seen that the second main factor of $k_1(\mathbf{x})$ (respectively of $k_2(\mathbf{x})$) containing variable X_6 (repectively X_1, X_2 and X_3) was already present in Equation (5.12). As proven in Theorem 5.2, the addition of one edge only causes changes in cliques that contain variables in the closure of the vertices that form the edge. The choice of the edge to be added determines the extent of the necessary modification to the original clique-based decomposition. With $\mathbf{X}_A$ being

the region to be added, in Equation (5.13) only one clique belongs to $X \setminus cl(\mathbf{X}_A)$. In Equation (5.14), three cliques belong to this region.

5.4 Algorithms to learn Kikuchi approximations from data

Theorem 5.2 enables the reuse of the marginal probability distributions already calculated for variables that are in regions that do not contain variables $cl(X_i, X_j)$. The rest of the regions are recalculated as the intersections of the maximal cliques, and their overcounting values are found using Equation (4.4). The initial maximal cliques produced by the addition of the edge are created finding subsets of maximal cliques that are joined both to X_i and X_j.

Corollary 5.3 also facilitates the design of an algorithm to learn the Kikuchi approximation from data by changing only one edge at a time. The pseudocode of the algorithm to update the Kikuchi approximation is shown in Algorithm 5.2.

Algorithm 5.2: Updating the Kikuchi approximation by $X_i \sim X_j$ addition

1. Separate the regions corresponding to the current Kikuchi approximation $k(\mathbf{x})$ into two sets O and Q. O contains regions with variables in $\mathbf{X} \setminus cl(X_i, X_j)$. The remaining set of regions is called Q.
2. Form the set S with the intersections between every possible pair of maximal cliques in Q, such that one of the two cliques contains X_i and the other one, X_j.
3. Add variables X_i and X_j to every clique in S.
4. Find the set of new regions Q from the maximal cliques in S and the rest of maximal cliques that do not contain X_i or X_j.
5. Calculate the c_Q values for the cliques in Q applying Equation (4.4).
6. Remove cliques with $c_Q = 0$ from Q.
7. From the union of regions in O and Q, compute the new Kikuchi approximation $k'(\mathbf{x})$.

Example 5.2. This example presents the application of the steps shown in Algorithm 5.2 to the initial Kikuchi approximation $k(\mathbf{x})$ presented in Example 5.1, when edge $X_2 \sim X_4$ is added.

Step 1 $O = \{\{X_6\}, \{X_9\}\}$
$Q = \{\{X_1, X_2, X_3\}, \{X_3, X_4, X_5)\}, \{X_1, X_6, X_7\}, \{X_5, X_6, X_8\}, \{X_1\}, \{X_3\}, \{X_5\}\}$

Step 2 $S = \{\{X_3\}\}$

Step 3 $S = \{\{X_2, X_3, X_4\}\}$

Step 4 $Q = \{\{X_1, X_2, X_3\}, \{X_3, X_4, X_5)\}, \{X_2, X_3, X_4\}, \{X_1, X_6, X_7\}, \{X_5, X_6, X_8\}, \{X_2, X_3\}, \{X_3, X_4\}, \{X_1\}, \{X_5\}, \{X_3\}\}$

Step 5 $c_Q = \{1, 1, 1, 1, 1, 1, -1, -1, -1, -1\}$

Step 6 The final Kikuchi approximation is shown in Example 5.1, Equation 5.13

5.4.1 Search methods

We now present a greedy-based local search method to find the Kikuchi approximation that optimizes the score.

Every possible subgraph of the independence graph $G = (V, E)$ has a different set of maximal cliques and thus represents a different Kikuchi approximation. There are $2^{|E|}$ possible Kikuchi approximations corresponding to every subset of edges of G. When no information is available in the form of an independence graph, we assume that all the edges could be added. Therefore, the maximum number of edges that can be considered for addition are $\frac{n(n-1)}{2}$.

The local search procedure is a greedy algorithm that, at each iteraction, analyzes all possible one-edge additions to the current graph. Solutions are transformed into clique-based decompositions and the corresponding Kikuchi approximation is evaluated using Equation (5.2). Transition to the Kikuchi approximation with the best score is done if its score is better than the current one. The algorithm ends when improvement is not higher than a given threshold γ. The pseudocode is shown in Algorithm 5.3. The algorithm begins with an empty graph.

5.5 Applications of the learning algorithm

Algorithm 5.3 can be applied to learn probability models from data. The convenience of applying the learning algorithm is given by the fact that it allows an alternative representation to traditional models. As discussed in Chapter 4, the complexity of the Kikuchi approximation depend on the size of the maximum clique of the graph and the cardinality of the variables. The complexity of other models depend on the size of the

Algorithm 5.3: Kikuchi approximation learning algorithm

1. Start from a complete disconnected graph.
2. **do** {
3. Calculate marginal probabilities for the cliques of the graph.
4. $best_g = 0$.
5. Mark all vertices in the graph to be updated.
6. For every disconnected pair of vertices (X_i, X_j) in which at least one of the vertices has been marked for updating.
7. Calculate the Kikuchi approximation of $cl_1(X_i, X_j)$.
8. Add edge $(X_i \sim X_j)$ to the graph.
9. Calculate the Kikuchi approximation of $cl_2(X_i, X_j)$.
10. Calculate the gain in the approximation $g(cl_1, cl_2)$ using a score metric (e.g. Equation (5.2)).
11. If $g(cl_1, cl_2) > best_g$, then $best_g = g(cl_1, cl_2)$.
12. If $best_g > \gamma$.
13. Add the corresponding edge $(X_i \sim X_j)$ to the graph.
14. Mark all vertices in $cl(X_i \sim X_j)$ to be updated and unmark the rest.
15. } **until** $best_g \leq \gamma$.

maximum clique of the triangulated graph. Additionally, partial Kikuchi approximations can be used to approximate only some groups of variables corresponding to regions of the graph.

There is a number of contexts where this problem rises. One example is optimization. Given the results achieved by EDAs that use Kikuchi approximations learned by means of independence tests, the use of the Kikuchi learning algorithm in this context is a promising research area. One step to study the convenience of applying the learning algorithm in optimization is the study of probability distributions calculated from fitness functions. In this sense, a natural candidate is the Boltzmann distribution. Similarly, Kikuchi approximations learned from probability distributions encoded by Bayesian networks serve to study the differences between these two types of models. In this thesis, we present results on the use of Algorithm 5.3 for approximating the Boltzmann distribution calculated from a fitness function and learning Kikuchi approximations of points generated from Bayesian networks.

The learning algorithm can be employed in supervised classification problems by considering the class variable part of the joint distribution and using its conditional probability given the rest of variables to classify incumbent points. Given a classification problem where $X_1, \ldots, X_{n-1}$ are the predictive variables and X_n is the class variable with r

possible values, the Kikuchi approximation $k(\mathbf{x})$ is learned from the empirical joint probability distribution calculated from the data. Given the values of the predictive variables, classification is done by selecting the highest Kikuchi value of the class variable from $k(x_n \mid x_1, \ldots, x_{n-1}) \approx \frac{k(x_1,\ldots,x_n)}{k(x_1,\ldots,x_{n-1})}$.
Algorithm 5.3 can be modified by considering the gain in the classification accuracy given by the inclusion of each edge during the search. However, this approach would clearly be more expensive than the use of the KLD.

5.6 Experiments

Experiments investigate the performance of Algorithm 5.3 in learning Kikuchi approximations of distributions generated from Bayesian networks and Boltzmann distributions, and in the solution of classification problems. In the first set of experiments, the Kikuchi approximation is learned directly from the known exact joint probability distribution. The use of the exact distribution allows us to focus on the ability of the learning algorithm to find accurate Kikuchi approximations, giving an exact measure of the divergence between the target distribution and the approximations.

In the rest of the experiments, the exact probability distribution is not available. A dataset is used instead. Marginal distributions are learned from the dataset using maximum likelihood estimation. The divergence is calculated between the empirical distribution and the Kikuchi approximation. In the classification experiments, the KLD is minimized and, in every step, the classification accuracy of the best approximation is determined.

5.6.1 Bayesian network `Asia`

The aim of the first experiment is to calculate the Kikuchi approximation of a well-known Bayesian network. We analyze the behavior of Algorithm 5.3 for the `Asia` network (126), which calculates the probability of a patient to have tuberculosis, lung cancer or bronchitis based on a number of factors. The structure of the network is shown in Figure 5.2.

First, the joint probability distribution determined by the Bayesian network is calculated from the conditional distributions. A minor modification is done to the conditional probabilities to guarantee that the joint distribution is positive. Conditional probabilities equal to zero are replaced by 10^{-8}, and complementary probabilities are modified accordingly. This is our target probability distribution $p(\mathbf{x})$. It is defined on a domain of eight binary variables. The joint probability distribution can also be calculated using the factorization determined by a triangulation of the graph. Figures 5.3a) and 5.3b) respectively show the moralized graph and a possible triangulation of the `Asia` network.

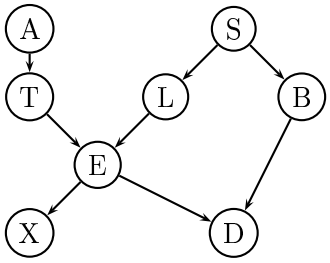


Figure 5.2: `Asia` network.

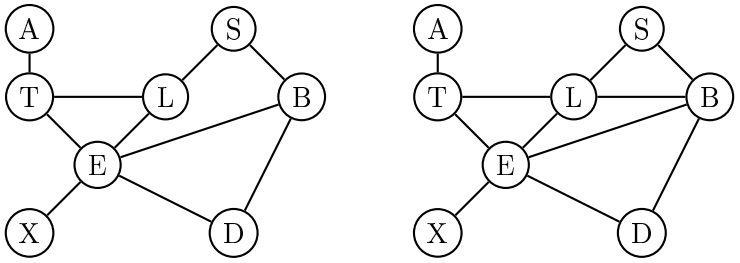


Figure 5.3: a) Moralization of the `Asia` network; b) One possible triangulation.

Initially, the learning algorithm is applied using the score $D(p||k)$ presented in Section 5.2.1. The learning algorithm receives the joint probability distribution $p(\mathbf{x})$ and outputs a Kikuchi approximation in the form of a clique-based decomposition of the independence graph. The stop criterion value γ is set at 0.001.

Figures 5.4 and 5.5 show the values that describe the accuracy of the approximation at each step. Figure 5.6 b) shows the edges that are added at each step of the algorithm. Each time an edges is added we have a different Kikuchi approximation.

The analysis of Figures 5.4, 5.5, and 5.6 reveals that the algorithm can learn accurate factorizations of $p(\mathbf{x})$ that are not probability distributions. During the first seven steps the approximations are probability distributions. At iteration 8, the learned Kikuchi approximation does not correspond to a probability distribution. However, $D(p||k)$ is smaller than in the previous iteration. The corresponding distribution $\tilde{p}(\mathbf{x})$ also gives a better approximation than the factorization obtained in the previous iteration. Although the $D(p||k)$ value corresponding to iteration 9 indicates that an improvement has been achieved in the approximation, the calculation of $D(p||\tilde{p})$ shows that $\tilde{p}(\mathbf{x})$ does not improve the results achieved at the previous step. We conclude that $D(p||k)$ is not always a good indicator of the accuracy of $\tilde{p}(\mathbf{x})$. It serves as a rough estimator.

Finally, we find the Kikuchi approximations calculated from the clique-based decompositions computed using the structures of the initial undirected and moralized graphs.

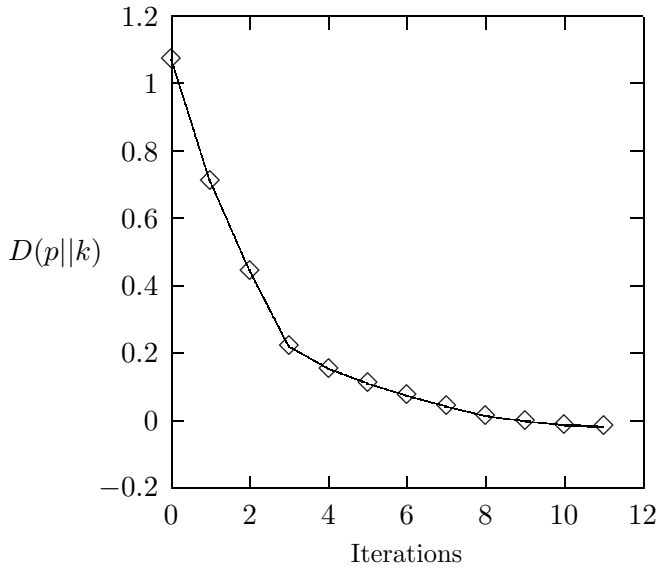


Figure 5.4: Kullback-Leibler divergence between $p(\mathbf{x})$ and the Kikuchi approximation at different iterations of the learning algorithm.

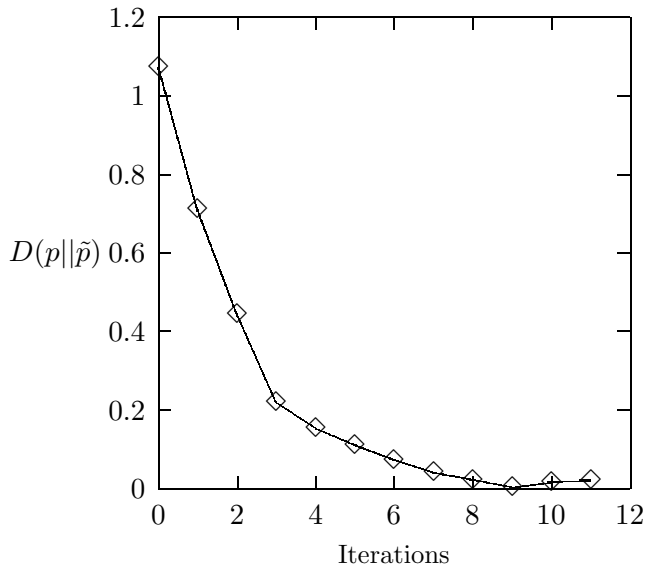


Figure 5.5: Kullback-Leibler divergence between $p(\mathbf{x})$ and the normalized Kikuchi approximation at different iterations of the learning algorithm.

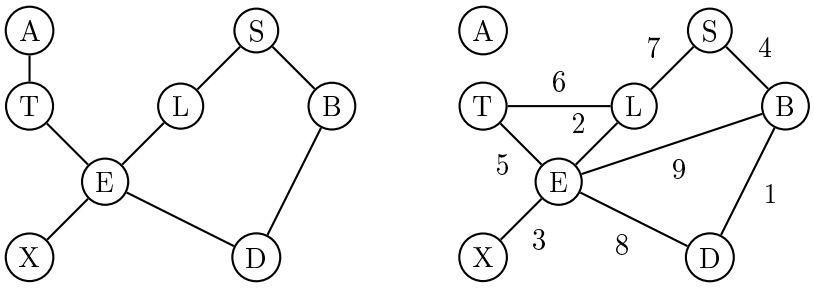


Figure 5.6: Kikuchi approximations of `Asia` network: a) Computed from the undirected graph; b) Learned by the algorithm.

In Table 5.1, different statistics corresponding to the probability distributions are presented. The table shows this information about the original probability distribution (E), the Kikuchi approximations calculated from the undirected (U) and moralized (M) graph, and the Kikuchi approximations learned by the learning algorithm (A).

In the table, $\sum_{\mathbf{x}} k(\mathbf{x})$ refers to the sum of the probabilities of the distribution or Kikuchi approximation. This value is 1 only when $k(\mathbf{x})$ is a probability distribution. $D(p||k)$ is the divergence between $p(\mathbf{x})$ and $k(\mathbf{x})$. For the cases where $k(\mathbf{x})$ is not a probability distribution, we include an additional row where the $D(p||\tilde{p})$ value is also shown. This allows us to appreciate the change due to the probability normalization. The last colums of the table represent the univariate marginal values of $k(\mathbf{x})$ for all the variables. Notice that, since the univariate values do not necessary add up to 1, both values need to be shown. The former row shows the values for $k(X_i = 0)$ and the latter, $k(X_i = 1)$.

The approximation achieved from the moralized graph is better than the one calculated from the original undirected graph. At least from this Bayesian network, it is clear that the dependencies represented in the moral graph cannot be discarded without incurring an important increase in the approximation error. Another conclusion is that the approximation achieved from the moral graph is only slightly better than the one achieved with the learning algorithm. Whenever the graphical structure of the probabilistic dependencies arising in the data, in the form of an undirected graph, is available, it can be used to find the corresponding clique-based decomposition and the associated Kikuchi approximation.

5.6.2 Approximation of Boltzmann probability distributions

The objective of the following experiment is to evaluate the performance of the learning algorithm in the approximation of a Boltzmann distribution. This type of distribution is

	$\sum_{\mathbf{x}} k(\mathbf{x})$	$D(p\|\|k)$	k_A	k_S	k_T	k_L	k_B	k_E	k_X	k_D
E	1.0000	0.0000	0.9900	0.5000	0.9896	0.9450	0.55	0.9352	0.8897	0.5640
			0.0100	0.5000	0.0104	0.0550	0.45	0.0648	0.1103	0.4360
U	1.0074	0.0332	1.0498	0.5302	1.0483	0.9952	0.5825	0.9849	0.9372	0.5971
			0.0106	0.5302	0.0121	0.0652	0.4780	0.0755	0.1232	0.4633
	1.0000	0.0406	0.9900	0.5000	0.9886	0.9385	0.5493	0.9288	0.8838	0.5631
			0.0100	0.5000	0.0114	0.0615	0.4507	0.0712	0.1162	0.4369
M	1.0024	−0.0015	0.0138	0.5120	1.0134	0.9650	0.5629	0.9550	0.9056	0.5799
			0.0102	0.5120	0.0136	0.0590	0.4611	0.0690	0.1154	0.4441
	1.0000	**0.0015**	0.9900	0.5000	0.9896	0.9424	0.5497	0.9326	0.8673	0.5663
			0.0100	0.5000	0.0104	0.0576	0.4503	0.0674	0.1127	0.4337
A	1.0030	−0.0011	0.9929	0.5015	0.9926	0.9452	0.5513	0.9354	0.8899	0.5680
			0.0100	0.5015	0.0104	0.0578	0.4516	0.0676	0.1130	0.4350
	1.0000	0.0019	0.9900	0.5000	0.9896	0.9424	0.5497	0.9326	0.8873	0.5663
			0.0100	0.5000	0.0104	0.0576	0.4503	0.0674	0.1127	0.4337

Table 5.1: Original distribution and three Kikuchi approximations of the `Asia` network. Statistics are calculated from the original probability distribution (E), the Kikuchi approximations calculated from the undirected (U) and moralized (M) graph, and the Kikuchi approximations learned by the learning algorithm (A).

usually employed in statistical physics associated with the system energy. It represents a different domain of application to the Bayesian network example treated before.

Our objective is to define a Boltzmann distribution. We start from the following function $f_d(\mathbf{x})$.

$$f_d(x_1, x_2, x_3) = \begin{cases} 0.9 & for \quad x_1 + x_2 + x_3 = 0 \\ 0.8 & for \quad x_1 + x_2 + x_3 = 1 \\ 0.0 & for \quad x_1 + x_2 + x_3 = 2 \\ 1.0 & for \quad x_1 + x_2 + x_3 = 3 \end{cases} \tag{5.15}$$

Function $f_d(\mathbf{x})$ is used in the definition of function $f(\mathbf{x})$, where $\mathbf{x} = (x_1, \ldots, x_8)$ is a tuple of eight binary variables.

$$f(\mathbf{x}) = f_d(x_1, x_2, x_3) + f_d(x_3, x_4, x_5) + f_d(x_1, x_6, x_7) + f_d(x_5, x_6, x_8) \tag{5.16}$$

Finally, the Boltzmann distribution is defined using $f(\mathbf{x})$,

$$p(\mathbf{x}) = \frac{1}{Z(T)} e^{\frac{-f(\mathbf{x})}{T}} \tag{5.17}$$

where T is the temperature of the system, and $Z(T)$, the corresponding partition function.

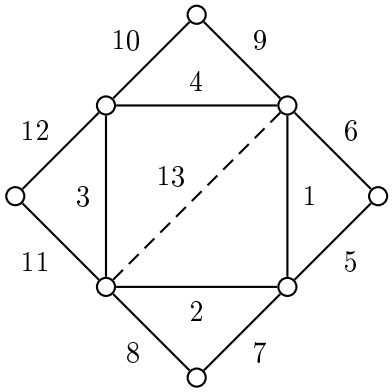


Figure 5.7: Edges of the graph in the order in which the Kikuchi approximation learning algorithm learns them.

We consider $T = 1$. When calculating the Boltzmann distribution, the whole space of solutions has been taken into account. This implies 256 points. The choice of the f_d function for each definition set of $f(\mathbf{x})$ determines the probabilistic dependencies among each subset of variables within the same definition set. The largest connected component in Figure 5.1 shows the independence graph corresponding to the Boltzmann probability model.

Figure 5.7 shows the learned edges numbered according to the order in which Algorithm 5.3 learned them. Figure 5.8 shows the Kullback-Leibler divergence $D(p||k)$ at different steps of the learning algorithm.

After studying both figures, the change in $D(p||k)$ due to the addition of each edge can be seen. The Kikuchi approximation at iteration i is formed by edges $1, \ldots, i$. The Kikuchi approximation at iteration 12 is not a probability distribution. In fact, it corresponds to the Kikuchi approximation shown in Equation (5.12), but excluding factor $p(x_9)$ from the equation. At iteration 13, the algorithm adds an edge that is not in the original graph. The addition of this edge produces a triangulated graph. Hence, the approximation is exact with $D(p||k) = 0$.

This case is an example where the Kikuchi approximation can approximate the Boltzmann distribution well. In this case, the Kikuchi approximation is the second best one after the exact probability distribution obtained from the triangulated graph. The example also illustrates that the learning algorithm is able to find this approximation.

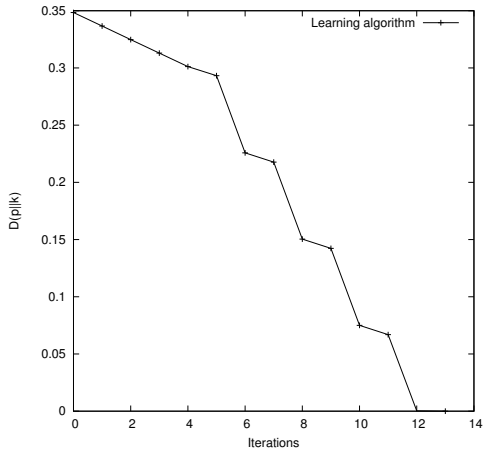


Figure 5.8: Kullback-Leibler divergence $D(p||k)$ at different iterations of the algorithm that learns the Kikuchi approximation of a Boltzmann distribution.

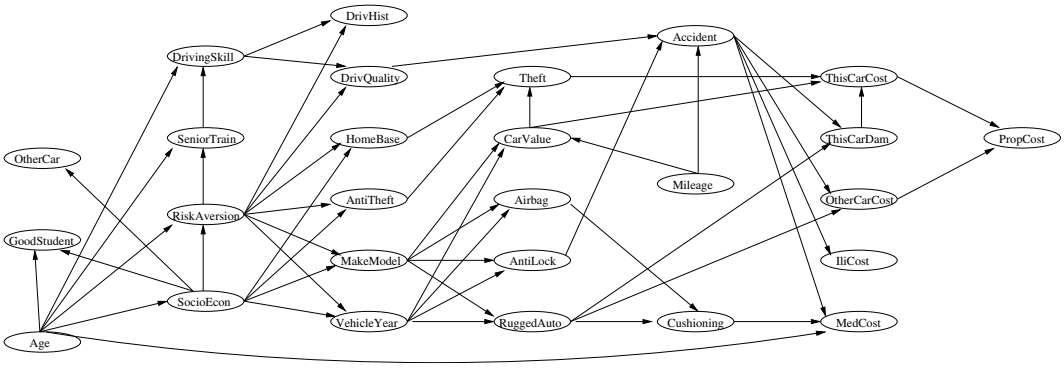


Figure 5.9: `Insurance` network structure.

5.6.3 Results in the approximation of empirical distributions in large datasets

We present experiments with the `Insurance` network (26). The purpose of this network is to evaluate the expected cost of insuring an automobile driver. There are 12 observable nodes, 12 hidden nodes, and 3 query nodes. The observable nodes correspond to questions that a typical insurance agent would ask any prospective client. The hidden nodes correspond to variables that cannot be observed, but can be reasonably inferred. The query nodes represent the expected cost of insuring a driver, divided into property cost, medical costs, and legal costs. The network, which is shown in Figure 5.9, has 52 arcs.

The goal of our experiment is to evaluate the performance of the learning algorithm when the size of the dataset is increased. A dataset of $10,000$ cases, generated from the `Insurance` network using PLS, is employed. The empirical probability distribution calculated from the dataset is denoted $\hat{p}(\mathbf{x})$.

We apply the learning algorithm to appproximate $\hat{p}(\mathbf{x})$. We require that the difference between two successive values of the KLD be less than 0.005. This way, we try to avoid the phenomenon of overfitting. Figure 5.10 shows the KLD at different iterations of the algorithm. The stop condition is satisfied after 92 iterations. It can be seen that the KLD rapidly decreases with iterations. The best KLD is 1.4591, which is considerably smaller than the initial value, 12.3315, obtained for the complete disconnected network.

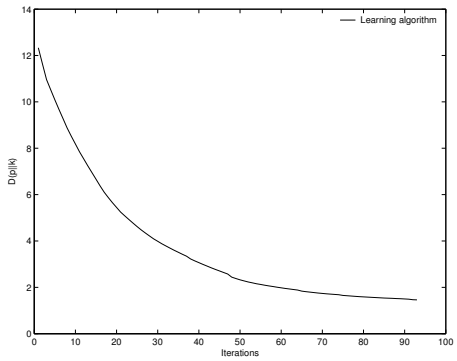


Figure 5.10: Kullback-Leibler divergence $D(\hat{p}||k)$ at different steps of the algorithm for learning the Kikuchi approximation of the `Insurance` network.

We investigate the type of Kikuchi approximations that the algorithm learns at each step. The first Kikuchi approximation that is not a probability distribution is achieved at iteration 74. This means that good approximations of the probability can be achieved in the space of the probability distributions. One reason that could explain this behavior

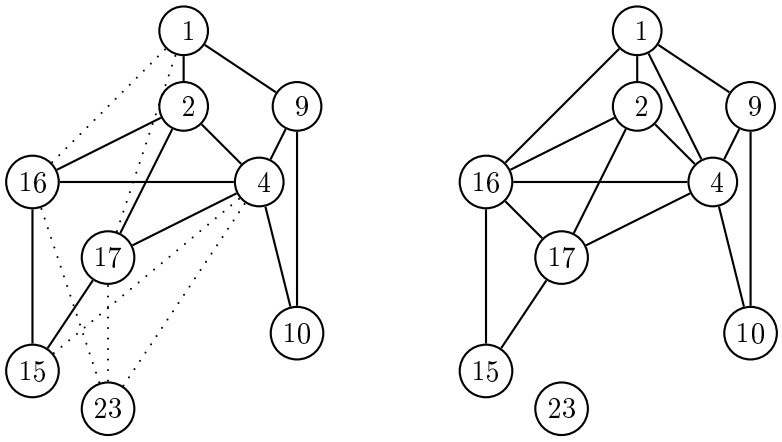


Figure 5.11: Subgraphs from the `Insurance` network. a) Learned by the algorithm; b) Moralized subgraph.

for the `Insurance` network is that there are few loopy regions in the graphs obtained at each generation. The first size 3 clique is formed at iteration 32.

In order to thoroughly analyze the implications of reaching, at step 74, a Kikuchi approximation that is not a probability distribution, we extract one relevant subgraph from the current graph kept by the algorithm at this iteration. This subgraph, which is shown in Figure 5.11 a), contains all the edges to which node X_4 is connected. Additionally, other nodes not connected to X_4 are shown for reasons explained below. Dashed edges correspond to edges that are not in the original moralized subgraph of the `Insurance` network. Therefore, the algorithm was able to learn from the data all the remaining edges that were in the original moralized subgraph.

Node X_4 is the one that determines that the Kikuchi approximation at iteration 74 is not a probability distribution. Equation (5.18) shows part of the Kikuchi approximation obtained at iteration 74. As this is only a part of the Kikuchi approximation, we have called it $k^*(\mathbf{x}_s)$. Notice that, to obtain this valid Kikuchi approximation, it was necessary to multiply by factor $p(x_4)$. X_4 is the only clique in the clique-based decomposition obtained as an overlapping of overlappings. The inclusion of $p(x_4)$ is the only way to guarantee that the overcounting number associated with X_4 is 1.

$$k^*(\mathbf{x}_s) = \frac{p(x_2,x_4,x_{16})p(x_2,x_4,x_{17})p(x_4,x_9,x_{10})p(x_4,x_{15},x_{16})p(x_4,x_{15},x_{17})}{p(x_2,x_4)^1 p(x_4,x_{15})^1 p(x_4,x_{16})^2 p(x_4,x_{17})^2} \frac{p(x_4,x_{16},x_{23})p(x_4,x_{17},x_{23})}{p(x_4,x_{23})^1} p(x_4) \tag{5.18}$$

In Figure 5.11, we have included some nodes that are not connected to X_4 in order to compare the graph learned by the algorithm and the original moralized graph. The moralized graph is shown in Figure 5.11 b). Notice that the size of the maximum clique in the moralized graph is 4. The maximum clique of the graph obtained after the triangulation of the `Insurance` network is expected to be greater than 4. However, the Kikuchi approximation obtained after 92 iterations has a size 3 maximum clique. One of the main advantages of using the Kikuchi approximation is that it can avoid the computational complexity associated with the increase in the dimension of the cliques after moralization and triangulization.

5.6.4 Classification experiments

For the classification experiments, we use the `Zoo` and `King and Rook against Black King (KRK)` databases[1]. The 10 cross validation scheme (212) is used to validate the models. The Kikuchi approximation is learned using 9 of the 10 data slots and used to classify the other one. The process is repeated another 9 more times each time a different slot for classification. Accuracy is the ratio between the number of correct classified cases and the total number of cases. When a case is classified in more than one class, the score is divided among all these classes.

Description of the databases

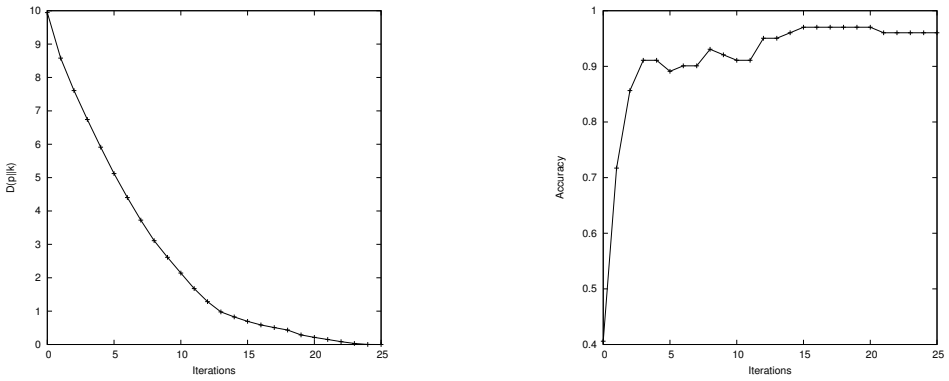
The `Zoo` database describes 101 animals using 15 Boolean attribute variables (hair, feathers, milk, etc...) and one numerical variable (number of legs). The class variable has 7 possible values that describe the classes where animals can be grouped.

The `KRK` database (12) is a chess endgame database. In this game, black cannot win and is unable to draw at dephs greater than zero. Each case stores information about the current positions (row and column) of three chess pieces (white king, white rook, and black king). These are the 6 attribute variables. The class variable is the number of moves (between 0 and 16) needed for white to win or draw (draw is represented with an additional value "draw"). Minimax optimal play is assumed to calculate the number of moves from the board configuration. This sort of database has been used in the Inductive Logic Programing framework (155). These systems are provided no only with the coordinates of the pieces but also with background knowledge about the problem in the form of row and column differences between the pieces. The database is automatically learned, and symmetry in the space of positions of an endgame is exploited in an attempt to minimize storage size and generation complexity for that endgame (12).

[1]These databases are available at the UCI repository database http://www.ics.uci.edu/~mlearn/MLSummary.html

Results for the Zoo database

Figure 5.12 shows the KLD and the classification accuracy of the Kikuchi approximation learned at each iteration of the learning algorithm. As the KLD is minimized, the accuracy of the classifier tends to increase. At iteration 15, accuracy stops improving but the KLD keeps decreasing on. At generation 20, accuracy slightly deteriorates. From this example, we corroborate that accuracy does not necessary improve with the KLD. Another interesting result is that the first 13 learned edges are between the class variable and the rest of the 16 atribute variables.



a) b)

Figure 5.12: Zoo database: a) KLD and b) accuracy at different steps of the algorithm for learning the Kikuchi approximation.

Results for the KRK database

The KRK database is very difficult to learn. It has 18 classes and 28056 cases. Additionally, the cases are irregularly distributed among the classes, with 3 classes with fewer than 100 cases each and 9 classes with more than 1500 cases each. We define 3 different classification problems from this database. In all the problems, the task is to predict the number of moves to win. The first problem considers only 5 values for the class variable (they represent configurations that lead to win in $0, \ldots, 4$ steps). Similarly, problems 2 and 3 consider 9 and 18 values for the class variable. The number of cases for each problem is shown in Table 5.2.

Table 5.2 also shows the iteration where the Kikuchi approximation with the best accuracy has been reached. The accuracy and the KLD learned at this iteration are also

Init	Problem 1			Problem 2			Problem 3		
N. classes	5			9			18		
N. instances	630			5521			28056		
	Iter.	*KLD*	*Acc.*	*Iter.*	*KLD*	*Acc.*	*Iter.*	*KLD*	*Acc.*
Kikuchi	8	1.1983	0.59	10	1.2675	0.56	12	0.3809	0.44
	17	0.0002	0.25	17	0.0784	0.25	16	0.0064	0.41

Table 5.2: Results of the algorithm for learning the Kikuchi approximation in the KRK database.

shown. These results are presented in the row corresponding to the Kikuchi approximation. In the next row, we present the KLD achieved at the last iteration of the algorithm, and its corresponding accuracy.

The decrement in the KLD is mainly due to interactions between the attribute variables. Therefore, there is no strong relationship between the KLD minimization and the increase in accuracy. The increase in the model's complexity during learning points out the need to use a complexity measure to avoid too complex models. This is particularly important when the cardinality of variables is very high. In these cases, a link in the graphical model accounts for an important increase in the number of parameters that need to be stored.

However, as mentioned before, KRK is a very particular database, where the relationship between attribute and class variables is very intricate and where the incorporation of additional information about the problem knowledge seems to be critical for accurate prediction results.

5.7 Conclusions

In this chapter, we have introduced an algorithm to learn Kikuchi approximations from data. In comparison with two previous approaches that exist for this problem, the proposal introduced has a number of advantages:

- In contrast to the Kikuchi-Bayes method used for classification tasks, the Kikuchi learning algorithm takes advantage of the theoretical properties satisfied by the clique-based decomposition to use the Kullback-Leibler divergence as a scoring metric that can be efficiently optimized by locally updating the current approximation.
- In comparison with the learning method based on independence tests, the Kikuchi learning method does not need the use of heuristic methods to make the graph

sparser in order to avoid an increase in the dimension of the cliques beyond a feasible limit. Additionally, local and global search methods can be used to find appropriate Kikuchi approximations.

The metric chosen in the chapter is the Kullback-Leibler divergence. Nevertheless, the properties fulfilled by the Kikuchi approximation can prove decomposability properties for other measures, like the equivalent of the likelihood. Potentials applications of the Kikuchi learning algorithm and research topics that are worth future attention include the following:

1. Other graphical representations (e.g. factor graphs) can be used to represent the Kikuchi approximation and to define more general learning algorithms.
2. The learning algorithm introduced is a natural candidate to learn mixtures of Kikuchi approximations by means of the expectation maximization (EM) algorithm.

6 Region-based decompositions, propagation and abductive inference in optimization

6.1 Introduction

Recently, certain developments in probabilistic graphical models have extended the class of models that can be used to model probability distributions, and have introduced new algorithms for inference based on these models. In this chapter, we show how these results can be used to improve the use of graphical models in optimization. We introduce a number of proposals that expand the type of models and algorithms based on graphical models that can be used for optimization.

The analysis presented in this chapter focuses on a number of research trends that we consider relevant for the work in optimization. They are summarized in the following list:

- Development of inference algorithms in graphs with cycles.
- Use of region-based approximations.
- Study of the properties of propagation algorithms for different graph topologies.
- Design of methods for abductive inference.

In Section 3.2.3, we have reviewed a number of early applications of the research trends listed above to optimization. In Chapters 4 and 5, we presented results on the use of region-based decompositions to approximate probability distributions. Now, we present a number of other ways in which optimization algorithms can use these results. The chapter is organized as follows. In the next section, a framework for the characterization of EDAs is introduced. Different EDAs are grouped according to the way the probabilistic model is learned and sampled. In Section 6.3, we study learning and sampling in region-based approximations and how these methods can be inserted and combined with other approaches in EDAs. Finally, Section 6.4 presents the conclusions of our work and some lines for further research.

6.2 Generalized EDAs based on undirected graphs

Algorithm 6.1: Generalized EDA

1 Set $t \Leftarrow 0$. Generate M points randomly.
2 **do** {
3 Select a set S of $N \leq M$ points according to a selection method.
4 Learn an undirected-graph-based representation of the dependencies in S.
5 Using the graph, determine a class of graphical model or approximation strategy to approximate the distribution of points in S.
6 Determine an inference algorithm to be applied in the graphical model.
7 Generate M new points from the model using the inference method.
8 $t \Leftarrow t + 1$
9 } **until** Termination criteria are met.

We start with the introduction of a generalized EDA that analyzes, from a unified perspective, several of the developments of EDAs. We constrain our analysis to EDAs based on undirected graphical models. The pseudocode of the generalized EDA, shown in Algorithm 6.1, also serves as a framework for the description of the algorithms introduced in this chapter. Based on this generalized framework, we compare the proposals introduced with previous works.

There are a number of specific features of the generalized EDA. One important characteristic is that it allows the use of different classes of graphical models at each generation. This is motivated by the fact that the distributions of points in the selected set could be very different at each stage of the evolution. These differences can, in some cases, be captured by probability approximations arising from the same class of graphical models. However, in many cases, one class of graphical models is more suitable than others to approximate the distribution at one generation, while the opposite situation can happen at another generation. This fact can be seen by analyzing the structure of the probabilistic models learned at different generations, as has been done in (19).

The dynamic change of the probabilistic model would require an automatic procedure to select among the different types of graphical models. The topological characteristics of the undirected graphs learned are plausible information for this decision. The number, size, and cardinalities of the variables of each clique are three of the issues that determine the feasibility of the model for estimating the marginal probabilities and sampling new solutions. Once the model has been chosen, different types of sampling algorithms can be employed.

We analyze the main components of Algorithm 6.1 in detail.

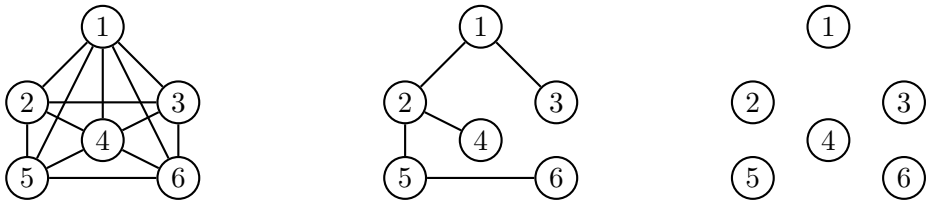


Figure 6.1: a) Cyclic graph; b) Tree shaped graph; c) Graph without edges.

Example 6.1. Figure 6.1 describes an example of the different types of graphs that can be learned by EDAs. This corresponds to a problem with six variables. At initial generations, many dependencies between the variables are usually detected. This case may be represented by the graph shown in Figure 6.1 a). As evolution advances and diversity of the population is lost, fewer dependencies may be detected and the graph can be chordal or easily triangulated. This is the case represented in Figure 6.1 b), where a tree shaped model is presented. Eventually, the population will converge to a single solution and diversity will be lost. This case is shown in Figure 6.1 c), where the absence of edges between vertices mean that variables are independent.

6.2.1 Learning of the undirected graph

Although the scheme we have proposed to analyze the different alternatives of learning and sampling in EDAs is based on undirected graphical models, this scheme could be generalized to cover algorithms that employ directed graphical models.

We will assume that the undirected graph encodes the dependence/independence relationships between the variables. The graph can be learned from data by means of score+search methods or using independence tests. When independence tests are used, it is important to take into account that the final graph may have cycles, which is not the case in traditional applications of score+search methods.

In (190), an algorithm that constructs the independence graph by means of the application of Chi-square independence tests has been used to learn Kikuchi approximations. In this thesis, the algorithm used to learn the undirected graph employs the mutual information between variables. The pseudocode of the algorithm is shown in Algorithm 6.2. The algorithm receives as an input parameter a threshold value γ. This value is used to select those edges that will be removed from the graph. γ is used to measure the strength of the interactions measured by the mutual information values. When variables have different numbers of values, the mutual information values should be normalized before the selection of the edges.

We consider two variants of the algorithm. The first variant considers cases in which problem information about the interactions between variables is available. In other words, instead of calculating the mutual information for all pairs of variables, it only does so when there is evidence of interaction between variables. This way, fewer parameters must be learned.

The second variant resembles the method used when learning the maximum weight spanning tree from a matrix of mutual information (49). Edges are selected according to their weight until all the nodes of the graph that have at least one edge have been added. The difference from the method for learning the tree structure is that cycles are allowed in the latter.

Algorithm 6.2: Algorithm for learning the independence graph

1. Compute the univariate and bivariate marginal frequencies $p(x_i)$ and $p(x_i, x_j)$ of S.
2. Calculate the mutual information value $MI_{i,j} = \sum_{x_i,x_j} \frac{p(x_i,x_j)}{p(x_i)p(x_j)}$ for each pair of variables.
3. The undirected graph will be formed by those edges whose corresponding mutual information is above a given threshold γ.

6.2.2 Determining the class of approximation strategies and graphical models

For this analysis, we will assume that the undirected graph is given and not reducible (i.e. the existence of an edge implies that the pair of variables is not independent given the rest of the variables). This assumption might be violated by the algorithm that learns the undirected graph, but it is convenient for the analysis that follows.

The determination of the class of graphical models to be employed depends on a number of factors. Since modeling is usually based on a factorization of the probability distribution in marginal and conditional probability factors, the size of the factors play an important role in the feasibility of the approximation. One important condition for these factorizations to be feasible is to constrain the size of the marginal tables. This size is the product of the cardinality of each variable in the table minus one. Another issue related to the complexity of the approximation is the number of cliques which it comprises.

Therefore, one way to choose among the class of possible approximations is to constrain the size of the largest marginal table as well as the number of factors.

The first step is to calculate all the maximal cliques of the graph and determine the size of the probability tables. To simplify our analysis, we will assume that all the variables have

Graphs	Graphical model	Inference	Approximation
exact graph	univariate	PLS	exact
	univariate	MPC	exact
	junction tree	PLS	exact
	junction tree	MPC-BP	exact
	junction graph	PLS	approx.
	junction graph	MPC-BP	approx.
	clique-based Kikuchi approx.	GS	approx.
	Bethe approx.	MPC-loopy BP	approx.
	Kikuchi approx.	MPC-generalized BP	approx.
subgraph	univariate	PLS	approx.
	univariate	MPC	approx.
	junction tree	PLS	approx.
	junction tree	MPC-BP	approx.
	junction graph	PLS	approx.
	junction graph	MPC-BP	approx.
	clique-based Kikuchi approx.	GS	approx.
	Bethe approx.	MPC-loopy BP	approx.
	Kikuchi approx.	MPC-generalized BP	approx.
triangulated graph	junction tree	PLS	exact
	junction tree	MPC-BP	exact

Table 6.1: Approximation strategies, graphical models, and inference methods to be employed by EDAs based on undirected graphs.

the same cardinality and, therefore, the largest table will correspond to the maximum clique of the graph. The analysis can be generalized to the case where variables have different cardinalities. If the graph is triangulated, and the maximum clique of the graph fulfills the complexity constraint, any of the alternatives listed in Table 6.1 as *exact graph* could be applied. Notice that univariate factorizations are a particular case of junction trees, which, on the other hand, are a particular case of junction graphs.

If the graph is not triangulated, then one possibility is to triangulate, calculate the maximum cliques of the graph, evaluate whether the complexity constraint is fulfilled for the triangulated graph, and in that case, apply any of the alternatives listed in Table 6.1 as *triangulated graph*. Another possibility that avoids the use of triangulation is to use a junction graph approximation (189), or a Kikuchi approximation of the probability that uses a clique-based decomposition of the graph (see Chapter 4).

As analyzed in Chapter 4, region-based approximations are used in statistical physics to calculate the free energy of the system. Since the use of region-based decompositions and the design of suitable inference algorithms for their application is one the main goals of this chapter, we analyze these topics in Section 6.3.

If the complexity constraint is not fulfilled in the original or in the triangulated graph, then other types of approximation must be tried. One possibility is to simplify the graph by spliting the largest cliques, something that can be done by removing edges. Another possibility is to make the graph sparser in one step previous to the calculation of the cliques. In (189; 190), this has been done by allowing a maximum number $r-1$ of incident edges to each vertex. If one vertex has more than $r-1$ incident edges, those with the lowest weights (if available) are removed. This way, the size of the maximum clique will always be smaller than or equal to r. The refinement algorithm avoids introducing a bias in the way the edges are removed. However, it has a main drawback: there could be more than $r-1$ variables depending on a single one, but the maximum clique where this variable is included could be smaller than r. In this case, the procedure that eliminates the edges would remove dependencies from the graph without a real need to do so. An alternative to avoid this situation is to use score+search methods.

Maximal cliques of the graph

To find all the cliques of a graph, the Bron and Kebosch algorithm (38) is used. This algorithm uses a branch and bound technique to cut off branches that can lead to cliques. Once all the cliques have been found, they are stored in a list L, and their weights are calculated from the information about the strength of interactions. When no information is available, all the edges have identical weight. The weight of any subgraph G' of G is calculated as $W(G') = \sum_{i \sim j \in G'} w(i,j)$.

6.2.3 Determination of the inference algorithm

There is one characteristic feature about the use of probabilistic graphical models in optimization. The ultimate goal of learning the graphical model in optimization, particularly in EDAs, is to sample new solutions. In most other applications of graphical models, the final goal is not sampling but doing inference in the presence of evidence, or estimating the probability associated with a given configuration. This difference must be taken into account in the analysis that follows. We will use a number of results that can be conceived either to extend the class of models where inference is possible, or to increase the efficiency of the inference algorithms. Therefore, the translation of these results to a domain where the main objective is sampling is not always straightforward.

The most common method applied for inference in the context of EDAs is PLS. It starts from an order of the variables imposed by the structure of the graphical model. Each variable is sampled given the values assigned to its ascendants in the order. PLS can be applied to the junction tree and junction graph, but it cannot be applied to any other approximation listed in Table 6.1 because, in the general case it is not possible to find an order of the variables for these approximations.

For Kikuchi approximations that use clique-based decompositions, Gibbs sampling can be employed. In this case, the conditional probability distributions serve to determine the transitions in the Markov chain. The drawback of using Gibbs sampling is that if the most probable configuration has an exponentially small probability a large number of configurations will need to be visited to hit the optimum. Sampling methods used with region-based approximations are analyzed in the next section.

6.3 Learning and sampling from region-based decompositions

In Chapter 4, we have considered the use of region-based decompositions to approximate probability distributions that are sampled later using Gibbs sampling. In this chapter, we focus on the application of these approximations for abductive inference within EDAs.

A first step in this direction is the work presented in (95). In this work, the definition set is a subvector of the variables on which the ADF is defined, and the undirected graph is the Markov network. A region-based decomposition of the graph is defined. Beliefs are calculated using the subfunctions defined in each definition set. Belief propagation is used to find a consistent probabilistic model. Using the beliefs obtained after the belief propagation has converged, sampling takes place. As, in the general case, it is not possible to find an order of the variables from the region-based decomposition, a model building operator (called subfunction join or merge) is introduced before to do

the sampling. This method adds edges but does not necessarilly fulfill the running intersection property. Three aspects are worth being outlined:

1. The problem structure is given a priori.
2. GBP with marginalization is used.
3. Sampling is done using a subgraph from the initial graph.

Another development was presented in (96). This approach combines the use of Kikuchi approximations with the application of GBP using maximization instead of marginalization. It is shown that the combination of GBP with an algorithm to find the most probable configurations of a loopy graphical model is an efficient solution to the Ising model, which is the benchmark problem used in the experiments.

When the message passing algorithm has converged, this means that all regions are consistent. In this situation, the maximum can be constructed by setting the variables of each region at the values with maximal belief. Yet sometimes the algorithm does not converge. In these cases, the regions are not consistent, and the beliefs could be contradictory. To solve this task, an ordering of the maximal regions (those without parents) is defined in (96). The order is used to estimate the maximum by setting, in each region, the variables at the values with maximal belief, given the values chosen in the previous step. This procedure is similar to the application of PLS for a junction graph (189).

The main characteristics of this method are the following:

1. The problem structure is given a priori.
2. GBP with maximization is used.
3. The algorithm used for finding the most probable configurations is the one presented in (164).

Additionally, experimental results presented in (96) are only for binary problems. We introduce a new proposal to learning and sampling region-based decompositions from data. Therefore, the general scheme of our proposal is presented in Algorithm 6.3.

The rationale of Algorithm 6.3 is to allow learning from data. In (95), the structure of the function and the subfunctions of an additive function are given. However, in many real black box optimization problems, this information is not available. Another difference is the type of method used to find the most probable configurations.

In general, methods used to find the most probable configurations combine the application of max-propagation with a particular partition of the solution space. The efficiency

Algorithm 6.3: Learning and sampling region-based decompositions

1. Learn the undirected graph from the data.
2. Determine the region graph decomposition to be employed.
3. Find the marginal probabilities associated with each region.
4. Using the marginal probabilities, calculate the local potential in each region.
5. If necessary, apply a propagation scheme to find consistent marginal probability distributions.
6. If convergence is achieved, sample using an algorithm for finding the most probable configurations. Otherwise, organize the sampling from the inconsistent marginals.

of the algorithm critically depends on the quality of the partition of the space used. Nilsson's algorithm (164), employed in (95) to find the most probable configurations, needs $O(kn)$ calculations of the max marginals (propagations), where k is the number of configurations to be found. In this dissertation, we propose the use of the most probable configuration algorithm introduced in (242), which only requires $O(2n)$ calculations of the max marginals. The main steps of this algorithm can be shown in Algorithm 6.4. We have respected the notation used in (242).

Algorithm 6.4: Best max-marginal first (BMMF)

1. $SCORE_1(i,j) = max_{x:x(i)=j} Pr(X = x|y)$
2. $x_1(i) = argmax_j SCORE_1(i,j)$
3. $CONSTRAINTS_1 = \emptyset$
4. $USED_2 = \emptyset$
5. **for** $t = 2$ **to** T
6. $\quad SEARCH_t = (i,j,s < t : x_s(i) \neq j, (i,j,s) \notin USED_t)$
7. $\quad (i_t, j_t, s_t) = argmax_{i,j,s \in SEARCH_t} SCORE_s(i,j)$
8. $\quad CONSTRAINTS_t = CONSTRAINTS_{s_t} \bigcup \{(x(i_t) = j_t)\}$
9. $\quad SCORE_t(i,j) = max_{x:x(i)=j, CONSTRAINTS_t} Pr(X = x|y)$
10. $\quad x_t(i) = argmax_j SCORE_t(i,j)$
11. $\quad USED_{t+1} = USED_t \bigcup \{(i_t, j_t, s_t)\}$
12. $\quad CONSTRAINTSs_t = CONSTRAINTS_{s_t} \bigcup \{(x(i_t) \neq j_t)\}$
13. $\quad SCORE_{s_t}(i,j) = max_{x:x(i)=j, CONSTRAINTS_{s_t}} Pr(X = x|y)$

6.3.1 Other extensions

Propagation also allows information to be integrated from different sources. Belief available from different evolving populations (e.g. island models) can be passed into the main graphical model as evidence. The search of a consistent model would be an alternative way to combine information from different sources.

Another possibility is the use of partial information about the function. In cases where the structure of the interactions is known only for some of the variables, the graphical model can combine the information available with that learned from the data. Propagation of evidence in the resulting model would lead to the generation of solutions that incorporate the available information about the problem.

6.4 Conclusions

The main contributions of the work presented in this chapter are the following:

- The application of the propagation has been extended to discrete variables.
- The structure of the problem is learned from data.
- The approach allows the combination of information from different sources.
- A more efficient algorithm to calculate of the most probable configurations has been introduced to EDAs.
- The region-based decompositions are not learned a priori. They depend on the graph structure.

The use of region-based decompositions is another important element of recent development on the use of probabilistic models for optimization. In probabilistic inference, GBP that uses region-based decompositions allows the free energy to be computed in cases where loopy belief propagation algorithms do not converge.

There are a number of research trends where the work described in this chapter could be applied. Most existing applications of region-based decompositions are constrained to pair-wise Markov fields. Therefore, applications in optimization, where complex interactions between sets of three or more variables can arise, are an interesting test field for propagation methods.

Most probable configurations can be used to evaluate the convenience of using different probability models in optimization. We can exactly calculate which are the most probable configurations for a given model. Given the k most probable configurations of a model,

it is possible to use the average fitness of the configurations as a measure to evaluate the "efficiency" of the model to generate good solutions. This measure would enable the evaluation of models learned from the same set of original solutions according to their capacity to exploit the information learned. For instance, in (193), the role of malign and benign dependencies is studied in the framework of EDAs. Most probable configurations can further evaluate whether models that comprise only malign interations are significantly different from those that only employ benign interactions.

Part III

Protein problems

7 Computational protein problems

7.1 Introduction

In this chapter, we present the context of application of the results introduced in the second part of the thesis. Optimization techniques based on graphical models will be applied to protein problems. Therefore, we briefly analyze current research in the field of computational biology, focusing on the application of machine learning techniques to protein problems.

The chapter is organized as follows. In the next section the field of computational biology is introduced, and some of its main research trends are described. Section 7.3 presents the biological foundations of proteins. In Section 7.4, we discuss a number of issues related to protein modeling. Several computational approaches to protein problems are reviewed in Section 7.5.

7.2 Computational biology

Computational biology comprises the development and application of data-analytical and theoretical methods, mathematical modeling, and computational simulation techniques to the study of biological, behavioral, and social systems (50).

For practical purposes, the computational techniques applied to the biological domain have been further divided into subdomains that include neuroinformatics, social models and genetics. Similarly, the set of methods that can be included in computational biology is very broad. They cover well-established data collection and validation procedures, statistical tests to analyze these data, and machine learning techniques to model and simulate biological processes and extract relevant information from these data. This chapter will concentrate on the study of machine learning approaches to biological problems.

In (123), biological problems where machine learning techniques can be applied are divided into seven main categories: proteomics, genomics, microarrays, systems biology, evolution, text mining, and other applications. This classification, which is relevant to our analysis, is described in Figure 7.1, which reflects the overlapping between different categories.

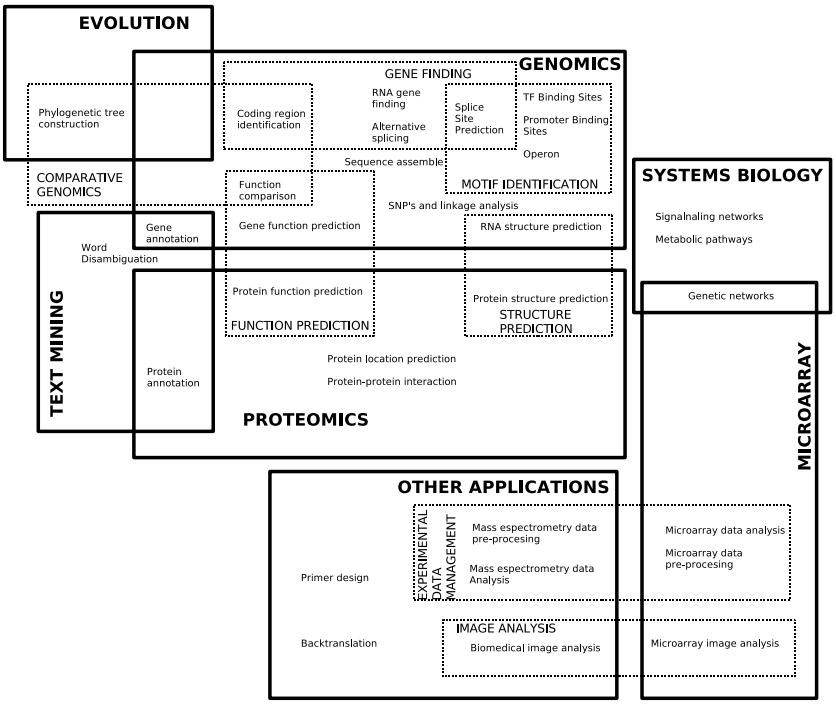


Figure 7.1: Classification of the topics where machine learning methods are applied.

Genomics and proteomics are understood as the study of nucleotide chains and proteins, respectively. Microarrays refers to the application of computational methods to the management of complex experimental data derived from experiments, mainly from microarray essays. Modeling biology includes the modeling of life processes that take place inside the cell and of others of great interest, like biological networks (34). Evolution is understood as the application of machine learning techniques to the study of evolutionary processes, particularly phylogenetic tree reconstruction. Text mining techniques cope with the problem of extracting relevant information from the increasing amount of data resulting from the growth in available publications. Other applications is an umbrella term that covers the rest of applications like primer design for polymerase chain reaction, biological image analysis, etc. These categories should be understood in a very general way.

Machine learning applications can be roughly divided into two classes. Machine learning applications to modeling and machine learning applications to optimization.

In a modeling problem, learning pursues the construction of a simplified representation of the process that is being modeled after. Since one of the goals of computational modeling is to make inference from the model, machine learning can use statistical theory to build such models. The two main steps in this process are (123), first, to induce the model, processing the huge amount of data, and second, to represent the model and make inference efficiently.

The optimization problem has been formally posed in Section 3.1. In the next sections, some of the particularities of the optimization and learning problems in the bioinformatics domain are analyzed.

7.2.1 Modeling in computational biology

Modeling is an essential step to the understand the very complex structures and dynamics of biological systems. A number of factors must be considered:

- Definition of the specific features of phenomena that will be modeled.
- Capacity of representation of the modeling framework used.
- Data available for the model validation.
- Definition of the validation procedure and determination of the statistical tests to be employed.

In this thesis, we have applied graphical models to model some dynamics that arise during the protein folding process, and that depend on structural features of the proteins. The factors listed above were taken into account for our analysis. This topic will be analyzed in detail in the next chapter.

7.2.2 Optimization in computational biology

Many biological problems can be defined as the task of finding an optimal solution in a space of multiple (sometimes exponentially sized) possible solutions. Usually, the definition of the problem as an optimization one is not straightforward. Three questions must be taken into account in these situations:

- Definition of the solution representation: The same problem can be approched using different representations. This choice can notably influence the search efficiency.
- Definition of the objective function: The objective function should express the available information about the problem. In the biological arena, where domain information can be scarce or imprecise, this issue deserves particular attention.
- Choice of the optimization method to be used: There are many optimization algorithms available. The convenience and efficiency of the selected method is crucial for a successful implementation.

For the problems treated in the thesis, we have previously analyzed these factors. The choice of the optimization method has been constrained to the class of EDAs. The power of expression as well as the storage and efficiency concerns, associated with the learning and using the probabilistic model, have determined the type of EDA selected to address each problem.

7.3 Protein definitions

Proteins are essential components of living organisms. They are formed by a set of amino acids or residues which, under suitable conditions, fold to form a functional structure. Amino acids are combined to form sequences which are considered the primary structure of the peptides or proteins. The secondary structure is the locally ordered structure caused by hydrogen bounding mainly within the peptide backbone. The most common secondary structure elements in proteins are the alpha helix and the beta sheet. The tertiary structure is the global folding of a single polypeptide chain.

There are twenty different amino acids. Each 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 an atom of hydrogen, and a side chain group R. A peptide bond is formed by the dehydration of the carboxyl group of one amino acid and the amino group of the next.

Figure 7.2 shows the structure of the amino acid called alanine. The amino and carboxyl groups are in the upper and lower parts of the molecule, respectively. At the center,

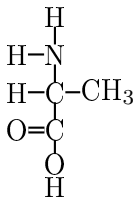


Figure 7.2: Structure of alanine.

connected to the alpha carbon, are the hydrogen atom on the left, and the side chain group CH_3 on the right.

The dihedral angles between the amino group and the alpha carbon and carboxyl group are free to rotate. These angles are respectively referred to 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. Folds can be described as the architecture of a protein. Two proteins will have a common fold if they have comparable elements of secondary structure with the same topology of connections.

Figure 7.3 shows[1], from left to right, the complete native structure of the $pdb1mrj$ protein[2], only the backbone of the protein, and only the side chains.

7.3.1 Protein folding

Under specific conditions, the protein sequence folds into a unique native 3-d structure. Each possible protein fold has a value for its associated free energy. The *thermodynamic hypothesis* states that the native structure of a protein is the one for which the free energy achieves the global minimum. Another important fact that illustrates the complexity of protein folding is known as the *Levinthal paradox*: the size of the conformational space of a protein makes finding the single native state by combinatorial search impossible. The sequence must not only specify a native state, but, at the same time, a pathway to arrive there (210).

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

[2] In this thesis, all the proteins used in our research are refered to using their protein data bank identifier (PDB ID) (22)

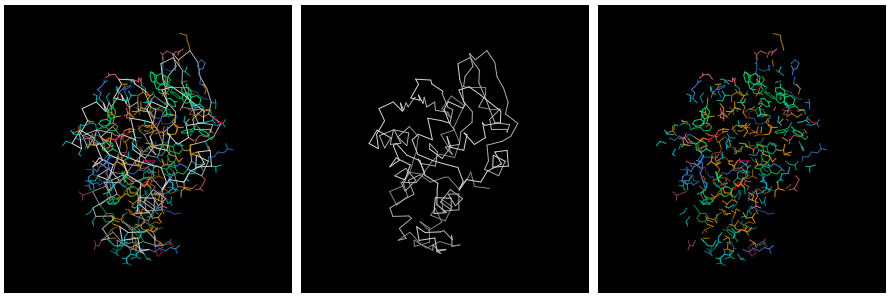


Figure 7.3: From left to right: Native structure of the $pdb1mrj$ protein, backbone of the protein and side chains.

However, the exact laws that govern protein folding are unknown. Therefore, some researchers have attempted to explain how amino acid sequences specify the 3-d structure of proteins.

There are two main approaches to the explanation of the protein folding. They are commonly referred to as the "classical" and "new" views. The "classical" view considers folding as a defined sequence of states leading from the unfolded to the native state. This sequence is called the pathway (174). In the "new" view approach, folding is seen as the progressive organization of an ensemble of partially folded structures through which the protein passes on its way to the folded structure (172). This approach emphasizes the idea of each state being an ensemble of rapidly interconverting conformations. One of the main differences between both approaches is that the "new" view allows for a more heterogeneous transition state than the "classical" view, which concentrates on a single, well-defined folding pathway (13).

Figure 7.4 shows one schematic representation of the "classical" (left) and "new" (right) views of protein folding. In the figure, a circle represents each possible protein configuration, and an arrow represents a possible transition between configurations. In both approaches, the native state (dark circle) is achieved when the energy is minimized. We will go back to the analysis of the "new" view of protein folding in Section 8.6.1.

The role of proteins cannot be seen in isolation. Biological processes depend on the interaction of several proteins that form what are known as *protein networks*. These are highly structured networks that allow proteins to interact together to coordinate their functions.

Beyond the problem of identifying the protein structure from its sequence, is the more challenging question of predicting its function and the way it interacts with other proteins in its network. Machine learning techniques are important tools for function prediction and the discovery of protein networks. So far, in this thesis we concentrate on the use of protein modeling for function structure prediction and protein design.

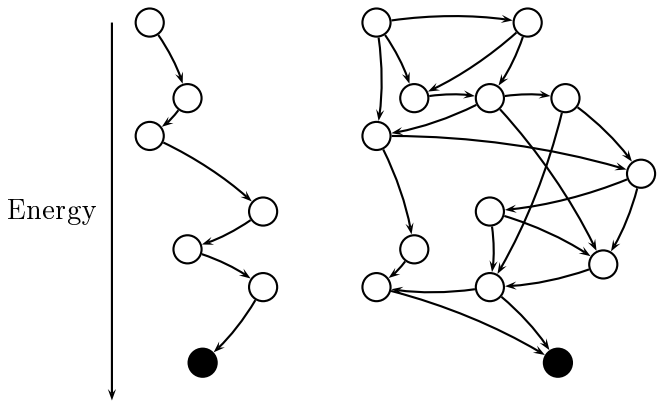


Figure 7.4: Schematic representation of the "classical" (left) and "new" (right) views of protein folding.

7.4 Protein modeling

Inferring the protein tertiary structure from its sequence is an important problem in molecular biology (7). Computational models of proteins are an important component for the solution of the protein structure problem. While, in protein structure prediction, the goal is to predict the native structure of a protein from its sequence, protein design intends to create new proteins that satisfy some given structural or functional constraints.

Both problems are very much related to each other. Protein design challenges the general assumptions used to predict protein structures. Similar energy functions can be used to solve both problems. The protein design problem can be seen as the inverse folding protein.

Almost all existing approaches for protein design and protein structure prediction use, to different extent, homology modeling. Two sequences are homologous if they descend from a common ancestor. Homology is a clear indication of shared structure and frequently related function. Proteins can be grouped into families of homologous sequences with a clear evolutionary relationship. Homologous sequences can share similar folds, although the opposite is not necessarily the case. Superfolds are folds taken up by a large number of non-homologous sequences.

The determination of homologous proteins is therefore a required step of many approaches to protein design. At this step, sequence alignment algorithms are used to determine homologous proteins. Although there are efficient algorithms for pairwise alignment, the determination of a multiple alignment that optimizes a given score is a NP problem (105; 228). Here, heuristic optimization techniques are widely used to obtain near to

optimal solutions.

The case of threading illustrates one of the existing protein design methods. This examples we explain the case of threading. This example introduces the important role which objective functions play in protein design and protein structure prediction. This role will be analyzed in the next section.

The basic concept of threading is to decorate a given backbone structure with the side chains from a sequence to be analyzed and then evaluate this model with a given objective function. The sequence can be moved along the model structure and threaded along alternative structures. If any of the sequence-structure combinations scores significantly better than the average, this particular combination may be a valid structural model (210).

7.4.1 Energy functions

There are many factors that influence the stability of proteins and have to be taken into account to evaluate candidate structures. The native state is thought to be at the global free energy minimum. The free energy has contributions from electrostatic interactions, including hydrogen bonds, van der Waals interactions, intrinsic propensities of the amino acids to take up certain structures, hydrophobic interactions and conformational entropy. Determining to what extent all of these factors can be represented in the function, as well as the weight each one must receive are difficult issues.

Simplified models that ignore some of these factors are a first approximation. For example, the approximate fold of a protein appears to be specified by the sequence of hydrophobic and hydropholic residues, irrespective of what the actual amino acids in that sequence are (210). Therefore, a first approximation might simply be constructed by a binary patterning of hydrophobic and hydrophilic residues to match the periodicity of secondary structural elements.

Simplification can be further developed to consider proteins represented using this binary patterning and to approach the protein structure prediction problem in two and three dimensional lattices. In this case, the energy function measures only hydrophobic and hydrophilic interactions. This simple approach for protein structure prediction is considered in the next chapter.

Based on the thermodynamic hypothesis, many methods that search for the protein native structure define an approximation of the protein energy and use optimization algorithms that look for the protein fold that minimize this energy. These approaches mainly differ in the type of energy approximation employed and in the characteristics of the protein model.

In the general case, most molecular potentials used for protein design and protein structure prediction result from (176):

- Fitting interatomic interactions to data derived from quantum electrical calculations or from spectroscopic experiments. Examples of the force fields obtained with these methods are CHARMM (39) and AMBER (231).
- Fitting the potential to the structural preferences observed among atomically detailed structures in the protein data base (PDB). An example of this type of function is the rotamer library approach used for side chain prediction (42).

One of the problems related to energy definition is that protein structures must be uniquely specified in order to maintain the native state efficiently. Maintaining a precise folded structure implies that no alternative structure is accessible with comparable energy. This fact complicates the design of accurate energy functions.

7.5 Machine learning approaches to the solution of protein problems

Main efforts in computational studies of proteins have been put in the problems of inferring the protein secondary and tertiary structure from the sequence, and in protein design.

7.5.1 Applications in protein structure prediction

There are a variety of approaches that differ in the complexity of the protein models used, or in the particular domain of application (silico protein design, evolutionary biology, etc.).

Several optimization approaches have been used for protein folding in simplified models. These include tabu search (28; 131), Monte Carlo methods (17; 75; 83; 98; 99; 134), GAs (52; 113; 117; 118; 121; 181; 206; 221), immune algorithms (58; 59), ACO (203; 204), and EDAs (192).

Protein side-chain prediction has been approached using dead-end elimination algorithms (64; 65; 136), GAs (135; 220; 239) and other population-based search methods (78). Simulated annealing (129), optimization methods based on inference from graphical models (240; 241), and the self consistent mean field approach (112) have also been employed to solve this problem. Protein folding with complex models has been also approached using Monte-Carlo optimization (188).

Some of the approaches mentioned above will be analyzed in detail, along with our proposals for the solution of these problems, in Chapters 8, 9 and 10.

Although the main attention of protein research has mainly focused on protein structure prediction and protein design, other domains of applications include the recognition of protein receptor sites (103), the prediction of small molecule binding modes to macromolecules of known three-dimensional structures (104), and the identification of membrane protein solely from its sequence analysis (254). Contact map prediction in proteins (139) and automatic annotation of protein function based on family identification (1) have been treated too.

8 Protein folding in simplified models with estimation of distribution algorithms

8.1 Introduction

In this chapter, we concentrate on a class of coarse-grained models that have been extensively used to study approximations of the protein folding problem. Using this model, we propose the use of EDAs for two related problems: to find the native structure of the protein from its sequence, and to simulate the protein folding mechanism.

The protein model of choice is the hydrophobic-polar (HP) model (66), which is based on the fact that hydrophobic interactions are a dominant force in protein folding. Although more complex models have been proposed, the HP model remains a focus of research in computational biology (44; 45; 46; 57).

In the optimization approach, the search for the protein structure is transformed into the search for the optimal configuration given an energy function that takes into account the HP interactions that arise in the model. The problem of finding such a minimum energy configuration is NP-complete for the 2-d (55) and 3-d (21) lattices. Performance-guaranteed approximation algorithms of bounded complexity have been proposed to solve this problem (89), but the error bound guaranteed is not small enough for many applications.

8.1.1 Overview of the chapter

The chapter is arranged as follows. In the next section, we introduce the HP model and the functional model protein. Section 8.3 reviews a number of previous approaches to the solution of simplified models using evolutionary and Monte Carlo (MC) based algorithms. Section 8.4 introduces the problem representation and discusses how the probability model can capture the regularities which may arise in the HP problem. In Section 8.5, the probabilistic models and the EDAs used for the protein structure problem are introduced. This section also presents the EDA model of protein folding. In

Section 8.7, the experimental benchmark is introduced and numerical results of our experiments are presented. Finally, in Section 8.8, the conclusions of our research are given, and further work is discussed.

8.2 The HP and functional model protein

The HP model considers two types of residues: hydrophobic (H) residues and hydrophilic or polar (P) residues. A protein is considered a sequence of these two types of residues, which are located in regular lattice models forming self-avoided paths. Given a pair of residues, they are considered neighbors if they are adjacent either in the chain (connected neighbors) or in the lattice, but not connected in the chain (topological neighbors). The total number (z) of topological neighboring positions in the lattice is called the lattice coordination number.

For the HP model, an energy function that measures the interaction between topological neighbor residues is defined as $\epsilon_{HH} = -1$ and $\epsilon_{HP} = \epsilon_{PP} = 0$. The HP problem consists of finding the solution that minimizes the total energy. In the linear representation of the sequence, hydrophobic residues are represented with letter H and polar ones, with P. In the graphical representation, hydrophobic proteins are represented by black beads, and polar proteins, by white beads. Figure 8.1 shows the graphical representation of a possible configuration for sequence $HHHPHPPPPPH$. The energy that the HP model associates with this configuration is -1 because there is only one HH contact, arisen between the second and fifth residues.

Although more complex models have been proposed (70; 116; 128; 188), the HP model remains a focus of research in computational biology (44; 45; 57). Among other reasons, it is considered as a useful model of an exhaustive sequence-structure map for the study of evolution (45), and it has been acknowledged that the hydrophobicity pattern in real proteins has statistical properties similar to those of 2-d HP model proteins (57). In evolutionary computation (58; 59; 69; 97; 117; 181; 192; 206), the model is still employed given its simplicity and its usefulness as a test bed for new evolutionary optimization approaches.

The functional model protein is a "shifted" HP model. The name comes from the fact that the model supports a significant number of proteins that can be characterized as functional. This model has native states, some of which are not maximally compact. Thus, in some cases, they have cavities or potential binding sites, a key property that is required in order to investigate ligand binding using these models (93). The energy values associated with the model contain both attractive $\epsilon_{HH} = -2$ and repulsive interactions $\epsilon_{PP} = 1$ and $\epsilon_{HP} = 1$. Again, the objective is to minimize the total energy. For example, the energy that the functional model protein associates with the configuration shown in Figure 8.1 is 0 because there is one HH and two PP contacts.

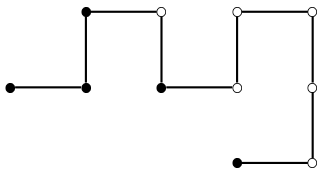


Figure 8.1: One possible configuration of sequence $HHHPHPPPPPH$ in the HP model.

8.3 Review of previous evolutionary methods

Previous population-based approaches to simplified protein folding include the use of nature-inspired and MC methods (144). Some versions of these methods are compared with EDAs in the experiments section.

Since the publication of the Unger and Moult paper (221) on the use of GAs for protein structure prediction, new issues have arisen along with new points of view on the protein folding problem. However, GA applications to the HP problem are many and varied. In (113), a search strategy called pioneer search was used together with a simple GA. Although the algorithm improved some of the results achieved in (221), it was unable to find the optimal solutions for the longest instances considered.

In (52) and (107), evolutionary algorithms for the 3-d HP problem are proposed. While in (107) a simple GA showed no better results than those achieved in (221), a more sophisticated approach is presented in (52). By using a backtracking-based repairing procedure, the latter algorithm guarantees that the search is constrained to the space of legal solutions. Since the number of self-avoided paths on square lattices is exponential in the length of the sequence (221), generating legal solutions with a backtracking algorithm is a feasible alternative.

The multimeme algorithm (MMA) for protein structure prediction (117) is a GA combined with a set of local searches. From this set, the algorithm self-adaptively selects which local search heuristic to use for different instances, states of the search, or individuals in the population. This algorithm was used to find solutions of the functional model protein. A relevant issue of this algorithm is the use of a contact map memory as a way to collect and use important problem information. Contact maps abstract the geometric details of the structures, keeping only the essential topological features of the configurations. In (181), MMA was extended by the incorporation of fuzzy-logic-based local searchers. The modifications led to obtain a more robust algorithm that improved previous MMA results in the protein structure prediction problem. Memetic algorithms were also combined with a population of rules (206) to solve the HP model in a two-

dimensional triangular lattice. The algorithm proposed outperformed simple versions of GAs and memetic algorithms.

Immune algorithms (IA) (58; 59) have been recently proposed for the HP problem. These evolutionary algorithms, inspired in the theory of clonal selection, use hypermacromutation and aging as important operators to proceed the search. In (59), the algorithm found the optimal configurations of the regular 2-d HP model for the smallest problems. The algorithm failed to find the optimum for the longest instances. In (58), the original IA is developed to include a memory mechanism that improves results for the 2-d regular lattice. HP problems on the 3-d lattice are also treated. When the feasible-space approach is used, IA results outperform those obtained in (52). However, IA results shown in (58) are no better than those achieved in (52) with the repair-based approach.

Traditional MC methods that use Markov chains sample from the protein folding space one point at a time. Due to the rugged landscape, these methods tend to get trapped in local minima. New MC methods have been proposed to cope with these problems (86). Among the alternatives proposed, two main classes of the strategies used by the MC methods can be distinguished: to use chain growth algorithms (17), or to sample the space with a population of Markov chains in which a different temperature is attached to each chain (134). Chain growth algorithms (75; 83; 98; 99) like the pruned-enriched Rosenbluth method (PERM) (99) are based on growing the sequence conformation by successively adding individual particles, guiding the growth towards configurations with lower energies, and using population control to eliminate bad configurations and increase good ones (83). Chain growth methods have achieved some of the best results for HP models in regular 2-d and 3-d lattices.

All the above-mentioned algorithms either use genetic operators or Markov chain transitions. They do not use any model of the search space. An algorithm that incorporates, to a certain scale, the modeling step is the ACO method presented in (203; 204). In this approach, the simulated ants construct candidate conformations for a given HP protein sequence, apply a local search to achieve further improvement, and update a probability value based on the quality of the solutions found. In ACO terminology, this value is called the pheromone trail.

Even though this short overview has focused on MCs and nature-inspired methods, we emphasize that there are several applications of heuristic algorithms to the protein structure problem that are beyond the scope of this thesis.

8.4 Dependencies in the simplified protein models

In this section, we show evidence on the emergence of regularities in the search space of the HP model. In order to achieve this goal, we employ the Boltzmann probability

$H(\mathbf{x})$	-4	-3	-2	-1	0	1	2	3	4	5	invalid	total
HP_f	0	0	16	1428	9581	0	0	0	0	0	8658	19683
FP_f	2	0	426	490	3407	3376	2020	912	350	42	8658	19683

Table 8.1: The density of the different energy levels HP_f and FP_f corresponding respectively to the HP and functional model protein of sequence $HHHPHPPPPPH$.

distribution. We start by introducing the problem representation.

8.4.1 Problem representation

Let n be the sequence length. For a given sequence and lattice, X_i will represent the relative move of residue i in relation to the previous two residues.

Taking as a reference the location of the previous two residues in the lattice, X_i takes values in $\{0, 1, \ldots, z-2\}$, where $z-1$ is the number of movements allowed in the given lattice. These values respectively mean that the new residue will be located in one of the $z-1$ numbers of possible directions with respect to the previous two locations. Therefore, values for X_1 and X_2 are meaningless. The locations of these two residues are fixed. A solution $\mathbf{x}$ can be seen as a walk in the lattice, representing one possible folding of the protein. The codification used is called relative encoding, and has been experimentally compared to absolute encoding in (118), showing better results.

We use 2-d and 3-d regular lattices. For regular d-dimensional lattices, $z = 2d$, where d is the lattice dimension.

8.4.2 Regularities and dependencies in the HP model

We illustrate the emergence of regularities in the search space of the HP model using the $HHHPHPPPPPH$ sequence, introduced in (93). For this sequence, and using the solution representation previously introduced, we find all possible solutions and evaluate them according to the HP and functional protein energy functions. The number of solutions evaluated are $3^9 = 19683$. Of these configurations, 8658 are not self-avoiding and they are assigned a very high energy equal to 100. In Table 8.1, $H(\mathbf{x})$ denotes all the possible values that the two evaluated energy functions can reach for sequence $HHHPHPPPPPH$. HP_f and FP_f respectively indicate the number of solutions where the corresponding value of $H(\mathbf{x})$ has been achieved for the HP and functional protein energy functions.

It is important to highlight that there is not a one-to-one mapping between each solution and each state of the sequence. The reason is that one state can have more than one

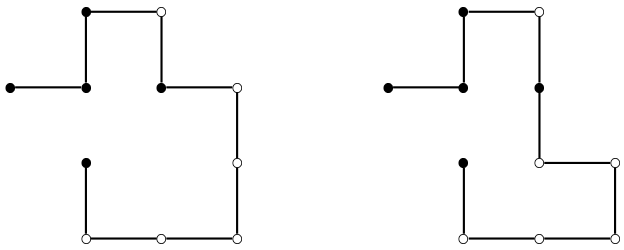


Figure 8.2: Best solutions of the functional model protein (left) and HP model (right) for sequence $HHHPHPPPPPH$.

solution representation, i.e. the representation is redundant. For instance, while there exists only one optimal state for the functional model protein, in our representation this optimal state has two possible representations corresponding to symmetrical configurations. In the case of the HP model, there are sixteen optimal solutions. Figure 8.2 shows optimal configurations for the functional model protein and the HP model.

To associate a probability value with every point of the search space, we will use a theoretical benchmark based on the Boltzmann distribution. In this benchmark, the probability of each solution is equal to the Boltzmann distribution calculated from the energy evaluation in the following way:

$$p(\mathbf{x}) = \frac{e^{\frac{-H(\mathbf{x})}{t}}}{\sum_{\mathbf{x}'} e^{\frac{-H(\mathbf{x}')}{T}}} \tag{8.1}$$

Equation (8.1) shows the expression of the Boltzmann distribution. We set the temperature at 1, but t can be changed to simulate different experimental conditions. The Boltzmann distribution is a natural candidate for the fitness distribution. From a theoretical point of view, the Boltzmann distribution allows us to associate probabilistic independence properties between the variables of the problem with certain characteristics of the energy function (160). It exhibits another convenient feature: higher probabilities are associated with points of the search space with better function evaluation. Therefore, these probabilities describe the desired performance of an ideal optimization algorithm: better points are visited with a higher probability. The Boltzmann distribution has also been used together with MC-based methods for HP model optimization (134).

From the Boltzmann distribution, we calculate the marginal probability distributions corresponding to variables (X_3, X_4, X_5) for probability distributions p_{HP} and p_{FP}, which are shown in Table 8.2. The table shows the marginal probabilities of the $3^3 = 27$ configurations (from 000 to 222) of variables (X_3, X_4, X_5).

	$p_{HP}(x_3, x_4, x_5)$			$p_{FP}(x_3, x_4, x_5)$		
Seq	$0--$	$1--$	$2--$	$0--$	$1--$	$2--$
-00	**0.000**	0.057	<u>0.069</u>	**0.000**	0.091	<u>0.151</u>
-01	0.035	0.036	0.038	0.009	0.029	0.033
-02	0.035	0.036	0.035	0.009	0.029	0.026
-10	0.029	0.034	0.035	0.021	0.026	0.027
-11	0.038	0.040	0.038	0.033	0.034	0.033
-12	0.035	0.034	0.029	0.027	0.025	0.021
-20	0.035	0.036	0.035	0.026	0.026	0.009
-21	0.038	0.036	0.035	0.034	0.030	0.009
-22	<u>0.069</u>	0.057	**0.000**	<u>0.151</u>	0.091	**0.000**

Table 8.2: Marginal probability distributions for (X_3, X_4, X_5) calculated from the Boltzmann distributions for HP and the functional model protein for sequence $HHHPHPPPPPH$.

Figure 8.3: Self-intersecting short paths: Configurations with zero probability.

Figure 8.4: Short helices: Configurations with the highest probability.

In Table 8.2, the lowest probability values are bold-faced, while the highest values are underlined. The two configurations with zero probability[1] are shown in Figure 8.3, where the direction of the sequence is represented as an arrow between the first and second residue of the sequence. Unsurprisingly, these are the only two self-intersecting configurations that can be formed with three contiguous moves. Any solution that contains subchains 000 or 222 is not self-avoiding, and therefore receives a very low probability.

The two configurations with the highest probabilities are shown in Figure 8.4. These are two symmetrical helices. In the HP and functional model protein, these types of substructures can significantly contribute to the final energy. Helices are present in the optimal solutions for both models, shown in Figure 8.2.

A remarkable difference between the marginal probabilities corresponding to the HP and functional model protein is related to the probabilities given to the helices. The difference between the probabilities of the best and second best configurations is 0.012 for the HP model, and 0.06 for the functional model protein. This gives an idea of the difference due to the energy function used.

One conclusion from this experiment is that, by inspecting the marginal distributions from the search distribution, we can extract relevant information about the problem, expressed in optimal and poor problem substructures. These substructures are likely to be present in population-based optimization algorithms able to respect the relevant interactions between the variables.

We investigate the effect that disregarding the potential interactions between variables may have on the modeling of the problem. In the next experiment, a univariate probability approximation of $p(x_3, x_4, x_5)$ is calculated. First, univariate marginal distributions are calculated for the three variables, $p(x_i) = \sum_{\mathbf{x}|X_i=x_i} p(\mathbf{x})$. Afterwards, the approximation $p_a(x_3, x_4, x_5)$ is computed as the product of the univariate marginals $p_a(x_3, x_4, x_5) = p(x_3)p(x_4)p(x_5)$. The approximation is shown in Table 8.3.

As in Table 8.2, the lowest probability values are bold-faced, while the highest values are underlined. Due to the effect of rounding, some other configurations appear in the table with probabilities that are equal (rounded) to the lowest (respectively highest) ones.

Table 8.3 shows that the best and worst configurations do not agree with the ones obtained using the whole three-variate marginal probability distribution. The univariate

[1]Strictly speaking, the probabilities are never zero, but they approximate this value.

	$p_{HP}(x_3, x_4, x_5)$			$p_{FP}(x_3, x_4, x_5)$		
Seq	$0--$	$1--$	$2--$	$0--$	$1--$	$2--$
-00	0.036	0.042	0.036	0.044	0.029	0.044
-01	0.036	0.042	0.036	0.029	**0.019**	0.029
-02	0.036	0.042	0.036	0.044	0.029	0.044
-10	**0.033**	0.038	**0.033**	0.054	0.035	0.054
-11	0.033	0.039	0.033	0.035	0.023	0.035
-12	**0.033**	0.038	**0.033**	0.054	0.035	0.054
-20	0.036	0.042	0.036	0.044	0.029	0.044
-21	0.036	0.042	0.036	0.029	**0.019**	0.029
-22	0.036	0.042	0.036	0.029	0.029	0.044

Table 8.3: Univariate approximation of the marginal probability of (X_3, X_4, X_5) calculated from the Boltzmann distributions for HP and functional model protein for sequence $HHHPHPPPPPH$.

approximation is not able to capture the structural features of the problem represented in Table 8.2. This experiment illustrates the convenience of using higher order interactions to capture relevant features of the problem structure. As analyzed in previous sections, traditional crossover operators do not respect these interactions. Furthermore, as explicit modeling of the search space is missing in most nature-inspired algorithms, it is impossible to detect, represent, and store these regularities efficiently. In the following sections, we show how different probability models can detect and exploit this information.

The experiments presented in this section have been conducted using a single instance. Obviously, there are other factors that influence the marginal probability distributions corresponding to the different energy models. We have focused on showing the way in which structural regularities are exposed by the probability models learned. The analysis of other factors is beyond the scope of this thesis.

8.5 EDAs for protein structure prediction

The existence of regularities in the search space, expressed in probabilistic dependencies between subsets of variables, naturally leads to the convenience of using EDAs to take advantage of these regularities by capturing the dependencies. Probability models used by EDAs are built from the selected set of solutions. Therefore, the type of selection method used also influences the number and strength of the interactions learned by the model. Similarly to the Boltzmann distribution analyzed before, selection methods assign higher selection probabilities to solutions with higher fitness values.

In this section, we detail the main contributions of this chapter: The introduction of EDAs to face the solution of the protein structure prediction problem, and the definition of the EDA-based model of protein folding. The section starts by introducing the probability models used by EDAs and explaining the rationale behind our choice. At the end, we define the EDA-based model of protein folding and describe the analogies between this model and some of the known behavioral characteristics of protein folding.

8.5.1 Probabilistic models used

We propose three types of probabilistic models to be applied to the protein structure prediction problem. In every case, solutions are represented using the vector representation introduced in Section 8.4.1.

The first model considered is a k-order Markov model in which the configuration of variable X_i depends on the configuration of the previous k variables, where $k \geq 0$ is a parameter of the model. The joint probability distribution can be factorized as follows:

$$p_{MK}(\mathbf{x}) = p(x_1, \ldots, x_{k+1}) \prod_{i=k+2}^{n} p(x_i \mid x_{i-1}, \ldots, x_{i-k}) \tag{8.2}$$

The second probabilistic model is based on a tree where each variable may depend on no more than one variable that is called the parent. A probability distribution $p_{Tree}(\mathbf{x})$ that is conformal with a tree is defined as:

$$p_{Tree}(\mathbf{x}) = \prod_{i=1}^{n} p(x_i \mid pa(x_i)) \tag{8.3}$$

where $Pa(X_i)$ is the parent of variable X_i in the tree, and $p(x_i \mid pa(x_i)) = p(x_i)$ when $Pa(X_i) = \emptyset$, i.e. when 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 undirected graphs.

The third model considered is a mixture of trees (142). A mixture of trees is defined as a distribution of the form:

$$p_{MT}(\mathbf{x}) = \sum_{j=1}^{m} \lambda_j p_{Tree}^{j}(\mathbf{x}) \tag{8.4}$$

with $\lambda_j > 0$, $j = 1, \ldots, m$, and $\sum_{j=1}^{m} \lambda_j = 1$.

In this case, the tree distributions are the mixture components. The m trees may have different structures and different parameters.

The models proposed can be separated into two classes according to the part of the problem structure that they exploit. The first class of probability models is based on the existence of connected neighbors. The assumption behind the use of Markov models is that the most important source of problem interactions comes from the connected neighbors. Markov models have been used in computational biology to identify coding regions in proteins, to align sequences, and to predict the protein secondary structure.

The second class of models allows for the existence of arbitrary connections between the variables of the problem subject to the representation constraints determined by the probabilistic model. This choice of the models tries to capture interactions arising from both, connected and topological neighbors. Therefore, algorithms that learn the structure of the model from the data (49; 142) are incorporated. Models that belong to this class differ in the type of structural constraints which they represent.

Results of an EDA based on the Markov model, for the solution of 2-d lattice problems, were presented in (192). These EDAs make a parametric learning of the model. An EDA called combining optimisers with mutual information trees, which searches for probabilistic models that can be represented using tree-shaped networks, was introduced in (16). The mixture of trees FDA (MT-EDA), an EDA that uses mixture of trees models to estimate and sample the probability, was presented in (196). These algorithms have mainly been applied to binary problems. Poor results for preliminary experiments conducted using EDAs based on unconstrained Bayesian networks (72) determined to discard these algorithms from our experimental benchmark.

8.5.2 Implementation

In the chosen representation, there might be invalid vectors that correspond to self-intersecting sequences. To enforce the validity of the solutions, we employ a variation of the backtracking method used in (52). A solution is incrementally repaired in such a way that the self-avoidance constraint is fulfilled. At position i, the backtracking call is invoked only if self-avoidance cannot be fulfilled with any of the possible assignments to X_i. The order of the assignment of values is random. If all the possible values have been checked, and self-avoidance is not fulfilled yet, backtracking is invoked.

On the other hand, if the number of backtracking calls have reached a pre-specified threshold, the repair procedure is abandoned. This is a compromise solution for situations in which the repair procedure can be too costly in terms of time. The threshold for the number of backtracking calls was set to 500 and this values was determined empirically. Further details about the original backtracking algorithm can be found in (52).

In our implementation of EDAs, the truncation selection of parameter $T = 0.1$ is used. Let M be the population size. In this type of selection, the best $N = T \cdot M$ individuals, according to their function evaluations, are selected. 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 the first one.

We name EDAs according to the probability model that they use. MK-EDA$_k$ is the Markov EDA, where k is the order of the interactions. The EDA that uses a tree model is called Tree-EDA (see Algorithm 3.3). MT-EDA$_m$ corresponds to the EDA that uses a mixture of trees, m being the number of components in the mixture.

8.6 EDAs as a model of the protein folding mechanism

8.6.1 The "new" view of protein folding

The EDA-based model of protein folding presented in this chapter adopts the "new" view of protein folding (see Section 7.3.1). Therefore, we must study in greater detail some of the aspects related to it.

In the "new" view approach, the *energy landscape of a folding protein* resembles a partially rough funnel. The local roughness of the funnel reflects transient trapping of the protein configurations in local free energy minima. In the protein structure, *frustration* is the result of competition between many conflicting energy contributions. The various interactions are "frustrated".

Order parameters or *progress coordinates* help to describe and quantify the protein ensembles during the protein folding process. They are used to explore the connection between the folding process and the topology of the protein native state. Examples of order parameters are the *contact order* and the *volume of the protein*. Another quantifying measure is the *folding rate*.

The *contact order* of the protein is the average sequence separation between residues that make contact in the three-dimensional native structure. The *volume* is a measure of the degree of folding of the protein, allowing to distinguish between compact and extended conformations. The *folding rate* is the amount of time which the protein takes to fold.

In the case of small proteins, other measurements of folding reactions can be made. Among them are the distribution of structures in the transition state ensemble, and the structure of the native state. The *fraction of native contacts* Q that exist in the current conformation (171) can be used as a measure of frustration. For a given conformation, Q varies between 0 and 1, with the native conformation at $Q = 1$. It is also possible to compute the total free energy $F_{tot}(Q)$ as a function of Q,

$$F_{tot}(Q) = F_{int}(Q) - T \cdot S_{conf}(Q) \tag{8.5}$$

where $F_{int}(Q)$ is the average internal energy of conformations with Q native contacts, T is the temperature of the system, and $S_{conf}(Q)$ is the corresponding conformational entropy (the logarithm of the number of accessible conformations with Q native contacts) (13).

We enumerate a number of facts commonly accepted and explained in the "new" view of protein folding (13; 47; 171; 172). Some of these issues will be investigated through simulation of the EDA-based model:

- The folding rates of small proteins correlate with their contact order. Proteins with a large fraction of their contacts between residues close in sequence tend to fold faster than proteins with more non-local interactions.
- Protein folding rates and mechanisms are largely determined by the protein native topology. Proteins with similar native states are expected to exhibit a similar protein folding behavior.
- Local interactions are more likely to form early in folding than non-local interactions.
- During the folding process, the energy of the structures will decrease on average as they become more and more similar to the native structure of a natural protein.
- Folding is not only determined by properties of the folded state but also by the energetic difference between the folded and unfolded ensembles of states.
- The geometrical accesibility of different native contacts is different, and therefore some are more easily formed than others.
- Some contacts may be topologically required (or at least be more likely) to be formed before others during folding.

The existence of a number of analogies between the "new" view of the protein folding mechanism and the way EDAs behave motive us to analyze EDAs as a model of protein folding. Basically, we highlight these coincidences, drawing parallels between both entities, and investigating to what extent each of the entities can provide answers to questions rising in the other domain.

To explain our model of the protein folding process, we will use the same representation introduced in Section 8.4.1. We will assume that all solutions are feasible (i.e. self-intersecting paths are repaired). At first, during the real folding process, a protein can only be in one state at each time t. However, in EDAs, at each time, more than one configuration can be part of the population. To cover this gap, we will assign the main role in modeling to the EDA probabilistic model. This resembles the "new" view of protein

folding, where proteins are seen as an ensemble of rapidly interconverting conformations. At each time, a probability $p(\mathbf{x})$ can be associated with every possible configuration $\mathbf{x}$ of the sequence. This probability is related to the energy of the configuration, usually using the Boltzmann distribution defined in Equation (8.1).

Consider that a given EDA shall model the protein folding process. Each generation of the EDA will be considered a time step of the folding process. We will assume that the probability of the sequence to fold to a given conformation is equal to the probability given to the same configuration by the probabilistic model of the EDA constructed at generation t.

Starting from this assumption, we advance the following statements:

- Both the "new" view and the EDA define a sampling of the space of configurations.
- The sampling process pursues to sample the sequence configurations with a probability that depends on the quality of their respective energy function evaluations.
- The goal of the EDA and the protein folding process is achieved when $p(\mathbf{x}) = 1$, where $\mathbf{x}$ is the protein's native state.
- Both entities tend to preserve local favorable conformational features through successive generations (time steps).

In principle, some of the features presented above are not exclusive attributes of EDAs. For instance, other population-based methods (e.g. GAs) can be used to sample the space of sequence configurations. The preservation of local favorable configurations, which may correspond to autonomous folding units in the real protein folding process, can also be accomplished to a certain extent by different evolutionary methods. However, the advantage of EDAs is that the probability model which they employ treats phenomena like solution disruption and frustration, which may arise in the protein folding process, more effectively. Although in GAs a probability distribution of the solutions is implicitly used for the search, this probability distribution is explicitly learned and used in EDAs.

As explained in previous sections, traditional crossover operators tend to disrupt the construction of relevant subsolutions. The probabilistic model used by EDA matches the statistical nature of the ensemble of conformations. This model is a condensed description of the selected population and, under suitable conditions, it also matches the EDA population at time $t + 1$ well. The main advantage of an EDA model of protein folding is that it can provide not only global statistical information, but also information about the local conformations in the ensemble. This information can be appropriately combined to avoid the disruption of relevant subsolutions.

Let us exemplify this with the frustration problem. In frustrated systems, there are contacts that are locally unfavorable but that exist in the optimal solution, or there

are favorable contacts that must first be broken to reach the optimal solution (171). Analogous to a frustrated system would be a population-based method that tries to optimize a frustrated function.

Let $f(\mathbf{x})$ be a function that can be decomposed into the sum of local functions defined on (possibly overlapping) subsets of its variables. $f(\mathbf{x})$ is said to exhibit frustration when the point $\mathbf{x}$ where its global optimum is reached does not maximize one or more of the local functions.

These types of functions are difficult to optimize because finding the optima of the local subfunctions does not guarantee that the global optimum will be found by the combination of these optima. The role of frustration as a source of hardness for evolutionary algorithms has been discussed in (106). In (160), it is shown that, taking into account the interactions that arise between the variables of the problem, EDAs can optimize frustrated functions. Although the capacity of EDAs to deal with frustration also depends on the probability model employed, we emphasize, to this respect, the suitability of using EDAs over GAs.

In the section of experiments, devoted to the simulation of the protein folding process using EDAs, we investigate some of the features exhibited by the EDA model that mimic the behavior of the protein folding process. The issues considered are the following:

1. Whether there is a correlation between the successful rate of EDAs and the contact order of the protein models.
2. Whether there is a relationship between the generation convergence of EDAs for the HP model, and the contact order of the optimal solution.
3. Whether there are differences in the rate of formation of native contacts, and if these differences are associated with their contact separation.

In our simulations, we will employ some of the order parameters commonly used to investigate the protein folding process, but adapted to the simplified models. For the functional model protein, the contact order is calculated as the average sequence separation of the HH contacts in the corresponding solution. For example, the contact order of the configurations shown in Figure 8.1 and Figure 8.2 (left) are, respectively, 3 and 6.

8.7 Experiments

In this section, we present experiments on the use of the EDAs presented in this chapter. The section is divided into four main parts. Section 8.7.1 presents the problem instances used for the HP model, and the benchmark used for the functional model protein. In

the second part, we present the results of the protein structure prediction problem in the 2-d (Section 8.7.2) and 3-d (Section 8.7.3) regular lattices. Finally, Section 8.7.4 shows the results of the study through EDA simulations of some factors related to the protein folding process.

8.7.1 Problem benchmark

The HP instances used in our experiments, and shown in Table 8.4, have previously been used in (17; 52; 132; 192; 203; 204; 221). The values shown in Table 8.4 correspond to the best-known solutions ($H\mathbf{x}^*$)) for the 2-d regular lattice. It is important to highlight that most of the randomly generated amino acid sequences do not behave like natural proteins, because the latter are products of natural selection. Likewise, most randomly generated sequences of H and P residues in the HP model do not fold to a single conformation (45).

inst.	size	$H(\mathbf{x}^*)$	sequence
$s1$	20	-9	$HPHPPHHPHHPHPHHPPHPH$
$s2$	24	-9	$HHPPHPPHPPHPPHPPHPPHPPHH$
$s3$	25	-8	$PPHPPHHP^4HHP^4HHP^4HH$
$s4$	36	-14	$P^3HHPPHHP^5H^7PPHHP^4HHPPHPP$
$s5$	48	-23	$PPHPPHHPPHHP^5H^{10}P^6$
			$HHPPHHPPHPPH^5$
$s6$	50	-21	$HHPHPHPHPH^4PHP^3HP^3HP^4$
			$HP^3HP^3HPH^4\{PH\}^4H$
$s7$	60	-36	$PPH^3PH^8P^3H^{10}PHP^3$
			$H^{12}P^4H^6PHHPHP$
$s8$	64	-42	$H^{12}PHPH\{PPHH\}^2PPH\{PPHH\}^2$
			$PPH\{PPHH\}^2PPHPHPH^{12}$
$s9$	85	-53	$H^4P^4H^{12}P^6H^{12}P^3H^{12}P^3$
			$H^{12}P^3HP^2H^2P^2H^2P^2HPH$
$s10$	100	-48	$P^6HPH^2P^5H^3PH^5PH^2P^4H^2$
			$P^2H^2PH^5PH^{10}PH^2PH^7$
			$P^{11}H^7P^2HPH^3P^6HPH$
$s11$	100	-50	$P^3H^2P^2H^4P^2H^3PH^2PH^2PH^4$
			$P^8H^6P^2H^6P^9HPH^2PH^{11}P^2$
			$H^3PH^2PHP^2HPH^3P^6H^3$

Table 8.4: HP instances used in the optimization experiments.

For the experiments with the EDA-based model, we have selected the functional model protein. The existence of a unique native state for the instances of this model is a

desired attribute for our analysis. We have employed a set of 15545 functional model proteins[2] that were optimally embedded on a 2-d square lattice (93). For each instance, the benchmark provides the energy of the unique native state, together with the energy value and number of structures that are in the first excited state (best sub-optimum). The length for each sequence is 23. Some of these instances have been previously used as a benchmark in (117).

	MK-EDA$_2$			Tree-EDA			MT-EDA$_4$		
inst.	$H(\mathbf{x})$	S	$\bar{g}$	$H(\mathbf{x})$	S	$\bar{g}$	$H(\mathbf{x})$	S	$\bar{g}$
$s1$	-9	50	3.34	-9	50	2.96	-9	50	3.18
$s2$	-9	50	3.68	-9	50	3.80	-9	50	3.84
$s3$	-8	50	4.14	-8	44	5.72	-8	45	5.24
$s4$	-14	4	8.75	-14	2	13.50	-14	8	9.50
$s5$	-23	7	19.57	-23	9	23.66	-23	2	21.00
$s6$	-21	43	11.72	-21	49	12.67	-21	48	12.95
$s7$	-35	5	55.06	-35	6	982.00	-35	9	121.22
$s8$	-42	3	31.18	-41	5	50.70	-42	6	78.16
$s9$	-52	2	218.00	-51	1	1355.00	-50	2	2161.05
$s10$	-46	3	922.50	-46	8	1445.70	-47	1	707.00
$s11$	-47	1	215.00	-47	6	1778.80	-48	1	845.00

Table 8.5: Results of EDAs for HP instances in the 2-d lattice.

8.7.2 Results for the HP model in the two-dimensional lattice

The first experiment consists of finding the optimum of sequences shown in Table 8.4 using EDAs with different probability models. The EDAs described in Section 8.5 (MK-EDA$_2$, Tree-EDA and MT-EDA$_4$) are used in our experiments. All the algorithms use a population size of 5000 individuals and a maximum of 5000 generations. The results of the experiments are shown in Table 8.5, where S is the number of times that the best solution has been found in 50 experiments, and $\bar{g}$ is the average number of generations needed to find the best solution for the first time. This value gives an idea of the number of function evaluations needed to reach the best solution.

The first remarkable result is that all EDAs are able to find the optimum solution for sequences $s1$-$s6$. All the algorithms find the second best solution for sequence $s7$, the best or second best for sequence $s8$, and very good solutions for the rest of the longer sequences. There are two facts that help to put these results in perspective: EDAs do not

[2] http://www.cs.nott.ac.uk/∼nxk/HP-PDB/2dfmp.html

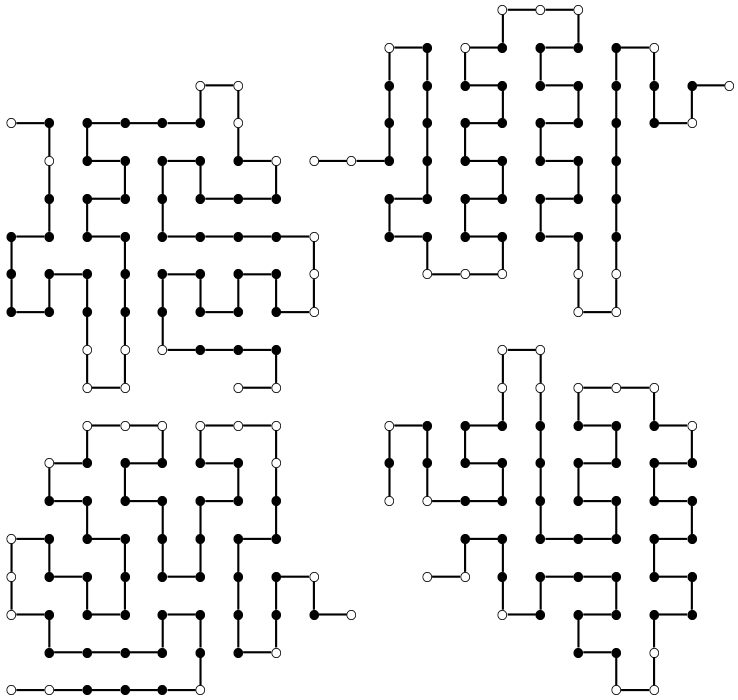


Figure 8.5: Optimal solution (bottom left) and three sub-optimal solutions for the $s7$ sequence.

use local optimizers that could improve the results, and the parameters of the algorithms have not been tuned for every instance.

Deceptive instances for EDAs

Instance $s7$ is deceptive for EDAs. We investigate the performance of the EDAs for this instance in detail. Detailed research on the dynamics of EDAs for deceptive problems throws light on the limitations of these methods and could contribute to their improvement.

The optimum of sequence $s7$ is -36. There are many suboptimal solutions with value -35. Figure 8.5, bottom left, shows the optimum solution that cannot be found by the EDAs. The rest of the solutions are the suboptimal ones ($H(\mathbf{x}) = -35$) found by the EDAs. A clear difference between the optimal and the rest of the solutions shown in the figure is the number of short helices. The optimum has fewer short helices than the other solutions. Most of the energy contributions come from interactions between a central cross-shaped structure and the neighboring residues. As the optimal solution cannot

k	$p(\mathbf{x}_{opt})$	$max(p(\mathbf{x}))$	$mean(p(\mathbf{x}))$	$min(p(\mathbf{x}))$	$N(p(\mathbf{x}) > p(\mathbf{x}_{opt}))$
0	$9.6546e^{-27}$	$2.0721e^{-20}$	$1.2853e^{-22}$	$2.9364e^{-36}$	952
1	$6.8792e^{-23}$	$2.1155e^{-11}$	$9.8979e^{-14}$	$9.0532e^{-37}$	4952
2	0	$1.7751e^{-06}$	$4.0864e^{-09}$	$1.9233e^{-30}$	5375
3	0	$1.4889e^{-05}$	$4.4991e^{-08}$	$1.9347e^{-27}$	5375

Table 8.6: Statistical information extracted from k-order Markov probabilistic models ($0 \leq k \leq 3$) of the 5375 solutions of the $s7$ sequence with energy lower than -32.

be constructed from the combination of the good substructures present in the other solutions, the EDAs cannot reach it. As a hypothesis for the reason of this deceptive behavior, we advance the existence of isolated optimal solutions with components that are not present in the suboptimal solutions.

To validate this empirical conclusion, we calculate different Markov probabilistic models ($k = 0, \ldots, 3$) from the 5375 solutions whose energy lies between -33 and -35. Using these models, the probabilities corresponding to each of the database solutions and to the optimal solution shown in Figure 8.5 are calculated. The models indicate the probability for the solutions to be present at the new EDA generation.

Results can be appreciated in Table 8.6. In this table, $p(\mathbf{x}_{opt})$, $max(p(\mathbf{x}))$, $mean(p(\mathbf{x}))$ and $min(p(\mathbf{x}))$ respectively correspond to the probabilities given by the models with different k values (from 0 to 3) to the optimum of the problem, and the maximum, mean and minimum probabilities given by the models to the 5375 solutions.

The most revealing fact is that the probability given to the optimal solution by probability models with $k = 2$ and $k = 3$ is zero. This means that the optimal structure does not share some of its substructures with any of the other 5375 suboptimal solutions. The analysis of the model revealed that, in the case of $k = 2$, only one of the substructures was absent in the other solutions, while when $k = 3$, four substructures were absent. $N(p(\mathbf{x}) > p(\mathbf{x}_{opt}))$ refers to the number of solutions that are assigned a probability by the model higher than the one assigned to the optimum.

This experiment shows that deception also arises in the case of the HP model, and that deceptive instances for the EDAs can be found and described as those which do not share a number of good substructures with most of the closest suboptimal solutions (in terms of the objective function value). These substructures can not be captured by the probability models used.

Comparison with other algorithms

The performance of EDAs is now compared with the best results achieved with other evolutionary and MC optimization algorithms. The results are shown in Table 8.7. The results of GA (221) and MMA (117) correspond to the best solution found in five runs. The results of ACO and NewACO (204) are based on 20-700 tries for the former algorithm and 300-500 for the latter (204). PERM (99) reports 20-200 tries for sequences $s9$ and $s11$; the other instances have been faced in (204). Results for other optimization methods (58; 59; 113) were not displayed because either they were unable to find the best results achieved by EDAs or the number of function evaluations required to find them was much higher. All the results shown for EDAs are obtained from 50 runs.

A first conclusion is that none of the algorithms is able to outperform the rest of algorithms for all the instances. PERM is the best contender in all cases except $s8$ in which its results are very poor. EDA and PERM score the same at sequence $s11$. In comparison with the NewACO, EDA reaches equal or better results in all instances except one. It should be noted that NewACO applies local optimization techniques. In this sense, a fairer comparison would be between EDAs and ACO. In such a comparison, EDA is the clear winner. Analysis shows that none of the rest of the algorithms achieves similar results.

Dynamics of the algorithms

Another relevant issue related to EDAs concerns their particular dynamics for the HP models. The existence of a model of the search space enables a compact representation of characteristic features of the best solutions but also determines the particular way in which the optimal solutions are constructed.

In the next experiment, we analyze the convergence dynamic of the three types of EDAs employed in our experiments in the optimization of sequence $s6$. The average fitness of the best solutions at each generation is calculated by running each algorithm 50 times. The population size and the rest of the parameters were the same as in previous experiments. The results are shown in Figure 8.6. The figure shows that, for all EDAs, a small number of generations is enough to find the optimal solutions. Nevertheless, there are differences in the dynamics of the algorithms. MK-EDA$_2$ reaches better solutions earlier than the other algorithms, but in the experiments conducted, MT-EDA$_4$ reached the optimum more often. Tree-EDA is the slowest algorithm.

8.7.3 Results of the HP model in the three-dimensional lattice

In the following experiment, we investigate the behavior of EDAs in the solution of the HP model in the regular 3-d lattice. EDAs are compared with an evolutionary algorithm

inst.	*BestEDA* $H(\mathbf{x})$	*GA* $H(\mathbf{x})$	*MMA* $H(\mathbf{x})$	*ACO* $H(\mathbf{x})$	*NewACO* $H(\mathbf{x})$	*PERM* $H(\mathbf{x})$
$s1$	-9	-9	-9	-9	-9	-9
$s2$	-9	-9	-9	-9	-9	-9
$s3$	-8	-8	-8	-8	-8	-8
$s4$	-14	-14	-14	-14	-14	-14
$s5$	-23	-22	-22	-23	-23	-23
$s6$	-21	-21		-21	-21	-21
$s7$	-35	-34		-34	-36	-36
$s8$	-42	-37		-32	-42	-38
$s9$	-52				-51	-53
$s10$	-47				-47	-50
$s11$	-48				-47	-48

Table 8.7: Results achieved by different search heuristics for the HP instances.

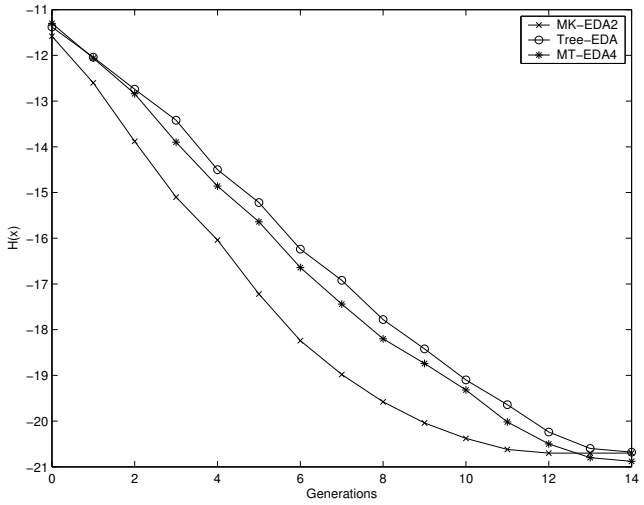


Figure 8.6: Average fitness of the best solution at each generation for different EDAs in sequence $s6$.

	hybrid GA		$MK\text{-}EDA_2$		TreeEDA		$MT\text{-}EDA_4$	
	$H(\mathbf{x})$	$mean \pm \sigma$	$H(\mathbf{x})$	$mean \pm \sigma$	$H(\mathbf{x})$	$mean \pm \sigma$	$H(\mathbf{x})$	$mean \pm \sigma$
$s1$	-11	-10.52 ± 0.54	-11	-10.82 ± 0.38	-11	-10.68 ± 0.51	-11	-10.84 ± 0.37
$s2$	-13	-11.28 ± 0.90	-13	-12.02 ± 0.94	-13	-11.30 ± 0.85	-13	-11.88 ± 0.93
$s3$	-9	-8.54 ± 0.64	-9	-8.96 ± 0.19	-9	-8.92 ± 0.27	-9	-9.00 ± 0.00
$s4$	-18	-15.76 ± 1.05	-18	-16.40 ± 0.80	-18	-16.24 ± 0.83	-18	-16.50 ± 0.96
$s5$	-28	-24.60 ± 1.57	-29	-27.24 ± 0.92	-29	-26.88 ± 0.93	-29	-27.06 ± 1.08
$s6$	-26	-23.02 ± 1.48	-29	-25.70 ± 1.26	-31	-25.94 ± 1.58	-28	-25.74 ± 1.22
$s7$	-49	-41.18 ± 2.75	-49	-46.30 ± 2.04	-49	-43.78 ± 3.10	-48	-42.00 ± 6.76
$s8$	-46	-40.40 ± 2.50	-52	-46.78 ± 2.28	-49	-43.72 ± 2.43	-50	-45.64 ± 2.03

Table 8.8: Results of the EDAs and the hybrid GA in the three-dimensional lattice.

hybridized with a backtracking method (hybrid GA). In (52), different variants of the evolutionary algorithm were employed. Absolut and relative encoding were evaluated. For the comparisons with EDAs, we have selected the variant that gives the best results. This corresponds to the evolutionary algorithm that uses an absolute encoding.

The hybrid GA used a population size of 100 individuals and a maximum of 10^5 evaluations. This is not an appropriate population size for EDAs. We use a population size of 5000 individuals and a maximum of 1000 generations. In almost all sequences, this implies a higher number of function evaluations than the 10^5 used in (52). The average number of evaluations needed by EDAs was between $6 \cdot 10^4$ for the shortest instances and 10^6 for the longest ones.

The results are shown in Table 8.8. In this case, the optimum is not known and it is critical to establish whether differences among the methods are statistically significant. We have used the Kruskal-Wallis test to accept or reject the null hypothesis that the samples have been taken from equal populations. The test significance level was 0.01. For all the instances considered, significant statistical differences have been found between the hybrid GA results and those achieved by the Markov ($MK\text{-}EDA_2$) and mixture ($MT\text{-}EDA_4$) EDAs. Significant statistical differences between the hybrid GA and the TreeEDA have been detected only for instances $s3$, $s5$, $s6$, and $s8$. All the EDAs have a better average fitness of solutions, showing that the algorithm clearly outperforms the hybrid GA. Furthermore, as can be observed in Table 8.8, EDAs find new best solutions for sequences $s5$, $s6$ and $s8$.

8.7.4 Results of the protein folding simulations

In this section, we present results on the use of EDAs to simulate protein folding in the HP model. We investigate to which extent EDAs are able to mimic some characteristic features of the protein folding mechanism. For all the simulation experiments, we use $MK\text{-}EDA_2$. In all the experiments, the population size was set at 2000 and the maximum number of generations was 20.

Energy landscape of the model

The first experiment investigates the energy landscape of the function for one of the 15545 functional model protein instances available. The goal of this experiment is to build some intuition about the fitness landscape of the functional model protein. Particularly, the goal is to illustrate the fact that, as solutions share more of the protein native contacts, their fitness (energy) decreases.

The experiments consist of executing the EDA and storing all the conformations visited during the search. Sequence $PHHPPHPPHHPPHPHPPHPPHHH$ was chosen for the experiment. For every conformation visited during the search, the total number of contacts (K) and the number of native contacts (Q) are calculated. We classify the conformations using these two parameters, and calculate the average energy of all conformations that share the parameters (K, Q). The native state of the sequence has 9 native contacts. Therefore $0 \leq K, Q \leq 9$.

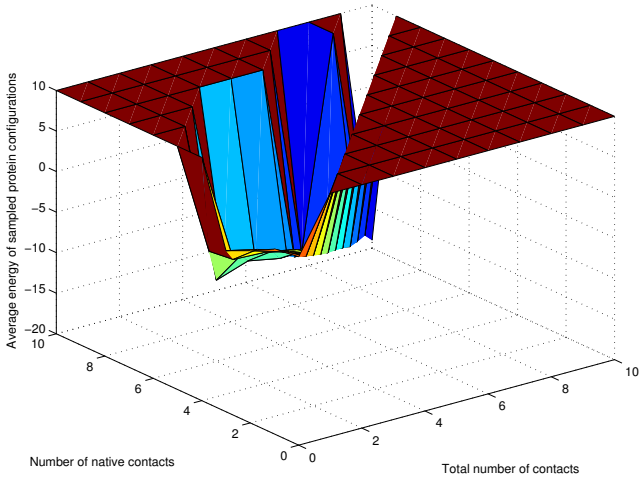


Figure 8.7: Energy landscape of the functional model protein corresponding to sequence $PHHPPHPPHHPPHPHPPHPPHHH$ as sampled by MK-EDA$_2$.

Figure 8.7 shows the average energy of all the sampled conformations grouped using the different values of K and Q. The figure reveals that, as the number of native contacts increases in the conformations, the average energy decreases.

Throughout the evolution, MK-EDA$_2$ is able to explore the different regions of the energy landscape, no only those regions with high energy which are abundant in the search space, but also those corresponding to low energy values where the number of conformations is scarce. For example, MK-EDA$_2$ is able to locate the optimum which is unique.

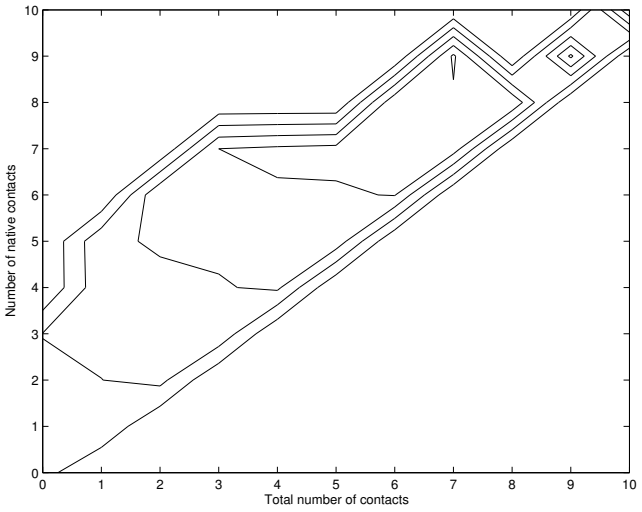


Figure 8.8: Energy landscape contour graph of the functional model protein corresponding to sequence $PHHPPHPPHHPPHPHPPHPPHHH$ as sampled by MK-EDA$_2$.

To appreciate the characteristics of this landscape in detail, Figure 8.8 shows the contour graph corresponding to the same experiment. Sets of conformations with similar average energy are joined in the graph by the same contour lines. Regions with lower energy are those with many native contacts.

The samples obtained by MK-EDA$_2$ make it possible to observe the correlation between the presence of native contacts in the conformations and the quality of the folding (low energy). This fact is expected in proteins that have evolved to diminish the degree of frustration and achieve a fast folding rate. However, it does not mean that, for every simplified energy function, the EDA will be able to move in the direction of the native configuration by augmenting the number of native contacts. First, the definition of realistic protein folding energy functions such that their only significant basin of attraction is the native state has been recognized as a very difficult task (174). Second, the phenomenon of frustration can arise in the functional model protein and other energy functions, moving the MK-EDA$_2$ away from the native state.

Influence of the contact order in the protein folding process

We investigate whether the EDA can reflect the influence of the contact order parameter in the protein folding process. As discussed in Section 8.6.1, proteins with a large fraction of their contacts between residues close in sequence tend to fold faster than proteins with

more non-local interactions. This section shows how the contact order of the functional model protein instances influences on the success rate and average number of generations needed by MK-EDA$_2$ to solve the problem.

As a preprocessing step, 100 executions of MK-EDA$_2$ are done for each of the 15545 functional model proteins instances. The goal of this step is to determine the easier instances for the EDA, and to calculate, for all instances, a set of statistics from the best solutions found at each run. Of the 15545 instances, MK-EDA$_2$ was able to find the optimum in 12588 instances at least once. For 703 instances, the algorithm found the optimum in at least 95 runs, and for 176 it found the optimum in 100 runs.

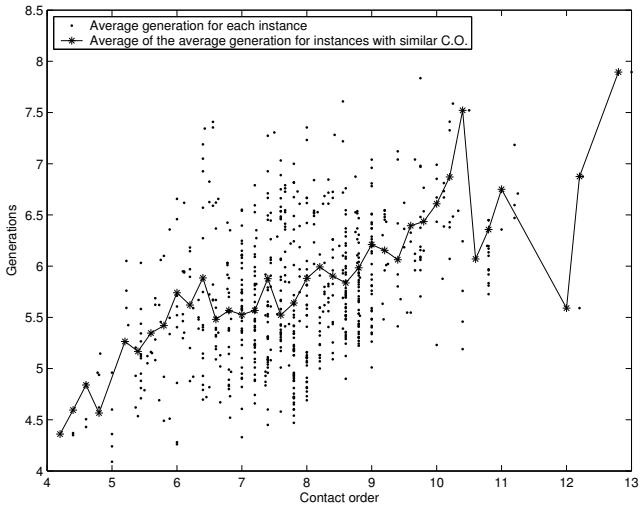


Figure 8.9: Relationship between the contact order of the sequences where MK-EDA$_2$ has a success rate above 95 and the average number of generations needed to converge.

To evaluate the relationship between the contact order of the sequences and the average number of generations needed to solve them, we consider those instances for which MK-EDA$_2$ has a success rate higher than or equal to 95. The choice of this set is determined by the need to have an accurate estimate of the average number of generations. From this set of instances, Spearman's rank correlation coefficient is calculated between the contact order and the average number of generations. Spearman's rank correlation is the statistic of a non-parametric test usually employed to test the direction and strength of the relationship between two variables.

With a significance level of 0.01, and Spearman's rank correlation equal to 0.3939, the test refuses the null hypothesis of lack of correlation. In Figure 8.9, the average generation and the contact order (C.O.) for the set of selected instances are plotted. Additionally,

the figure shows the average number of generations calculated from instances with similar contact orders. It can be seen that the number of generations needed to solve the problem grows if the contact order grows.

Using the test based on Spearman's rank correlation coefficient with the same significance level, we have evaluated the relationship between the contact order of the sequences where MK-EDA$_2$ found the optimum at least once, and the success rate achieved by the algorithm for these instances. The test rejected the hypothesis that the observed correlation was due to random effects. The parameter of the correlation provided by the test was equal to -0.2848. This means that, as the contact order of the instance grows, the success rate of the algorithm diminishes. Figure 8.10 shows the average success rate for instances with similar contact order.

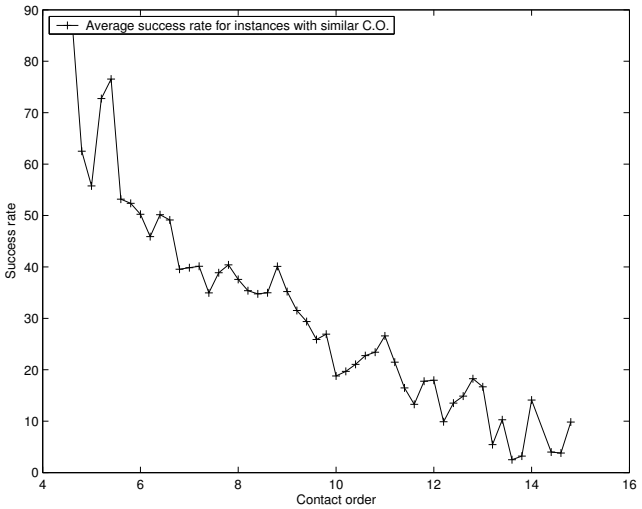


Figure 8.10: Relationship between the contact order of the sequences where MK-EDA$_2$ found the optimum at least once, and the success rate achieved by the algorithm for these instances.

The analysis of the EDA-based protein folding model has shown that, similarly to the real protein folding process, low contact order optimal conformations are easier and faster to find. This behavior has been observed in other optimization algorithms like Rosetta (188), which is one of the best algorithms for protein folding in the CASP competitions (223). In Rosetta, the structures sampled by local sequences are approximated by the distribution of structures seen for short sequences and related sequences in known protein structures. Using Rosetta, it can take high contact order proteins up to six orders of magnitude longer to fold than it takes low contact order proteins (29).

Difference in contact accessibility

In the following experiment, we investigate whether the EDA-based protein folding model is able to reproduce the difference in the rate of formation for different native contacts. Particularly, we will investigate whether the distance between contacting residues has an effect on their formation. As discussed in Section 8.6.1, local interactions are more likely to form earlier in the real folding process than non-local interactions.

Given a protein instance that is optimized using MK-EDA$_2$, the experiment consists of computing, at each generation, the frequency of each native contact of the protein from the selected set of the EDA population. The frequency is calculated as the fraction of all selected solutions that contain that contact. The EDA is executed 100 times for each of the 176 instances for which MK-EDA$_2$ had previously found 100% success.

From the information obtained from the 17600 experiments, the frequencies of the contacts with the same contact separation (C.S.) at the same generation are averaged. The results are shown in Figure 8.11. By selecting a set of instances, we intend to show that the effects observed are not particular of one single instance. On the other hand, by choosing the 176 instances for which MK-EDA$_2$ had previously found a 100% success rate, it is guaranteed that the native state will be reached in all the runs with a very high probability.

Figure 8.11 shows the evolution of the probabilities along the generations. Two aspects can be seen. First, the probability of contacts with low contact separation ($C.S. \leq 5$) is higher than the other probabilities since the first set of individuals is selected. Second, the difference in the rate of convergence determined by contact separation is evident. While contacts with low contact separation rapidly increase their probability, those contacts with a higher contact separation ($C.S. \geq 15$) grow at a slower rate.

The behavior of our model is once again consistent with what is observed in real protein folding. Moreover, these results can support hypotheses that help to explain the correlation between the contact order of proteins and the success rate and average number of generations needed to find them. As the increase of the probability of contacts with high contact separation is slower, it will take a longer time to obtain native states with higher contact order. Additionally, it will also be more difficult to find the optimum for this type of instances as the algorithm will tend to get trapped in suboptimal solutions with lower contact order.

8.8 Conclusions

The results of the experiments shown in this chapter confirm that EDAs are a feasible alternative for the protein structure prediction problem. Particularly, we recommend the

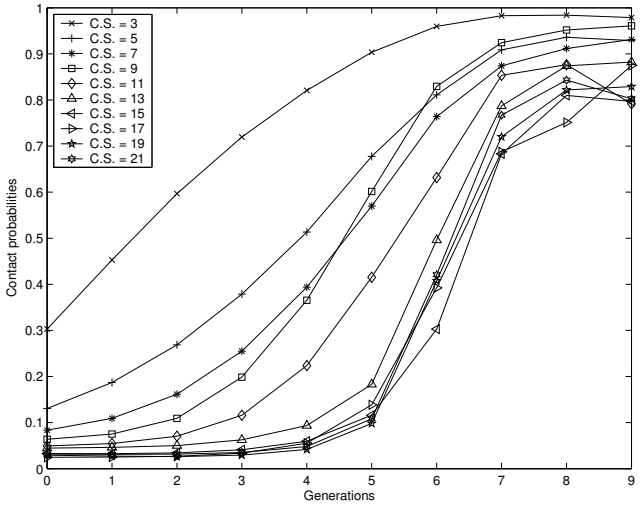


Figure 8.11: Relationship between the different contact separations and the evolution of their probabilities along the evolution of MK-EDA$_2$ for instances where it had a 100% success rate.

use of probabilistic models for the solution of coarse-grained protein folding problems where MC methods exhibit a poor performance.

The experiments that simulate the protein folding process presented in this chapter have only investigated a number of the potential issues that can be studied with the model. The analysis of EDA simulations leads to the study of other features exhibited by the model. One example is the emergence of nucleation events in protein folding.

It is generally accepted that some sort of nucleation event is a key to the protein folding mechanism. This means that some local partial structures are generally formed before the configuration of the whole protein. However, there are different ways to explain how this mechanism operates. Two opposite explanations are the "many delocalized nuclei" and the "specific nucleus" ideas. The former states that each conformation in the ensemble contains a different locally structured region. The latter suggests that the transition state ensemble consists of conformations that share the same set of essential contacts, which form a compact core inside the native state (the specific nucleus) (13).

Emergence of nucleation events in protein folding can be approached by the study of the marginal probability distributions associated with the local configurations. By adding *a priori* information about the local structure configurations to the probabilistic model, the EDA protein folding model can also be harnessed to test the "specific nucleus" or "many delocalized nuclei" hypothesis.

While the application of the model to real folding problems is clearly constrained by the

nature of the HP energy model and the specificities of EDAs, we recall that different applications of protein models require different levels of accuracy. Some models can be used to study catalytic mechanisms while other are more suitable to find functional sites by 3-d motif searching (14). Furthermore, as shown in this chapter, simple models can be analyzed in detail over a wide range of instances and parameters.

Finally, we point out that measures associated with the protein folding process, such as the contact order and the degree of frustration in proteins, are of interest for the design and study of hardness measures (e.g. fitness distance correlation, epistasis variance, etc. (163)) for evolutionary algorithms.

9 Protein side chain placement using estimation of distribution algorithms

9.1 Introduction

The problem of finding an optimal positioning for the side chain residues is called side chain placement or side chain prediction (129; 202; 222) and its discrete version is known to be NP-hard (182). The problem is important not only for homology modeling but also for protein design (183), 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 (70; 184). 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 chapter, we introduce a stochastic optimization algorithm for the solution of this problem.

The chapter is organized as follows. In the next section, the biological basis of the side chain placement problem is reviewed. Previous research on side chain prediction is discussed in Section 9.3. Section 9.4 presents the UMDA approach to side chain placement. Section 9.5 introduces the class of VNS algorithms. In this section, possible frameworks proposed for the combination of EDAs and VNS are discussed, and the EDA+VNS approach to the protein side chain placement problem is presented. In Section 9.6, numerical results of the application of the algorithm to a set of 425 proteins are presented and discussed. Section 9.7 thoroughly analyzes the relationship between the UMDA approach to side chain prediction and previous proposals to this problem. The conclusions of the chapter are outlined in Section 9.8 along with lines for further research.

9.2 Side chain placement problem

9.2.1 Rotamers and rotamer libraries

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 (70). A rotamer library is a collection of rotamers for each residue type. Rotamer libraries can be backbone-independent (220) or backbone-dependent (71). 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 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 huge in most cases. Therefore, the use of brute force algorithms would be unaffordable.

9.2.2 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 (233) van der Waals interactions, hydrogen bonds, solvation terms, and terms representing residue secondary structure propensities. Fitness functions have been proposed and tuned taking protein domain specificities into consideration.

In our representation, each residue will be associated with a variable X_i. x_i will be interpreted as the rotamer configuration associated with the i-th residue. When the backbone is fixed, the energy of a sequence folded into a defined structure can be expressed (224) as:

$$E(\mathbf{x}) = \sum_{i=1}^{n} E(x_i) + \sum_{i=1}^{n-1} \sum_{j>i}^{n} E(x_i, x_j) \tag{9.1}$$

where $E(x_i)$ is the energy interaction between the rotamer and the backbone, and $E(x_i, x_j)$, the interaction energy between the couple of rotamers. Energies are estimated using probabilities calculated from a rotamer library. The energy is usually proportional to the logarithm of the rotamer probability. Residues that do not interact at all have energy $E(x_i, x_j) = 0$ for every possible rotamer configuration.

The function represented by Equation (9.1) is used in this chapter 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 (183). 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 (224).

Therefore, some authors (133) 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.

9.3 Previous research on side chain prediction

Different optimization approaches to optimal side chain prediction have been proposed. We review a number of these state-of-the-art proposals that are relevant to our work because of the quality of the results achieved, their extent of application, and/or their relationship with our proposal. A number of the algorithms analyzed are compared with our proposal in the section of experiments. For a more complete review of these methods, see (224).

Optimization approaches to side chain prediction are usually grouped using two different classifications. They are classified into exact and approximate methods (42) according to the type of solutions found. Exact methods find the exact solution when convergence is achieved. However, there is no guarantee for them to converge in every instance. Approximate algorithms always output a candidate solution, but there is no guarantee for them to be optimal.

Another common classification (183) of optimization methods used for side chain placement divides them into stochastic and deterministic methods. For the same input parameters a deterministic method will converge to the same solution. Stochastic methods use a random component that may cause them to obtain different solutions when starting from the same input parameters. Stochastic algorithms can be divided into local and population-based search methods. Local search algorithms visit one point of the search space at each iteration. Population-based search methods use a set or population of points instead of a single point.

Figure 9.1 shows a classification of the algorithms. Exact algorithms are also deterministic. Approximate algorithms can be stochastic or deterministic.

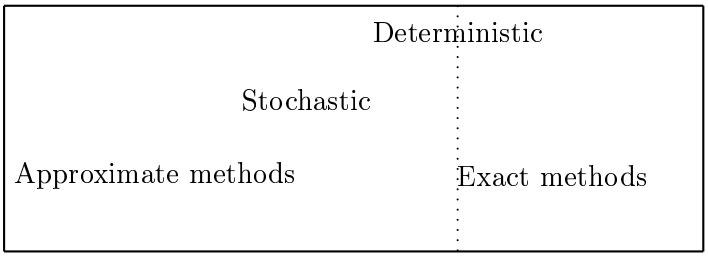


Figure 9.1: Classification of the algorithms used for side chain prediction.

9.3.1 Deterministic approaches

Among the most common deterministic approaches used for side chain prediction are dead-end elimination algorithms, the self consistent mean field approach, and SCWRL.

Dead-end elimination algorithms

,

Dead-end elimination (DEE) algorithms (64; 65; 136) belong to the class of algorithms that take advantage of the pairwise decomposability of the fitness function to eliminate rotamer configurations that been proven not to be among the optima. One of the simplest DEE implementations uses the Goldstein elimination criterion based on inequality (9.2) to iteratively eliminate rotamers.

$$E(x_i) - E(x_i') + \sum_{\substack{j=1 \\ j \neq i}}^{n} min_{x_j}(E(x_i, x_j) - E(x_i', x_j)) > 0 \tag{9.2}$$

This equation establishes a sufficient condition (64) 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. For example, the A* algorithm is usually employed to search in the remaining space. Unfortunately, this favorable scenario is not commonly found. New DEE theorems (136) have been added to the Goldstein criterion. These theorems allow the elimination of more rotamer configurations.

Self-consistent mean field approach

The self-consistent mean field (SCMF) approach uses a mean field description of the interactions between rotamers to smoothen the energy landscape. This smoothness is expected to help in the search for the global optima. The approach is based on a discrete, fixed ensemble of conformations for each subentity of the system considered. Each of these conformations is weighted by a probability which is refined by the SCMF procedure (112).

The mean-field energy for rotamer x_i at residue i is (111; 224):

$$E_{mf}(x_i) = E(x_i) + \sum_{\substack{j=1 \\ j \neq i}}^{n} \sum_{x_j} E(x_i, x_j) V(x_j) \tag{9.3}$$

The weight of each rotamer $V(x_j)$ is normalized to unity. The first term in Equation (9.3) is the contribution due to the interaction between the rotamer and the backbone, and the second term describes all the inter-rotamer pairwise interactions weighted by the probability of the rotamer. The conformational probability vector can be independently calculated by Gibb's ensemble:

$$V(x_j) = \frac{1}{q_j} e^{-\beta E_{mf}(x_j)}, \tag{9.4}$$

where q_j is the partition function and β is the inverse of the temperature.

After initialization, the mean-field potential $E_{mf}(x_i)$ is calculated from Equation (9.3) for each residue and rotamer. The energies are converted into probabilities using Gibb's equations. The algorithm iterates between Equations (9.3) and (9.4) until self-consistency is achieved.

In SCMF, a clear link between probability distributions and the objective functions defined from the rotamer energies is established. However, SCMF is a deterministic method because, given a set of input parameters, each run of the algorithm will converge to the same solution.

SCWRL

SCWRL (42; 70) is a heuristic algorithm that uses a representation of the problem based on graphs. In its initial version (70), residues were separated into inactive and active groups depending on whether or not they provably are in their minimum configuration. The combinatorial problem is reduced to find the minimum energy configuration of the active residues.

SCWRL3 is the most recent SCWRL version (42). It uses a divide and conquer strategy that is based on a graphical structure. This graph is a representation of the problem

structure. Any two residues that contain rotamers with non-zero interaction energies are considered to have an edge in the graph. First, the graph is split into clusters of interacting residues. If the cluster is small, its lowest energy is found by using a backtracking algorithm with pruning. If the cluster is too large, other partition strategies which divide the clusters into biconnected subgraphs (those that cannot be broken apart by removal of a single vertex) are used.

9.3.2 Stochastic approaches

Several stochastic approaches have been introduced for the side chain conformation problem. These include inference-based algorithms (240; 241), GAs (135; 220; 239), simulated annealing (129), and other population-based methods (78). While simulated annealing inspects a single point at each iteration, evolutionary and other population-based algorithms explore a set of solutions at each step.

Inference approach to side chain prediction

Inference-based methods (240; 241) can be used to find the exact solutions of the side chain prediction problem. This class of method comprises a set of algorithms, based on graphical models (127; 177), and is able to model the search space of the side chain prediction problem as a probability distribution defined on random variables which correspond to the residues. Each inference algorithm may solve the problem in a particular way, but they all represent it by using a probabilistic model.

Exact and approximate inference-based methods (164; 242) can be used to compute the most probable configuration. In the first case, the method is guaranteed to converge to the most probable configuration. In the second case, convergence is not guaranteed, and the solution found upon convergence may not necessarily be the optimal one. The inference-based approach to side chain prediction can be also seen as a mixed approach, where exact or approximate solutions will be found according to the variant of the inference algorithm used. If the problem is small enough, one might be able to use an exact algorithm, but in many cases only approximate algorithms can be applied. On the other hand, messages can be randomly initialized to obtain a stochastic version of the method.

In (240; 241) inference is used to find the k-lowest energy side chain configurations for a large set of protein structures. Different variants of belief propagation algorithms were evaluated in the experiments.

Evolutionary algorithms

An example of a successful application of evolutionary algorithms is presented in (135). In this paper, a combination of a GA with a mutation procedure improves the quality

of the solutions obtained using only the GA. The improvement is achieved by allowing perturbations in the optimal rotamer configurations found. The algorithm was applied to the rebuilding of 25 protein structures whose real size was less than 558 residues. Results were compared to the SCWRL algorithm showing equivalent power in rebuilding protein side chains. While the GA was relatively worse in the prediction of surface residues, it was more robust in the prediction of the buried part of the interface in the protein-protein complex.

A similar behavior was achieved by the Gaussian evolutionary method (GEM) introduced in (239). GEM combines both discrete and continuous global search mechanisms. Since many side-conformations cannot be covered if only discrete angle values are used, the inclusion of the continuous search is said to add robustness to the method (239). The algorithm is evaluated in a set of 38 proteins, with lengths between 40 and 325 residues. Results comparable to those achieved using SCWRL were obtained.

Another stochastic algorithm is presented in (78). This evolutionary algorithm is based on the elimination of variable values not likely to be present in the best configuration. At every step, the algorithm samples a set of points from the search space, evaluates them and selects two groups according to their fitness evaluation. A contrasting analysis of the frequency of rotamers between the solutions in the two groups leads to the elimination of some rotamer configurations. Those rotamers that appear only in the top high-energy conformations are excluded from future samples. This way, the algorithm constrains the search to those rotamer configurations more frequent in low energy solutions.

The process is repeated until the reduced number of variable values is smaller than a user-defined "end of stochastic stage criteria" (78). Then, an exhaustive search is applied to find the k-lowest configurations. The method was applied to 8 proteins, the largest with 263 residues. There is no guarantee for the algorithm to find the best configuration in every execution.

9.3.3 Univariate marginal distribution algorithm

We will focus on the UMDA (Algorithm 3.2), an EDA that uses a factorized probability model based on the univariate marginals calculated from the selected population.

One of the recognized approaches to the optimization of functions with multiple local minima is based on hypersurface deformation, in which the function is deliberately altered (227). The goal of different methods proposed (175; 211; 227; 250) is to smoothen the fitness landscape of the function and reduce the number of minima, thereby making the global optimization problem easier. One example of these strategies is the ant-lion method (211). This approach exploits shape modifications of the cost-function hypersurface which distend basins surrounding low-lying minima (including global minima). By

intertwining hypersurface deformations with steepest-descent displacements, the search is concentrated on a small relevant subset of all minima (211).

Among the examples of applications to the bioinformatics domain are the potential smoothing and search (PSS) algorithm (175), and the parallel hyperbolic sampling (PHS) algorithm. PSS has been applied to the prediction of the packing of transmembrane helices. PHS (250) is a Monte Carlo method that logarithmically flattens local high-energy barriers, allowing the simulation to tunnel more efficiently through energetically inaccessible regions to low-energy valleys.

Given a fitness function $f(\mathbf{x})$, UMDA transforms the original fitness landscape defined by $f(\mathbf{x})$ into a fitness landscape defined by $\widetilde{W}(p) = p(\mathbf{x})f(\mathbf{x})$, where $\widetilde{W}$ is the average fitness. This transformation smoothens the rugged fitness landscape of $f(\mathbf{x})$. UMDA converges to the local atractors of the average fitness. If there is a tendency towards the global optimum, UMDA may find it (157). Although many of the original local optima can appear flattened in the fitness landscape defined by $\widetilde{W}$, 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 (137; 162; 195).

9.4 UMDA approach to the side chain placement problem

In this section, we introduce a method to search for the optimal solution of the side chain placement problem. The pseudocode of the method is shown in Algorithm 9.1.

The algorithm starts by calculating the adjacency matrix that represents the graphical model topology inferred from the backbone structure, as described in (241). The calculation of the matrices simplifies the evaluation of the solutions by only considering 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 Dunbrack and Cohen (71). 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 derived from Equation (9.2). 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). The combination of deterministic and stochastic methods for side chain placement has been identified as an important development of these algorithms (183).

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

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. $\Omega_{X_i} = \{1, \ldots, K_i\}$, where K_i is the number of feasible rotamer configurations for residue i).

As the fitness function, the energy function in Equation (9.1) is used. The probability model in Equation (3.2) will represent the probability of a given side chain configuration.

In our implementation of the UMDA, we use a population size $M = 5000$ and truncation selection with parameter $T = 0.15$. In this type of selection, the best $N = T \cdot M$ individuals, according to their function evaluations, are selected. We use best elitism. 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 pseudocode of the algorithm is shown in Algorithm 9.1.

Algorithm 9.1: Proposed algorithm for side chain placement

1. Calculate the 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.

9.4.1 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 (42). The principal determinant of the computing 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 $O(r^3_{MEAN} n^2)$, where $r_{MEAN} = \frac{1}{n} \sum_{i=1}^{n} |K_i|$.

As can be seen from the UMDA pseudocode shown in Algorithm 3.2, 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 of 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 the 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 $Mlog(M)$.

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 (82). Let G be the maximal number of generations allowed. 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 proposal introduced is $O(r^3_{MEAN}n^2 + GM(nr_{MAX} + |D|))$.

9.5 Refinement of the solutions using variable neighborhood search

UMDA is an efficient algorithm that is able to find high quality solutions without much computational overload. However, its simple probabilistic model does not allow it to deal with problems that have strong interactions between the variables. One way to improve the results achieved by UMDA is to apply it combined with other optimization techniques. In this section, we propose the use of the variable neighborhood search (VNS) (149; 150) algorithm to search for optimal solutions starting from the solutions found by UMDA.

9.5.1 Variable neighborhood search

VNS is based on a simple principle: the systematic change of neighborhood within the search. Instead of following a trajectory, VNS explores increasingly distant neighbor-

hoods of the current solution, jumping from this solution to a new one if and only if an improvement has been made. Working in this way, favorable characteristics of the current solution will be often kept and used to obtain promising solutions. Moreover, in order to obtain local optima, a local search routine is repeatedly applied to these neighboring solutions.

More formally, let $\mathcal{N}_k, (k = 1, \ldots, k_{max})$ be a finite set of previously fixed neighborhood structures, and $\mathcal{N}_k(\mathbf{x})$, the set of solutions in the k^{th} neighborhood of $\mathbf{x}$, a possible solution of the optimization problem. The main steps of the basic VNS are presented in Algorithm 9.2.

Algorithm 9.2: Main steps of the basic VNS

1 **Initialization.** Select the set of neighborhood structures $\mathcal{N}_k$, $(k = 1, \ldots, k_{max})$
2 Find an initial solution, $\mathbf{x}$.
3 Choose a stopping condition.
4 **do** {
5 Set $k \Leftarrow 1$.
6 **do** {
7 *Shaking.* Generate a point $\mathbf{x}' \in \mathcal{N}_k(\mathbf{x})$ at random.
8 *Local search.* Apply some local search method with $\mathbf{x}'$ as the initial solution.
  Denote with $\mathbf{x}''$ the so-obtained local optimum.
9 *Move or not.* If this local optimum is better than the current solution, $\mathbf{x} \leftarrow \mathbf{x}''$.
  Otherwise, set $k \leftarrow k + 1$.
10 } **until** $k = k_{max}$.
11 } **until** Termination criterion is met.

Several stopping criteria can be used. The most common are to fix a maximum CPU time, maximum number of iterations, or maximum number of iterations between two improvements.

The basic VNS is a descent, first improvement method with randomization –notice that point $\mathbf{x}'$ is generated at random in step 7 of Algorithm 9.2. Modifications to basic VNS can be done in several ways. In the variable neighborhood descent (VND), steps 7 and 8 are replaced by the finding of the best neighbor of $\mathbf{x}$. If, in step 9, moving is done with some probability, making the selection of a solution feasible even if it is worse than the current one, the basic VNS is transformed into a descent–ascent method. Moving to the best neighborhood among all k_{max} of them, converts the basic VNS into a best improvement method. Further modifications to basic VNS as well as applications of this

metaheuristic to several interesting combinatorial optimization problems are discussed in (85). Comparisons of VNS with tabu search and GAs can be consulted in (60).

9.5.2 Hybridization between VNS and EDAs

The hybridization between VNS and EDAs can be achieved in different ways. A straightforward approach is to apply the alternatives that have been previously explored in GAs. Particularly relevant is the research done in memetic algorithms (165). This class of algorithms combines local search heuristics with crossover operators. Indeed, there are a number of EDA proposals that apply local search procedures to the solutions sampled from the probabilistic model. This is the case of the proposals presented in (95; 159; 185). A second possibility is the alternation of the search using EDAs and other schemes. In (186; 251), EDAs are used together with GAs.

Finally, the probabilistic model can be also used to design local optimization algorithms in the context of EDAs. In this case, the optimization method would take advantage of the information kept in the probability model to improve the existing solutions.

Combinations with VNS have been carried out in two different ways. A first type of hybridization consists of using another heuristic in a step of the VNS. This approach has been introduced for tabu search (114; 187) and for multistart search (18). The second proposed combination of VNS has embedded it into a given metaheuristic. For instance, in (37), VNS is combined in this way with tabu search, while (9) presents an application where VNS is embedded within a greedy adaptive search procedure.

Combination of EDAs and VNS

In (194), three main ways of combining VNS and EDAs are proposed:

- Application of VNS within EDAs.
- Use of the probabilistic model learned from the solutions in the context of VNS.
- Alternation of VNS and EDAs.

We focus on the alternatives for using VNS within EDAs. Several general ways of embedding VNS within EDAs can be obtained depending on the space where the neighborhood is taken. The simplest way is when the neighborhood is considered to be in the *space of points* of the search space and, consequently, the VNS heuristic is applied to each individual –representing a point of the search space– sampled from the probability distribution at each iteration of the EDA. This proposal can be very expensive computationally. In order to alleviate this load, it could only be applied to the best individual of the last iteration of the EDAs, as shown in Section 9.5.3.

The former approach can be extended considering the neighborhood in the *space of populations*. This idea can be considered if the best individual obtained when sampling from the probability distribution learned is worse than the current best individual. In this situation, several strategies inspired in the basic VNS can be taken into account. A first strategy is to keep the number of individuals sampled constant and to reduce the percentage of selected individuals. The objective of this strategy is to learn a probability distribution with higher density around the best individuals obtained in the simulation step. A second strategy could consist of bootstrapping from the selected individuals until the sampling of the learned probability model produces an individual with a fitness value better than the current best one. In this case, the bootstrap method provides a way to simulate a neighborhood. A third strategy is to increase the number of individuals sampled, but keeping the number of selected individuals constant. The aim of this strategy is to learn the probability distribution from a set of individuals selected with a higher selective pressure. In this case, the increase in selective pressure enables different neighborhoods to be obtained.

A different way to embedg VNS into EDAs is to define *neighborhoods in the space of probability distributions*. A simple idea is to increase the complexity of the probability distribution learned at each iteration. In case using the current individuals does not improve the current best solution.

9.5.3 EDA-VNS approach for the side chain placement problem

In this section, we introduce two different VNS schemes in the context of EDAs to solve the protein side chain placement problem.

A crucial element of the VNS algorithm is the definition of the neighborhood. The neighborhood will be defined only for the points represented by those variables and values that remained after applying the Goldstein criterion. As explained in Section 9.3.1, the Goldstein criterion allows the number of variables and the range of values for each variable to be reduced. For the side chain placement problem, we define a k-neighborhood of a solution $\mathbf{x}$ as the set of solutions that are different from solution $\mathbf{x}$ in exactly k variables. More formally,

$$\mathcal{N}_k(\mathbf{x}) = \{\mathbf{x}' \mid n - \sum_{i=1}^{n} I(x_i, x_i') = k\}, \tag{9.5}$$

where I is the indicator function, equal to one if both values are different.

Clearly, given a point $\mathbf{x}$, for all $j \neq k$, $\mathcal{N}_j(\mathbf{x}) \bigcap \mathcal{N}_k(\mathbf{x}) = \emptyset$. Additionally, we use the protein structure information to constrain the neighborhood. Particularly, we use the information contained in the adjacency matrix. In the analysis of the k-neighborhood ($k > 1$), we only consider those sets of k variables for which each pair of variable has

a non-zero entry in the adjacency matrix. This connection constraint naturally arises from the pairwise nature of the fitness function. Variables whose corresponding residues are not connected in the graph do not contribute together to the fitness function. The independent contribution of the variables to the fitness function is covered by the 1-neighborhood.

In the 1-neighborhood of $\mathbf{x}$, all the values for each of the variables are tried in the exploration of the neighborhood. For $k > 1$, the algorithm calculates all possible sets of k variables with non-zero entries in the adjacency matrix. Each possible configuration of each set that is different in all k variables to $\mathbf{x}$ defines a neighbor solution. By requiring that the solution has to be different in all the k variables, we guarantee that the neighborhoods do not overlap. On the other hand, by reducing the search of k-size sets to those interacting in the protein structure, the algorithm critically reduces the search space.

9.5.4 VNS schemes for protein side chain placement

Given this neighborhood, we define two alternative ways to define the local search step 8 of Algorithm 9.2. *Exhaustive* and *randomized* procedures can be employed.

To select $\mathbf{x}'$ in the exhaustive schema, the $\mathcal{N}_k(\mathbf{x}), (k = \{1, \ldots, k_{max}\})$ is completely searched to find one of the points where the fitness function is optimized. In the randomized scheme, point $\mathbf{x}'$ is randomly selected. A local search is conducted by randomly selecting solutions in the neighborhood and moving to this solution if fitness is improved. A parameter, *maxtries*, defines the maximum number of points of the neighborhood that will be searched.

Obviously, the exhaustive search can be more computationally costly than the randomized one. However, in the second case, the cost depends on *maxtries*. To reduce the computational cost of the exhaustive scheme, we can constrain the set of neighborhood structures.

9.6 Experiments

In this section, we present the results of the application of Algorithm 9.1 to a large set of protein instances. We also present the results of the application of the UMDA+VNS strategy. First, we introduce the protein benchmark used for our experiments. Then, we explain how the experiments were designed, as well as the numerical results of the comparison between UMDA and other optimization algorithms. Finally, the results of the UMDA+VNS approached are presented.

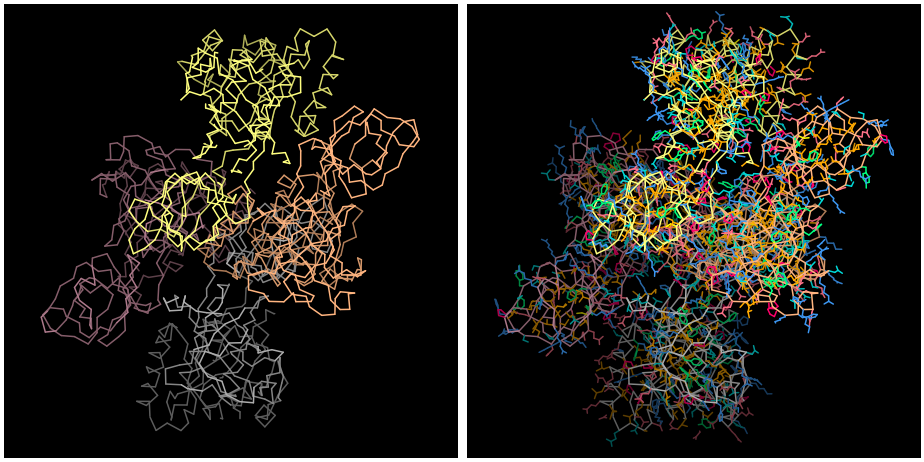


Figure 9.2: From left to right: Native backbone structure corresponding to the $pdb1d2e$ protein, side chain configuration found by UMDA.

9.6.1 Protein benchmark

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

For comparison, we have used the Side-chain PRediction INference Toolbox (SPRINT)[2], which is an implementation of the max-product belief propagation algorithm (241). To simplify the overload related to the calculation of the adjacency matrices, and to focus on the study of the UMDA, we have used the adjacency matrices available from Yanover and Weiss SPRINT implementation.

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, as well as the minimum and maximum size of the instances in each group, are shown in Table 9.1. In the case of the dimer set, each protein can contain up to five chains of residues. Figure 9.2 (left) 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

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

[2]http://www.cs.huji.ac.il/~cheny/sprint.html

for which the SPRINT algorithm converges. This information is summarized in Table 9.1 together with the number of instances of each group where UMDA is able to find the known optimal solution in at least one of 50 runs.

As can be observed in Table 9.1, 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 (241).

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

Table 9.1: Details of the protein instances.

9.6.2 Design of the experiments

Initial experiments intend to evaluate whether UMDA was able to achieve the optimum. We have excluded from the experiments the instances for which the Goldstein criterion eliminates all configurations but one. For the rest of the 314 instances, we run the UMDA and find the best solution that the algorithm can find in 50 runs. The last row of Table 9.1 shows the number of protein instances for which UMDA found the known optimal solution in at least one of the 50 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 (240), 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 chapter, we constrain the comparison to the results achieved by SPRINT.

To evaluate the performance of UMDA, we use measures PD and PE, expressed in Equations (9.6) and (9.7) respectively.

$$PD(\mathbf{x}) = \frac{\sum_{i=1}^{n} I(x_i, x_i^{opt})}{n} \tag{9.6}$$

$$PE(\mathbf{x}) = \frac{E(\mathbf{x}) - E(\mathbf{x}^{opt})}{E(\mathbf{x}^{opt})} \tag{9.7}$$

PD is the percentage, with respect to the number of side chain residues, of the number of residues different from the best known solution. In Equation (9.6), $I(x_i, x_i^{opt})$ is 1 if the side chain rotamer configurations of solutions $\mathbf{x}$ and $\mathbf{x}^{opt}$ 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 energy obtained 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 $\mathbf{x}^{best}$, found in the 50 experiments ($PD(\mathbf{x}^{best})$, $PE(\mathbf{x}^{best})$), and the average ($\overline{PD}$, $\overline{PE}$) of the evaluating measures calculated from the best solutions found in each experiment ($\overline{PD} = \frac{\sum_{j=1}^{50} PD(\mathbf{x}^j)}{50}$, $\overline{PE} = \frac{\sum_{j=1}^{50} PE(\mathbf{x}^j)}{50}$).

9.6.3 Numerical results

Figure 9.3 shows (from left to right, top to bottom) the histograms corresponding to $PD(\mathbf{x}^{best})$, $\overline{PD}$, $PE(\mathbf{x}^{best})$ and $\overline{PE}$ for the small set of instances. Similarly, Figures 9.4 and 9.5 respectively show the same measures for the large and dimer sets.

An analysis of the histograms of $PD(\mathbf{x}^{best})$ shows that the vast majority of solutions are less than 4% of the residues apart from the best known solutions. This difference increases when $\overline{PD}$ 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 values 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. This provides evidence of the emergence of a phenomenon called frustration (106).

9.6.4 Comparison with other methods

In the following experiments, we concentrate on those instances for which SPRINT did not converge. As the optimal solutions are unknown for these instances, they constitute a challenge for optimization methods. The first column of Tables 9.2, 9.3, and 9.4 shows pdb file identifiers of the proteins for which SPRINT did not converge from the set of small, large and dimer proteins respectively. Columns 2 and 3 respectively provide the remaining number (n) of residues after the application of the DEE[3] step, and the average

[3]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.

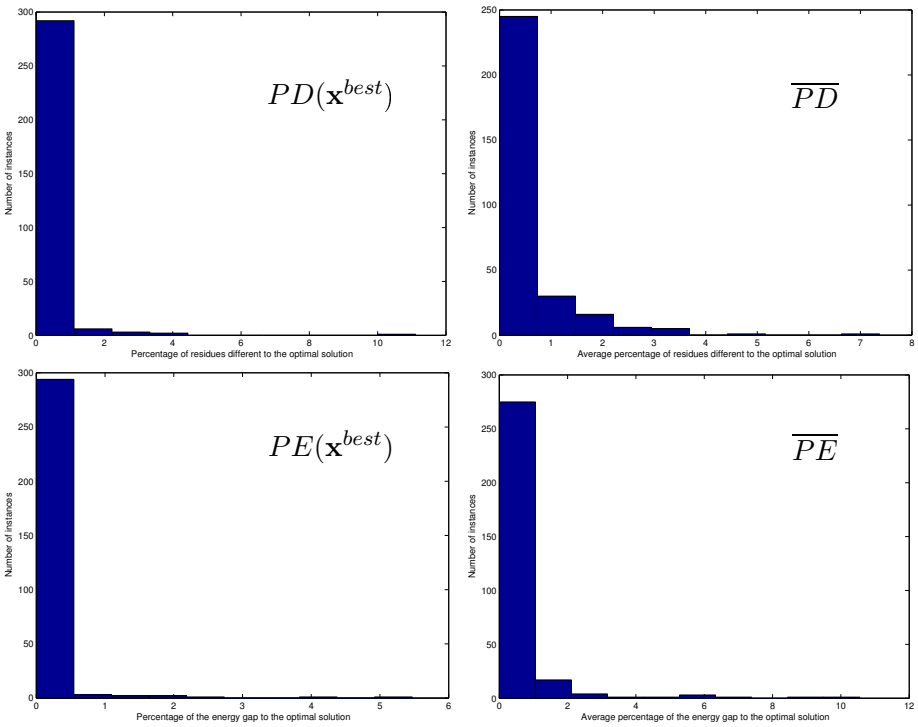


Figure 9.3: UMDA results for the small set of instances. From left to right, top to bottom, the histograms corresponding to $PD(\mathbf{x}^{best})$, $\overline{PD}$, $PE(\mathbf{x}^{best})$ and $\overline{PE}$.

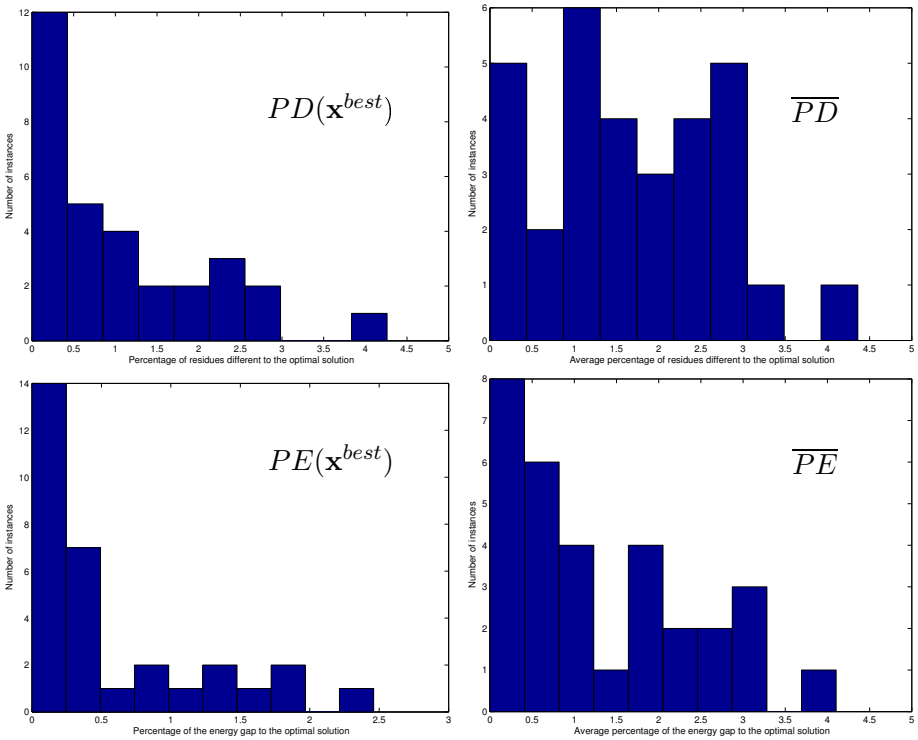


Figure 9.4: UMDA results for the large set of instances. From left to right, top to bottom, the histograms corresponding to $PD(\mathbf{x}^{best})$, $\overline{PD}$, $PE(\mathbf{x}^{best})$ and $\overline{PE}$.

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. 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 SPRINT (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 values corresponding to each instance appear underlined.

Table 9.2 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

pdb file	n	$\overline{K_i}$	f_{SPRINT}	f_{UMDA}	r_{SPRINT}	r_{UMDA}
pdb1buu	27	2.56	**200.23**	**200.23**	1.2551	1.2551
pdb1bv1	21	2.52	**132.54**	**132.54**	1.2359	1.2359
pdb1ema	45	4.35	**286.40**	**286.40**	1.0859	1.0859
pdb1et9	64	4.12	**226.80**	227.14	1.3090	1.3251
pdb1h6h	43	4.13	**97.95**	**97.95**	1.3986	1.3986
pdb1hh8	63	4.33	**369.57**	374.63	1.4601	1.5222
pdb1mrj	70	5.53	233.36	**232.66**	1.2253	1.2247
pdb2fcr	51	4.35	**236.28**	**236.28**	1.3366	1.3366
pdb2ilk	35	3.17	**138.55**	**138.55**	1.6039	1.6039
pdb2tir	28	5.28	**92.44**	92.60	1.1198	1.0833
pdb3kvt	35	4.91	**160.50**	163.95	0.8470	1.0111

Table 9.2: 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}
pdb1crz	75	3.84	628.67	**626.41**	1.2988	1.2968
pdb1ddt	146	4.22	767.04	**754.93**	1.3296	1.3105
pdb1dpe	185	4.69	914.48	**727.37**	1.2889	1.2665
pdb1e39	127	5.68	**545.30**	**545.30**	1.0239	1.0239
pdb1f5n	166	4.68	**585.78**	**585.78**	1.2480	1.2480
pdb1gsk	208	5.36	**1237.15**	**1237.15**	1.2031	1.2023
pdb1h3n	318	5.62	1635.33	**1626.09**	1.3473	1.3458
pdb1jy1	144	3.68	862.15	**861.92**	1.2760	1.2799
pdb1kmo	241	5.68	955.94	**925.90**	1.2901	1.3345
pdb1kwh	207	4.88	**959.17**	972.11	1.3838	1.4247
pdb1n5u	155	3.60	**858.89**	860.67	1.2845	1.2911
pdb1nqe	189	4.70	620.29	**570.29**	1.2556	1.3217
pdb1nr0	175	3.64	**908.02**	913.87	1.0409	1.0733
pdb2nap	292	5.26	**1100.75**	1108.79	1.2354	1.2656

Table 9.3: 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}
pdb1b25	916	5.62	4795.46	**4788.89**	1.2349	1.2628
pdb1d2e	281	3.75	2062.97	**1839.67**	1.2218	1.2087
pdb1dxr	353	4.72	2462.54	**1703.73**	1.2828	1.2683
pdb1dz4	288	5.28	974.17	**875.77**	1.1513	1.1881
pdb1e3d	454	4.23	2585.02	**2416.15**	1.1147	1.1259
pdb1e61	479	4.68	1938.85	**1936.92**	1.2375	1.2133
pdb1e6p	365	5.32	1971.44	**1681.67**	1.2812	1.3101
pdb1f60	123	4.59	543.75	**537.42**	1.2723	1.2548
pdb1fmj	294	4.59	1945.95	**1100.51**	1.2892	1.2370
pdb1fn9	239	5.77	**987.88**	989.51	1.3156	1.2304
pdb1fnn	240	4.45	765.49	**735.75**	1.2009	1.2358
pdb1giq	265	4.49	850.16	**806.53**	1.1998	1.2068
pdb1h0h	934	4.41	**4757.16**	4848.93	1.1085	1.2040
pdb1h3f	206	4.85	923.15	**785.56**	1.3351	1.3421
pdb1h4r	227	4.14	893.04	**825.63**	1.3567	1.3200
pdb1h80	229	3.88	1105.83	**1036.90**	1.1176	1.0630
pdb1hhs	728	5.97	2790.27	**2627.41**	1.3171	1.2802
pdb1iqc	288	3.88	1590.14	**1538.37**	1.2558	1.2534
pdb1j3b	289	5.06	1611.57	**1600.16**	1.3888	1.3999
pdb1j8f	329	4.10	**952.29**	957.08	1.2798	1.2932
pdb1jmx	285	4.66	1541.90	**1518.10**	1.2565	1.2891
pdb1lax	268	4.88	**924.98**	1017.56	1.1864	1.2136
pdb1lqa	244	4.00	1185.50	**935.95**	1.1911	1.2171
pdb1lsh	350	5.26	1131.90	**1125.04**	1.3524	1.2376
pdb1np7	424	5.24	**1765.15**	1783.09	1.3937	1.4025
pdb1tki	164	4.29	1083.53	**858.68**	1.2426	1.2226

Table 9.4: Results achieved by the different algorithms for the subset of dimer instances for which SPRINT does not converge.

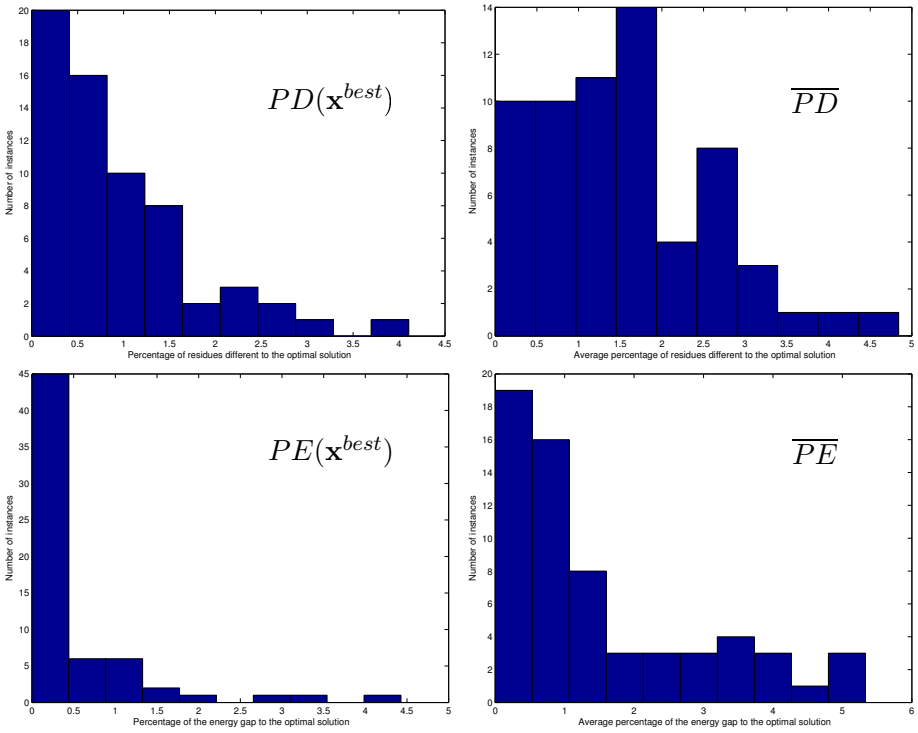


Figure 9.5: UMDA results for the dimer set of instances. From left to right, top to bottom, the histograms corresponding to $PD(\mathbf{x}^{best})$, $\overline{PD}$, $PE(\mathbf{x}^{best})$ and $\overline{PE}$.

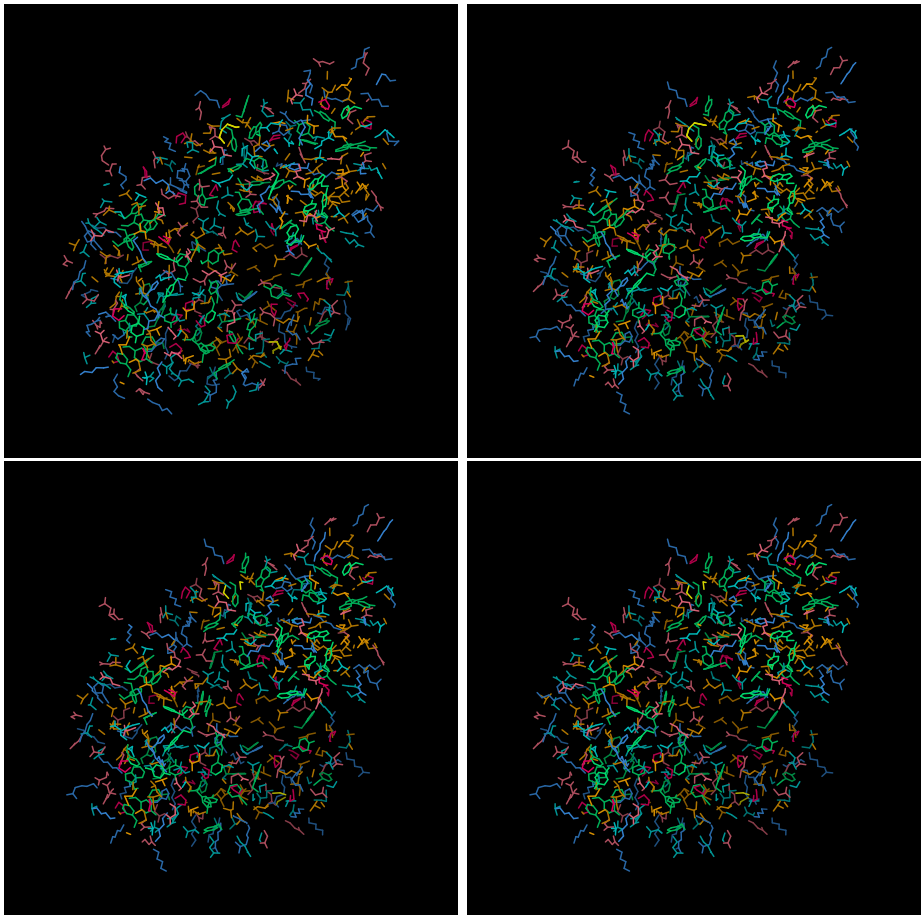


Figure 9.6: From left to right, top to down, the side chain configurations of the *pdb*1*dpe* protein: native, found by SCWRL, SPRINT, and UMDA.

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 where 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.

Figure 9.6 shows the side chain configurations found by SCWRL, SPRINT, and UMDA for the *pdb*1*dpe* protein. Notice that the structures are very similar. However, the side chain configuration found by UMDA has a better energy evaluation than the one found by SPRINT, and has a better root mean square distance to the native structure than the two other methods tested. Figure 9.2 (right) shows the side chain configurations of the *pdb*1*d*2*e* protein found by UMDA.

9.6.5 Analysis of the convergence time

Section 9.4.1 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 the average time needed by UMDA to reach convergence. Although there are other factors that influence the convergence of the algorithm, the number of residues can be useful to obtain an initial estimate of the time of convergence.

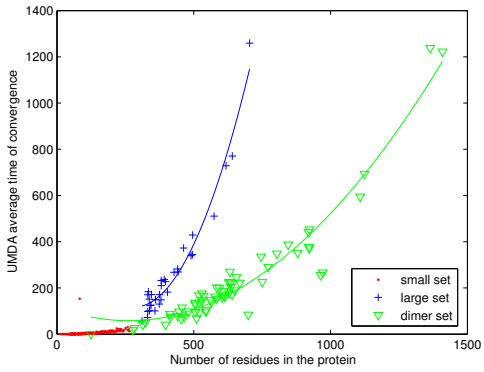


Figure 9.7: Dependence between the number of residues in all instances and the UMDA time of convergence. Only the instances where inference-based methods converged are included. Additionally, the points are fitted using second order polynomials.

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 50 experiments. Figure 9.7 plots the dependence between 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. The

polynomial that approximates data corresponding to the small, large and dimer datasets are respectively shown in Equations (9.8), (9.9), and (9.10). In these equations, x is the protein size and y, the expected time to converge.

$$y = 0.00033x^2 - 0.01182x + 0.93604 \tag{9.8}$$

$$y = 0.00586x^2 - 3.34053x + 593.83942 \tag{9.9}$$

$$y = 0.00085x^2 - 0.44991x + 117.33300 \tag{9.10}$$

An analysis of the equations shows that complexity is near quadratic in the number of variables. The equations show 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 problem can decrease 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 may be found 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.

9.6.6 Application of the UMDA+VNS approach

The objective of these experiments is to improve the solutions obtained by UMDA, and to evaluate the convenience of the VNS method. Additionally, we compare the performance of the exhaustive and randomized schemes introduced in Section 9.5.4. We use the instances for which the inference-based algorithm did not converge.

Initial experiments are conducted using the small set of instances. We apply different variants of VNS, starting from random solutions (VNS) and from the solutions found by UMDA (UMDA+VNS). The UMDA+VNS is applied to those solutions found by UMDA in the 50 experiments conducted for each instance and described in Tables 9.2, 9.3 and 9.4. On the other hand, VNS is applied on 50 randomly generated solutions for each instance.

Table 9.5 shows the results achieved when exhaustive search of the whole neighborhood is conducted. This table shows, for each algorithm, the number of times the best solution was found (S), the mean and stardard deviation ($mean \pm \sigma$) of the fitness values, and the average number of steps (s: the number of cycles of the loop between steps 4 and 11 of Algorithm 9.2 before convergence).

To evaluate the results of the algorithms, we analyze the value of the best solution found and the average fitness of the solutions. An analysis of the table shows that, in 4 of

the 11 instances, VNS achieves better results than those obtained by UMDA+VNS. In 4 instances, UMDA+VNS has better results, and in the rest of instances, the algorithms achieved equal results. VNS is able to find new best solutions in 4 instances while UMDA+VNS, in only 2. Regarding the average fitness, UMDA+VNS obtained better results in 5 of the 11 instances. For 3 instances, the algorithms achieved identical results and, in the remaining 3 instances, VNS has a lower average fitness evaluation. As expected, the average number of steps to convergence of the VNS initialized from random solutions was much higher than for UMDA+VNS.

	$f^{exhaustive,k=\{1,2,3\}}_{UMDA+VNS}$				$f^{exhaustive,k=\{1,2,3\}}_{VNS}$			
pdb id	*best*	S	$mean \pm \sigma$	s	*best*	S	$mean \pm \sigma$	s
pdb1buu	200.23	50	200.23 ± 0.00	0.78	200.23	40	200.52 ± 0.57	14.88
pdb1bv1	132.54	50	132.54 ± 0.00	0.00	132.54	50	132.54 ± 0.00	10.12
pdb1ema	286.40	49	286.41 ± 0.10	0.04	286.40	16	287.83 ± 1.20	30.32
pdb1et9	226.80	22	226.90 ± 0.09	1.22	226.80	38	226.84 ± 0.07	43.98
pdb1h6h	97.95	50	97.95 ± 0.00	0.72	97.95	50	97.95 ± 0.00	30.74
pdb1hh8	370.16	50	370.16 ± 0.00	3.06	369.57	10	370.04 ± 0.24	46.56
pdb1mrj	232.66	36	232.86 ± 0.31	2.78	232.66	38	232.99 ± 1.28	50.14
pdb2fcr	236.28	50	236.28 ± 0.00	0.00	236.28	26	242.01 ± 6.02	37.20
pdb2ilk	138.55	50	138.55 ± 0.00	0.00	138.55	29	145.74 ± 8.54	19.52
pdb2tir	92.60	50	92.60 ± 0.00	0.00	92.44	15	92.56 ± 0.08	22.26
pdb3kvt	160.50	50	160.50 ± 0.00	1.36	160.50	50	160.50 ± 0.00	29.94

Table 9.5: Results achieved by UMDA+VNS and VNS (exhaustive scheme) for the subset of small instances for which SPRINT does not converge.

In the experiments done for all the instance sets exhaustive search was only feasible for the set of small proteins. The computational cost associated with neighborhood structures $k = \{2, 3\}$ makes it impractical to use the exhaustive search of the whole neighborhood structure for these instances. Therefore, we investigate the use of reduced neighborhood structures with exhaustive search, and of randomized search in extended neighborhoods.

Table 9.6 presents the results of the comparison between VNS and UMDA+VNS using the neighborhood structure with ($k = \{1\}$). It can be observed that, considering the best solutions found, UMDA+VNS achieves better results than UMDA in only one of the instances. Considering the average fitness evaluation, UMDA+VNS outperforms VNS in all instances.

We evaluate the convenience of using the randomized scheme defined on the set of neighborhood structures $k = \{1, 2, 3\}$. Results are shown in Table 9.7. Considering the best solutions found, VNS finds better solutions than UMDA+VNS in 2 of the 11 instances, while, in the rest of instances, both algorithms found identical solutions. However, con-

	$f^{exhaustive,k=1}_{UMDA+VNS}$						$f^{exhaustive,k=1}_{VNS}$					
pdb id	*best*	S	*mean*	$\pm$	σ	s	*best*	S	*mean*	$\pm$	σ	s
pdb1buu	200.23	17	200.68	$\pm$	0.33	0.10	200.23	1	237.62	$\pm$	29.00	10.06
pdb1bv1	132.54	50	132.54	$\pm$	0.00	0.00	132.62	1	140.51	$\pm$	7.40	7.26
pdb1ema	286.40	49	286.41	$\pm$	0.10	0.04	288.70	1	333.33	$\pm$	39.66	23.88
pdb1et9	227.14	20	227.63	$\pm$	1.21	0.10	227.92	38	285.82	$\pm$	48.41	36.88
pdb1h6h	97.95	18	98.40	$\pm$	0.36	0.06	98.70	1	116.77	$\pm$	11.94	25.60
pdb1hh8	370.96	1	380.95	$\pm$	1.45	0.06	370.73	1	408.69	$\pm$	29.37	40.56
pdb1mrj	232.66	6	235.01	$\pm$	1.37	1.72	236.18	1	284.76	$\pm$	49.87	43.12
pdb2fcr	236.28	50	236.28	$\pm$	0.00	0.00	236.28	1	284.07	$\pm$	38.65	31.36
pdb2ilk	138.55	50	138.55	$\pm$	0.00	0.00	138.55	1	162.74	$\pm$	12.86	15.80
pdb2tir	92.60	50	92.60	$\pm$	0.00	0.00	92.60	6	106.98	$\pm$	20.15	19.56
pdb3kvt	163.95	41	164.03	$\pm$	0.21	0.18	163.96	2	235.16	$\pm$	65.68	21.52

Table 9.6: Results achieved by UMDA+VNS and VNS (exhaustive scheme ($k = 1$)) for the subset of small instances for which SPRINT does not converge.

sidering the average fitness of the solutions, UMDA+VNS is the best algorithm in 9 of the 11 instances.

The next experiment focuses on the application of the exhaustive scheme ($k = 1$) and the randomized scheme to the large set of instances. Results are shown in Table 9.8. Values in bold indicate that the result obtained is better than the best result known so far, either achieved by UMDA or by SPRINT as shown in Section 9.6.4. It can be noticed that UMDA+VNS improves the best results achieved by UMDA in 7 of the 14 instances. In each case, the results achieved with the randomized scheme were better than those obtained with exhaustive search ($k = 1$).

Finally, we apply the randomized scheme to the set of dimer instances for which SPRINT does not converge. For these instances, even the exhaustive scheme with reduced neighborhood was extremely expensive in terms of computational resources. The results of the experiments are displayed in Table 9.9. As in previous experiments, values in bold indicate that the result obtained is better than the best result known so far. These results confirm that UMDA+VNS can obtain state-of-the-art solutions to the side chain placement problem.

9.7 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

	$f^{randomized}_{UMDA+VNS}$						$f^{randomized}_{VNS}$					
pdb id	*best*	S	*mean*	$\pm$	σ	s	*best*	S	*mean*	$\pm$	σ	s
pdb1buu	200.24	50	200.24	$\pm$	0.00	0.76	200.24	43	200.43	$\pm$	0.50	2.92
pdb1bv1	132.54	50	132.54	$\pm$	0.00	0.00	132.54	50	132.54	$\pm$	0.00	2.46
pdb1ema	286.40	49	286.41	$\pm$	0.10	0.04	286.40	12	292.97	$\pm$	9.60	3.86
pdb1et9	226.80	21	226.95	$\pm$	0.25	1.10	226.80	30	227.10	$\pm$	0.48	3.94
pdb1h6h	97.95	32	98.20	$\pm$	0.33	0.36	97.95	45	98.06	$\pm$	0.43	3.30
pdb1hh8	370.16	1	380.91	$\pm$	1.55	0.12	369.57	6	381.78	$\pm$	12.89	3.14
pdb1mrj	232.66	14	234.16	$\pm$	1.52	1.52	232.66	3	236.80	$\pm$	4.20	4.32
pdb2fcr	236.28	50	236.28	$\pm$	0.00	0.00	236.28	10	249.93	$\pm$	11.46	4.32
pdb2ilk	138.55	50	138.55	$\pm$	0.00	0.00	138.55	32	144.72	$\pm$	8.31	2.76
pdb2tir	92.61	50	92.61	$\pm$	0.00	0.00	92.44	9	92.92	$\pm$	1.84	2.48
pdb3kvt	160.50	15	162.92	$\pm$	1.60	0.60	160.50	17	167.08	$\pm$	16.39	3.80

Table 9.7: Results achieved by UMDA+VNS and VNS (randomized scheme) for the subset of small instances for which SPRINT does not converge.

	$f^{exhaustive,k=1}_{UMDA+VNS}$						$f^{randomized}_{UMDA+VNS}$					
pdb id	*best*	S	*mean*	$\pm$	σ	s	*best*	S	*mean*	$\pm$	σ	s
pdb1crz	626.41	1	627.19	$\pm$	0.12	0.02	626.41	1	627.18	$\pm$	0.13	0.08
pdb1ddt	754.93	1	759.68	$\pm$	2.73	0.58	**753.38**	2	755.53	$\pm$	1.43	2.38
pdb1dpe	727.37	12	745.13	$\pm$	14.40	1.40	**725.50**	2	738.98	$\pm$	10.89	2.22
pdb1e39	545.30	25	545.74	$\pm$	1.23	0.06	545.30	39	545.56	$\pm$	1.07	0.36
pdb1f5n	585.78	4	595.57	$\pm$	8.09	0.12	585.78	33	589.72	$\pm$	6.29	1.34
pdb1gsk	938.97	1	942.89	$\pm$	4.90	3.28	**936.56**	15	938.77	$\pm$	3.41	3.98
pdb1h3n	1623.17	1	1634.37	$\pm$	6.82	2.78	**1620.04**	1	1626.82	$\pm$	4.30	3.78
pdb2jy1	856.84	4	859.58	$\pm$	7.24	2.52	**856.84**	4	858.41	$\pm$	1.06	1.12
pdb2kmo	917.36	1	935.28	$\pm$	7.59	4.28	**902.49**	1	917.83	$\pm$	8.38	4.60
pdb2kwh	961.21	2	973.55	$\pm$	13.36	4.30	960.73	1	972.09	$\pm$	13.01	2.78
pdb3n5u	860.67	1	876.39	$\pm$	6.92	0.84	858.89	8	870.50	$\pm$	7.95	1.98
pdb2nqe	565.36	2	587.34	$\pm$	11.43	2.66	**563.35**	2	579.29	$\pm$	11.43	2.44
pdb2nr0	913.87	13	919.69	$\pm$	11.30	1.02	912.54	24	917.04	$\pm$	9.84	2.04
pdb3nap	1101.67	1	1111.30	$\pm$	7.08	3.98	1101.21	4	1107.16	$\pm$	4.31	4.14

Table 9.8: Results achieved by UMDA+VNS (exhaustive scheme (k=1)) and UMDA+VNS (randomized scheme) for the subset of large instances for which SPRINT does not converge.

	$f^{exhaustive}_{UMDA+VNS}$					
pdb id	*best*	S	*mean*	±	σ	s
pdb1b25	**4727.90**	1	4772.87	±	21.13	12.18
pdb1d2e	**1826.88**	4	1830.49	±	3.24	5.02
pdb1dxr	**1695.16**	1	1704.84	±	6.67	4.84
pdb1dz4	**867.01**	2	874.46	±	3.50	3.92
pdb1e61	**1926.26**	5	1934.52	±	6.26	5.00
pdb1e6p	**1676.44**	1	1685.37	±	5.21	4.24
pdb1f60	**536.27**	1	539.83	±	1.84	1.44
pdb1fmj	**1088.80**	1	1098.68	±	11.67	3.32
pdb1fn9	**987.47**	2	991.60	±	2.59	1.38
pdb1fnn	**732.90**	2	737.93	±	6.19	12.34
pdb1giq	**800.92**	4	812.80	±	6.68	3.20
pdb1h0h	4768.22	1	4807.37	±	18.12	14.76
pdb1h3f	**782.97**	4	788.66	±	3.63	4.94
pdb1h4r	**815.84**	13	817.06	±	2.56	5.28
pdb1h80	**1034.77**	5	1035.31	±	0.73	3.34
pdb1hhs	**2586.92**	1	2604.39	±	10.16	9.80
pdb1j3b	**1586.47**	2	1599.79	±	9.48	3.90
pdb1j8f	**942.62**	26	943.44	±	1.07	5.40
pdb1jmx	**1514.22**	1	1534.84	±	14.35	4.04
pdb1lax	1015.88	1	1021.83	±	4.47	3.56
pdb1lqa	**926.16**	1	955.27	±	14.17	3.48
pdb1lsh	**1119.23**	1	1126.72	±	3.22	4.40
pdb1np7	**1767.72**	1	1778.87	±	8.08	4.62
pdb1tki	**855.56**	2	858.16	±	1.89	2.98

Table 9.9: Results achieved by UMDA+VNS (randomized scheme) for the subset of dimer instances for which SPRINT does not converge.

to illustrate the different aspects of the algorithm and the way in which they contribute to an efficient search. This analysis is also relevant for the identification of other possible applications of EDAs to Computational Biology.

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 (78). 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 constrast 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 drawn with any other selected set. Instead, a probability model of the solutions is constructed. Variable 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 (156).

Compared to SCMF, the strengh 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 to converge increases (224). 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 methods are two different ways to use graphical models in optimization. These approaches can be combined to obtain more efficient algorithms (96).

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.

9.8 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-based 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 solutions found. Additionally, we have presented a theoretical and empirical analysis of the computational cost of the algorithm introduced.

We have shown that, for the side chain placement problem, a hybrid approach between VNS and UMDA can improve the results achieved by using only UMDA. This is an example of problems that arise in the field of Computational Biology and that can be approached from an optimization point of view.

The chapter has reviewed some of the most used current methods for side chain prediction. We have pointed out the links between these methods 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 constrasts 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.

There are a number of branches of further research, both in the improvement of the method introduced, and in the generalization of UMDA and EDA applications to bioinformatic problems.

In our experiments, we have not considered a number of issues that may improve the efficiency of the algorithm. For instance, the population size has been fixed for all the instances. It has been shown that, for many problems, the selection of an adequate population size has an important impact on the convergence rate of EDAs. Furthermore, the use of local search optimization algorithms is a recurrent alternative to improve the best solutions found by evolutionary optimization applications. This issue should be investigated.

Protein length is not the only indicator of problem difficulty. The number of rotamers allowed at each residue is another factor that influences problem complexity. For future research, it is worth analyzing the elements that influence the complexity of the algorithm.

Koehl and Delarue (112) have identified three different directions to address the combinatorial problems that arise in computational protein design: to use empirical information about the structure, to reduce the problem dimension, and to simplify the global search for the minimum of the energy function either by non-deterministically searching approximate solutions or by modifying the potential energy function.

Some of these directions can be followed in the investigation of EDA applications to protein problems. For instance, UMDA does not take advantage of the problem information represented by the adjacency matrix. The investigation of the impact that including the interactions between the variables in the probabilistic model of the algorithm may have on the efficiency of the search is an open question. Nevertheless, it is important to take into consideration that, since the number of rotamer configurations can be very high, the inclusion of higher-order dependencies in the model will imply an important increase in the computational cost of the algorithm.

Moreover, preliminary experiments (data not included in the chapter) have shown that the graph structure can be used to split the graph into disconnected components and to independently execute UMDA in each component. The final solution is found combining the results obtained in every component. This is an approach similar to the one used by the most recent version of SCWRL. It could be carried out in a parallel scheme.

Another way to use problem domain information is to seed the initial population by sampling from the set of most probable rotamer configurations. Seeding has been shown to accelerate convergence for additive functions. Alternatively, solutions generated from the application of VNS could be used to seed the initial population.

Similarly to the way in which continuous search is employed in evolutionary algorithms to refine the final rotamer configurations (135; 239), its combination with the search in the space of discrete configurations deserves to be considered. Moreover, the search of the space of continuous rotamer configurations can be done using EDAs based on univariate Gaussian distributions (20) or EDAs that use higher-order interactions (20; 31). Continuous probabilistic models are able to represent higher-order interactions (e.g. multivariate Gaussian distributions) and need fewer parameters than their discrete counterparts.

10 Inference based methods for protein design

10.1 Introduction

In the previous two chapters, we have presented results from the application of probabilistic modeling to different computational models of the protein folding problem. In this chapter we study the application of these techniques to protein design. While protein folding treats the problem of predicting the structure of known proteins, protein design faces a related but completely different question. Protein design means the generation of novel proteins which are either compatible with existing target template structures or with arbitrarily postulated new three dimensional structural folds.

In this chapter, we treat the application of the proposals introduced in this thesis regarding the use of probabilistic model in optimization to the field of protein design. Particularly, we introduce the application of inference-based techniques to calculate the evolutionary niche of a protein structure.

The chapter is organized as follows. The next section presents a number of problems that fit under the umbrella of protein design. Section 10.3 reviews two current alternative approaches for protein design and shows that the proposal introduced in this chapter can be included in one of these approaches. Section 10.4 introduces the class of energy functions used in protein design, and describes the potential function TE13 used in our experiments. Section 10.5 introduces the problem of finding the evolutionary niche of a protein structure. The problem is reformulated in terms of searching the most probable configurations of an associated graphical model. Section 10.6 reviews a number of papers related to the proposal introduced in this chapter and discusses their relationship with our work. Section 10.7 presents numerical results of the application of the proposals discussed to a set of 3900 proteins. The conclusions of the chapter are outlined in Section 10.8 along with lines for further research.

10.2 Problems in protein design

There are two main approaches to the protein design problem (176). In *combinatorial library selection*, large numbers of sequences (libraries) are synthesized and screened for

evidence of folding to predefined structures. In *computational protein design*, computational protein models are employed to identify candidate protein sequences likely to fold to the given target structures. Another classification (151), considers the existence of two global strategies for constructing new protein sequences: Random sequence libraries and rational design. In the former, the goal is to generate random sequences that will yield proteins with specifically desired structures. In the latter, some designed procedure is applied to the construction of the sequence. In this chapter, we focus on the rational design strategy by means of computational protein design.

There are several related problems that imply different levels of protein design. One of the problems faced is the *protein redesign* (63; 136) of naturally occurring proteins. In this case, the known structure of a protein is modified in order to find a structure of lower energy.

The understanding of how the protein sequences influence its stability and structure is an important element in protein design. One of the ways this can be done is by identifying either randomized or simplified sequences that fold to the same structure (229). Sequences can be simplified by grouping the original set of amino acids in the alphabet into different classes. Among the criteria used to group the aminoa cids is their hydrophobicity level.

The *definition of potential functions* that lead the search for optimal sequences is an important research trend in this area that will be analyzed in more detail in Section 10.4. Decoy structures are possible structural conformations of a protein, usually obtained by slightly modifying the known native structure of the protein. Potential functions are essential to *distinguish decoy structures from the native sequences* (252).

In *negative design*, proteins are designed to not violate certain constraints or to not do something deleterious. For example, proteins might be designed to bind strongly to another protein to trigger the appropriate response, and not to bind to a set of other proteins to avoid triggering inappropriate reponses (200).

One way in which ideas from negative design can be applied is in the search of energy functions that do not only assign a good score to native sequences but also try to ensure that the desired sequence scores better than competing conformations. It has been stated (218) that protein design should not be possible with a simple energy-like function without explicitly considering negative design. A useful concept to assist in this search is that of the *evolutionary capacity of a protein structure* (145; 146), which is the number of protein sequences whose energy in the structure is lower than the energy of the native sequence. In Section 10.5, we analyze this and other related problems.

Other problems related to computational protein design is the *a priori determination of the potential of a protein hybrid to be functional*. Protein hybrids are proteins formed by the combination of two proteins. DNA mutagenesis and/or recombination permit the

creation of protein hybrids whose functionality is either substantially reduced or lost. In this case, protein hybrid functionality refers to their capacity to fold to a stable structure.

10.3 Directed and probabilistic protein design

In (176), protein design methods are classified into two categories: *Directed* and *probabilistic*. This classification refers to the way the search for solutions is addressed. When the space of solutions is directly explored to identify sequences with low energy, the method is *directed*. The use of site-specific amino acid probabilities rather than specific sequences is referred to as *probabilistic*.

The directed approach involves the definition of a potential or fitness function that works as a sequence-structure compatibility measure. Besides, an optimization algorithm to search the space of solutions has to be selected.

The rationale behind the use of the probabilistic approach to protein design considers the complexity and uncertainty associated with describing folded proteins. We remark that the same reason has motivated the introduction of the EDA model of protein folding for the HP model (see Chapter 8). On the other hand, the calculation of amino acid probabilities can guide the design for specific sequences and help to identify sequence regions likely to tolerate mutations (176).

10.3.1 Probabilistic methods for protein design

Residue probabilities can be used in two different ways. On one hand, they can be employed to calculate a low energy consensus sequence. i.e. the sequence comprising the most probable amino acid at each position. Clearly, as recognized by (176), the consensus sequence might not account for correlations that might arise between residues. On the other hand, they may be used to guide search algorithms, biasing the generation of trial sequences (253).

There are several methods that may be used to identify the site-specific probabilities of amino acids likely to fold to this structure. These include (176):

- Multiple sequence alignment. Similar sequences can be used to estimate the site specific probabilities as the frequencies of each amino acid at each position in the alignment.
- Directed search methods which can estimate the properties of ensembles of sequences by repeatedly applying the search to build up a set of low energy sequences.
- Entropy based methods which estimate the most probable set of probabilities by optimizing entropy functions subject to constraints.

10.3.2 Improving probabilistic methods for protein design using graphical models

Graphical models can be extensively applied to improve the efficiency of probabilistic methods in protein design. We enumerate a number of ways this improvement can be implemented.

1. Instead of an independent set of univariate probabilities, a graphical model representing the interactions between the residues can be used to find the most probable configurations. The graphical model can be learned from data (a set of aligned sequences) or defined based on available knowledge about the structure.

2. Directed search can be improved by using optimization methods that construct a model of a search space and lead the search to promising areas of the space of solutions (in this case, protein conformations of low energy). EDAs are among the natural candidates to be tried. They allow us to manipulate the complexity of the probabilitic model to be employed. Additionally, the model learned at different stages of the search can provide relevant information about the problem.

3. EDAs arise as a way to combine both directed and probabilistic search.

10.4 Empirical potential functions

Contact potentials or scoring functions measure how likely it is for a sequence to fold to a given structure. Alternatively, these potential functions can be used to distinguish native from decoy structures.

A contact potential assigns a single contact energy to two amino acids that are close to each other in the structure. Alternatively, it is set at zero if the two amino acids are not close enough.

There are two main approaches to calculate an empirical potential function (248). The first one is to use only native protein structures and apply statistical analysis to extract information important for protein stability. The second one is to use both native protein structures and decoy conformations and to apply some technique to derive a potential function that separates native structures from decoy structures.

Different techniques can be used to extract contact potentials. Perhaps the most used one is linear programming (74; 143; 217). The application of this method involves solving a very large set of inequalities to determine the parameters of the potential function. Sometimes, a feasible solution cannot be found. As a remedy, new functional forms of the potential, or the detection of an inconsistent subset of the data in the training set can be tried (143).

In the case of the identification of native structures, other methods have been proposed. These include the combination of contact interaction descriptors and local sequence-structure descriptors (248). Additionally, amino acid propensities to occur in secondary structures have been employed, by means of scoring functions, to estimate the energetic favorability of fragments of protein sequence to adopt the native conformation (205).

10.4.1 TE13 potential function

The $TE13$ potential function was introduced in (217) to correctly rate the native structure with respect to a set of decoy structures. This potential function was calculated using a linear programming approach. We introduce the $TE13$ that will be employed in our experiments.

$u_{\alpha\beta}(r)$ will denote a step potential between a pair of amino acids. The distance between the geometric centers of two amino acid side chains, r, is divided into 13 steps between 2 and 9 angstroms. The first step along r is between 2 and 3 angstroms, and the rest of the 12 steps are for half an angstrom each. Each of the $u_{\alpha\beta}(r)$ (as a function of the index β) is 1 only at one of the windows (steps) and zero elsewhere.

The total potential energy is:

$$E(\mathbf{x}, \sigma) = \sum_{\alpha,\beta} p_{\alpha\beta} n_\alpha u_{\alpha\beta}(r) \equiv \sum_{\lambda} p_\lambda n_\lambda, \tag{10.1}$$

where the index α parameterizes the type of the two interacting amino acids. n_α is the number of contacts of a specific type found when threading the complete sequence $\mathbf{x}$ into the known shape σ. n_α and $u_{\alpha\beta}$ are combined together to form n_λ, the number of contacts of a specific type and at a specific distance (λ) of structure σ. For each of the λ-s, there is a corresponding independent parameter p_λ. The total number of parameters is $((21 \times 20)/2) \times 13 = 2730$.

10.5 Evolutionary niche of protein structures as the search for the most probable configurations

Although the potential functions have been mainly used to discard decoy structures, they can also be employed to study the distribution of native-like features in sequence space. Analyzing the sequence space of proteins using tools from statistical methods is important for protein design.

In (145; 146), the sequence evolutionary selection mechanisms are analyzed focusing on the stability energy of sequences. Although the "survival probability" of a protein

sequence depends on a number of other factors such as protein function and protein flexibility, the sequence-structure relationship can be analyzed in terms of energy. The analysis assumes that native sequences were selected because they were highly probable as a function of energy. In (146), this assumption is tested by examining the energy distribution of homologous proteins.

We will denote the native sequence corresponding to the structure σ as $\mathbf{x}^\sigma$. $E_\sigma = E(\mathbf{x}^\sigma)$ is the native energy of sequence $\mathbf{x}^\sigma$ in structure σ. The quantity $N(E_\sigma)$ is the number of sequences that would have energy in σ no greater than that of the actual native sequence.

$N(E_\sigma)$ is called the *evolutionary capacity* of structure σ because it reflects how far the current state of molecular evolution σ is from the possible optimum in terms of energy (146).

Given a protein structure σ, an energy function defined on the space of amino acid sequences with cardinality 20^n, we address the problem of finding a set of sequences R_σ such that,

$$R_\sigma = \{\mathbf{x} : E(\mathbf{x}) < E(\mathbf{x}^\sigma)\} \tag{10.2}$$

We call this problem finding *the evolutionary niche* of the protein because it amounts to finding the niche of all proteins, which, regarding the energy value, are more evolved with respect to the structure. The calculation of the evolutionary niche of the protein structure adds up to determining its evolutionary capacity. However, in general, $N(E_\sigma)$ is not known in advance and can only be estimated (146).

The evolutionary niche of a structure can contain a huge amount of solutions. Therefore, we define a more stringent version of the problem. Given an input parameter k, the *restricted evolutionary niche of parameter k* is the set of sequences R_k, such that $R_k \subset R_\sigma$, $|R_k| = k$, and for all $\mathbf{x} \in R_k, \mathbf{x}' \in R_\sigma \setminus R_k$, $E(\mathbf{x}) \leq E(\mathbf{x}')$. Thus, R_k contains k of the lowest energy configurations in R_σ.

10.5.1 Reformulation of the problem in terms of probability distributions

We re-formulate the evolutionary niche problem in terms of probability distributions. A fundamental result of statistical mechanics is that, in thermal equilibrium, the probability of a state will be given Boltzmann's law:

$$p_B(\mathbf{x}) = \frac{1}{Z(T)} e^{\frac{-E(\mathbf{x})}{T}}$$

where T is the temperature of the system, and $Z(T)$ the corresponding partition function.

Clearly, the highest probability of the Boltzmann distribution corresponds to the configuration $\mathbf{x}$ with the lowest energy. The Boltzmann distribution has been successfully

used in optimization (95; 109; 160), where the idea is to sample with higher probability those areas of the search space with better fitness evaluation.

The problem of finding the evolutionary niche of a protein structure σ can be translated into the problem of finding the set of solutions A_σ.

$$A_\sigma = \{\mathbf{x} : p_B(\mathbf{x}) > p_B(\mathbf{x}^\sigma)\} \tag{10.3}$$

The drawback associated with the Boltzmann distribution is the calculation of the partition function. As the calculation of this term may be exponential, factorized approximations of the probabilities are normally used. Therefore, we will assume that the energy function is additively decomposable:

$$\begin{aligned} p_B(\mathbf{x}) &= \frac{1}{Z(T)} e^{\frac{\sum_s -E_s(\mathbf{x}_s)}{T}} \\ &= \frac{1}{Z(T)} \prod_s \Psi_s(\mathbf{x}_s) \end{aligned} \tag{10.4}$$

where $E(\mathbf{x}_s)$ is the result of evaluating the energy function on the subset of variables, and $\Psi_s(\mathbf{x}_s)$ is a potential defined on this set of variables $\mathbf{X}_s$. Decomposability of $E(\mathbf{x}_s)$ is mostly the case for the energy functions that have been proposed.

The dependence structure of $p_B(\mathbf{x})$ can be represented using a graphical structure where vertices corresponding to variables that belong to the same definition set of the additive function are joined by an edge. These variables are usually those that correspond to interacting residues in the protein structure. The graphical structure and the potentials will be used to find the set A_σ.

10.5.2 Directed and probabilistic approaches using graphical models

It has been pointed (151) that, to be successful, protein design strategies must incorporate enough diversity to cover a significant part of sequence space while simultaneously incorporating enough rational design to limit exploration to those regions of sequence space most likely to yield sequences that possess the desired qualities.

The reformulation of the evolutionary niche problem allows the optimization scheme presented in Chapter 6 to be applied. The optimization approach will look for the solutions with lowest energy. By constructing a probabilistic model of the best solutions so far visited, the idea of rational design is pursued, collecting information for the exploration of promising areas of the search space. On the other hand, sampling has a stochastic component that will allow to generate solutions from unexplored areas of the search space.

10.6 Related work

There are two main precedents to the work presented in this chapter. In (152), a second-order mean-field based approach is proposed to recognize functional protein hybrids. First, the model uses a statistical mechanics description of the residue/rotamer space of states. The state probabilities are determined by minimizing the free energy using the first-order mean approximation and the Bethe approximation whose solution is found using sum-belief propagation.

The objective of the research presented in (152) is not to find the most probable configurations. Marginal beliefs are obtained by using the tools from statistical mechanics. As an example of the application of the results obtained this way, the beliefs found after consistency are used to define a metric that helps to determine the tolerance of the protein structure to different residue combinations.

Work on the use of approximate inference for side-chain prediction (240; 241) is closely related to our proposal. As done in (240) for the side-chain prediction problem, we have re-formulated the search for the evolutionary niches of a structure as an inference problem. One of the alternatives we have followed is the application of the algorithm to find the most probable configurations as applied in (240; 241).

Rotamer libraries can be also employed for protein design (136). In such applications, each variable will encode all possible rotamer configurations for each residue. Therefore, the number of rotamer configurations can be huge. In (240; 241), the results presented were constrained to the use of inference for side-chain prediction. Therefore, no attempt was made to modify the energy function used.

There are other examples of the application of inference-based algorithms in protein problems. In (130), protein functions are predicted using message passing algorithms. This is a classification problem unrelated to protein design. Similarly, a graph-based propagation algorithm is introduced in (120) to find similarity relationships between proteins.

Regarding the use of directed search techniques for protein design, there are several examples of such applications. Relevant to our reseach is the early use of GAs (102) as an optimization method for protein design using pairwise potentials, and the application of a randomized algorithm that perform approximate enumeration of the evolutionary capacity of proteins (145). The first case shows the convenience of using population based search methods for protein design. In the second case, the search using the Monte Carlo method was effective to find an approximate solution to the problem. However, it is not clear whether a Markov chain where only one residue is changed at a time has a polynomial mixing rate. Furthermore, protein instances longer than 500 amino acids were removed from the initial set of proteins. For these instances, the counting procedure was admittedly poor.

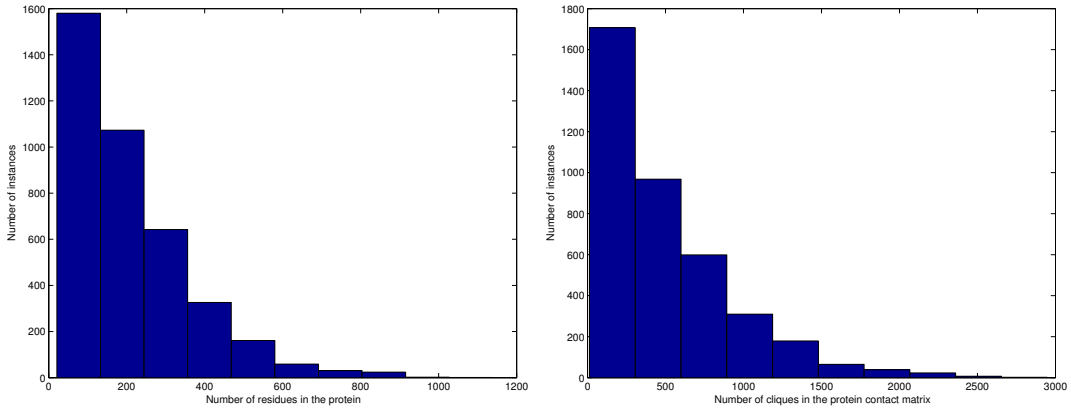


Figure 10.1: Distribution of the number of residues (left) and the number of maximal cliques (right) in the benchmark of instances used in the experiments.

10.7 Experimental results

In this section, we present the results of the application of two alternatives for the calculation of the evolutionary niche of a protein. The use of approximate inference and EDAs are evaluated on a set of 3899 protein instances. In the case of EDAs, the objective of the experiments is constrained to find the sequence with lowest energy. First, we introduce the protein benchmark and study the characteristics of the contact graph of each protein. Later, experiments illustrate the application of approximate loopy propagation and different variants of EDAs on these instances.

10.7.1 Protein benchmark

We start with an initial set of 3901 protein instances[1]. This is a reduced and non-redundant set of protein shapes that is used for fold recognition. It is a good representative of the known folds of the protein databank (145). Two protein instances have been removed because they contain one amino acid not included in the list of the 20 most common ones.

For each protein, information about the side chain geometric centers of the protein is available[2]. Since function $TE13$ is defined for the distances between the side chain centers, this distance has been calculated for each pair of residues in each protein.

[1]These instances have been obtained from Dr. Leonid Meyerguz's page: http://www.cs.cornell.edu/∼leonidm/counting/protein−list.txt

[2]This data is available from http://www.cs.cornell.edu/∼leonidm/counting/pdb−sample.tar

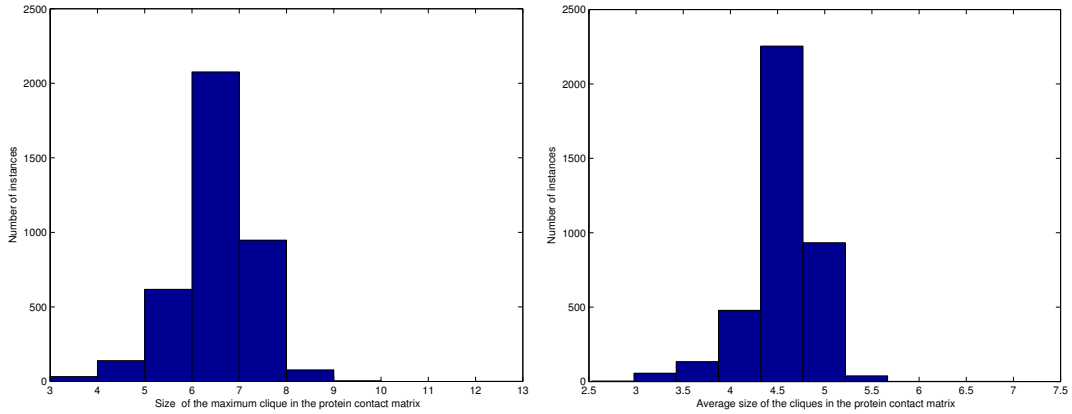


Figure 10.2: Distribution of the size of the maximum clique of the contact map (left) and of the average size (right) of maximal cliques in the benchmark of instances used in the experiments.

We construct the contact graph of each protein. In a contact graph, each vertex represents a residue. There is one edge between two vertices if the corresponding residues are in contact in the protein (for function $TE13$, contact distances are below 9 angstroms). The contact graph can be visualized using a binary contact matrix in which the entry at position i, j is one if residues i and j are in contact.

10.7.2 Analysis of the topological features of the graphs

The first experiments are oriented to evaluate the characteristics of the contact graphs associated with the proteins. For each of the 3899 instances, we calculate all the maximal cliques in the contact graph. The Bron and Kerbosch algorithm (38) has been used to calculate the cliques. Figure 10.1 shows the histograms of the number of residues (left) and the total number of maximum cliques (right) in the set of instances. As can be seen, the distribution of the sizes is not uniform. The shortest instances are more likely to be found in the set. Similarly, there are more instances with lower numbers of cliques.

As pointed out in Section 6.2.2, using the topological information of the graph, particularly the number and characteristics of the cliques, we can evaluate the complexity of using different approximations of the probability distribution associated with the energy functions. Therefore, for each protein, we have found the maximum clique and average size of all the cliques.

Figure 10.2 shows the histograms of the maximum clique size (left) and the average size of the cliques (right) for all the protein instances. The average size of the maximum clique

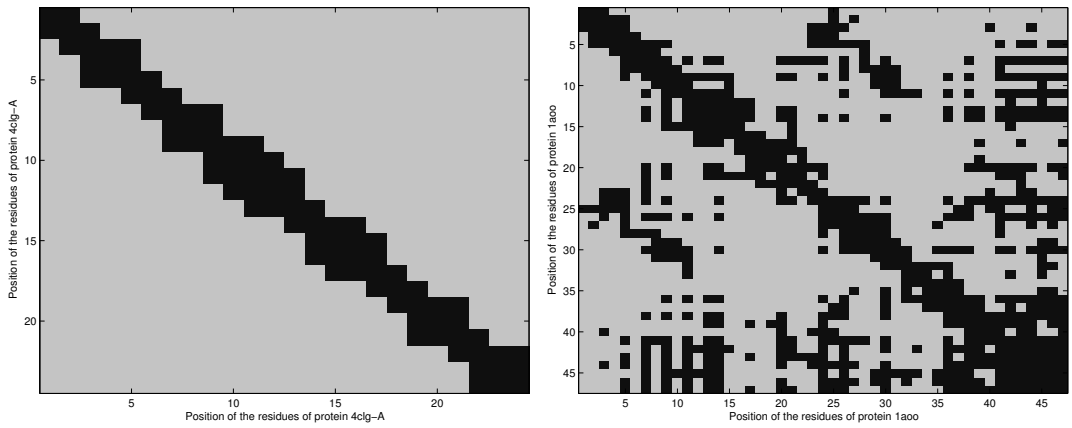


Figure 10.3: Contact matrices corresponding to two proteins for which exact inference prediction is feasible (left) and infeasible (right).

is 7.03. This means that the expected dimension of the biggest table is $\approx 20^{7.03} > 10^9$, which is clearly a prohibitive value. Since the size of the maximum clique of the graph gives a lower bound for the size of the maximum clique in the triangulated graph, we can conclude that exact inference is not expected to be feasible for the average case.

However, for certain instances exact inference is still possible. In Figure 10.3 (left), we show the contact matrix corresponding to protein 4clg-A. The maximum clique of this graph has size 3 and the graph is clearly chordal. Therefore, exact inference using belief propagation on a junction tree can be done. The opposite situation is illustrated in Figure 10.3 (right). The maximum clique of the contact graph associated with protein 1aoo has size 13 and, in this case, exact inference is infeasible. The analysis of the number of cliques and their sizes show one way in which problem information can help to choose the best approximation.

10.7.3 Evolutionary niche of proteins using BMMF and the TE13 function

We investigate the performance of BMMF to find the evolutionary niche of the proteins in our benchmark. Function $TE13$ is used to evaluate the energy of the sequences.

The experiment consists of applying BMMF to all the proteins to find the 1000 most probable configurations using the potential functions defined from $TE13$. We expect the most probable configurations to correspond to the evolutionary niche with $k = 1000$. BMMF will stop when all the configurations have been found or the loopy BP does not converge. The maximum number of steps for convergence of loopy BP was set at 200.

The results are in Table 10.1. For the instances for which BMMF finds at least one configuration, the table shows the name of the instance (pdb), its size (n), and the number of instances found (m). BMMF is able to find at least one solution in only 60 of the 3899 instances. For three intances, it is able to find the 1000 most probable configurations. The deceptive behavior of the method is due to the lack of convergence of loopy belief propagation for most of the instances. This performance is very different from the one displayed by BMMF for the rotamer problem explained in Chapter 9. Therefore, we investigate the performance of loopy BP in more detail.

pdb	n	m	pdb	n	m	pdb	n	m
1afp	51	1	1pyc	41	2	1doy	96	3
1apq	53	1	1qdp	42	42	1ehd-A	89	3
1bba	36	1	1qfn-B	25	187	1eo0-A	77	2
1bh4	30	2	1qk6-A	33	2	1fxr-A	64	2
1bkv-A	29	184	1res	43	6	1gam-A	86	4
1bqf-A	25	973	1roo	35	2	1gat-A	60	7
1btd-A	33	1	1sh1	48	4	1hd0-A	75	4
1ciq-B	24	16	1sp2	31	1000	1if1-A	105	1
1clv-I	32	9	1ter	21	1000	1imp	86	3
1dec	39	1	1zwd	35	1	1ivl-A	107	1
1dfn-A	30	1000	2mrb	31	2	1kst	68	1
1eiu	37	2	2pta	35	1	1nra	63	3
1gnf	46	2	3ins-A	21	572	1pba	81	1
1gps	47	1	3znf	30	16	1qd9-A	124	1
1iva	48	8	8tvf-A	21	5	vfy-A	67	2
1ktx	37	1	1aoj-A	60	3	1whf	86	4
1mct-I	28	1	1b7d-A	61	15	2hgf	97	1
1mea	28	75	1bbr-H	150	1	2r63	63	7
1myn	44	6	1bf0	60	7	4mt2	61	3
1pnh	31	3	1cdq	77	5	5cro-O	60	1

Table 10.1: Protein instances for which BMMF converges and finds at least one configuration.

Figure 10.4 (left) shows the evolution of the total difference between the approximation of the beliefs at different generations of loopy BP for protein 4clg-A. Equation 10.5 shows the value used for the error:

$$max_{i=1,\ldots,n:\ j=1,\ldots,20}|\hat{p}_t(x_i^j) - \hat{p}_{t-1}(x_i^j)|, \tag{10.5}$$

where $\hat{p}_t(x_i^j)$ is the approximation of the j-th marginal probability value taken by variable

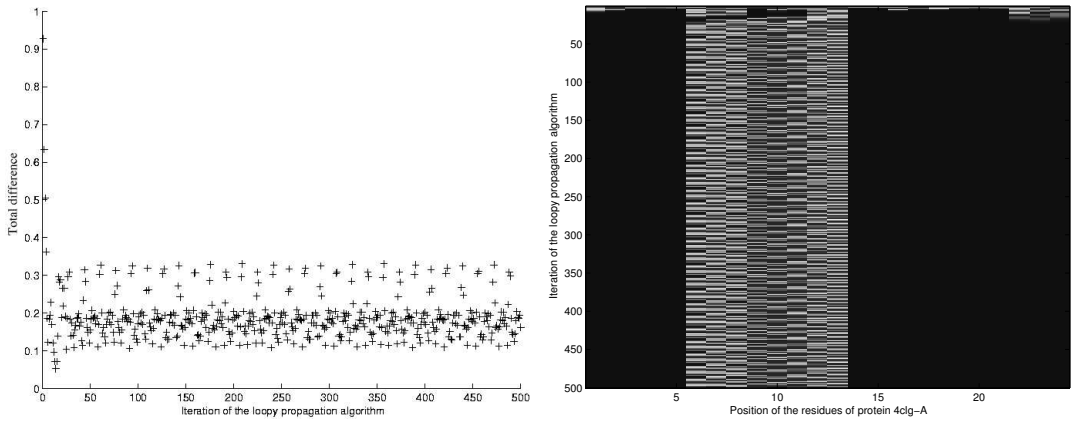


Figure 10.4: Difference in the belief approximations at different iterations of loopy BP for protein 4clg-A. Left, total difference. Right, difference for each residue.

X_i at iteration t.

For this example, the maximum number of iterations was set at 500. For the total difference, it can be observed that, after an initial drop of the values, they oscillate without converging to zero. Further analysis of the evolution of the difference can be done by observing Figure 10.4 (right). This figure reveals that there is only one subset of the marginals that do not converge (lighter colors in the figure). As explained before, for this protein the most probable configurations can be found by using exact probabilistic inference on a junction tree.

10.7.4 Evolutionary niche of proteins using EDAs and the TE13 function

The second part of the experiments has a three-fold purpose. First, we evaluate the performance of EDAs to sample the space of low energy configurations for function $TE13$. Second, we check whether the solutions found by EDAs agree with those found using BMMF. With this goal in mind, the instances for which loopy BP was able to converge, are used to evaluate EDAs.

In the following experiments, different versions of EDAs are run. All EDAs learn a tree from the data. Different are given by the way learning and sampling are done The EDAs used are the following:

1. Tree-EDA: EDA that learns a tree from the data. Sampling is done applying PLS.

2. Tree-EDA^r: EDA that learns a tree from the data, but in which only interactions between variables whose corresponding residues are in contact in the graph are considered to be included in the tree structure. Sampling is done applying PLS.

3. EDA-Loopy-P: EDA that learns a tree from the data. Sampling is done combining PLS and loopy BP. The graphical model structure for loopy BP is determined by the contact graph. Marginals are learned from data.

4. EDA-Loopy: EDA that learns a tree from the data, but in which only interactions between variables whose corresponding residues are in contact in the graph are considered to be included in the tree structure. Sampling is done combining PLS and loopy BP. The graphical model structure is used for loopy BP, and the bivariate marginals are learned from the data.

In the case of the EDAs that incorporate sampling using Algorithm 6.4, a maximum number of the 10 most probable solutions are found at each generation. The rest of the solutions are generated using the tree learned from data. EDA-Loopy-P does propagation on the full contact graph structure. EDA-Loopy does propagation on the tree (forest) learned from the data. Both algorithms learn the potentials from the bivariate marginals. The rationale behind introducing a maximum of most probable configurations is to avoid the computational cost associated with the method to find the most probable configurations.

All EDAs use truncation selection with truncation parameter $T = 0.15$. Best elitism and a maximum of 50 generations are employed. We establish the best solution found and the number of times it has been obtained in 100 experiments. The results are shown in Tables 10.2 and 10.3. The tables show the best value of the fitness found in all the experiments (f), the number of times the best solution has been found (S), the average fitness ($\bar{f}$), and the average number of generations ($\bar{g}$) needed to find the optimum. Stop criteria considered are the maximum number of generations and the lack of diversity in the selected population (there are only 10 or less different solutions).

As an initial experiment, we evaluate the tree structure learned by Tree-EDA in the first generation of the search for protein instances 4clg-A and 1aoo. 1000 runs of the algorithm are done for each instance. The tree is learned from the data and the edges in the tree are stored. We count the number of times each edge has appeared in the trees. Figure 10.5 shows the contact matrices constructed for edges that have been in at least 50 of the 1000 trees learned. The similarity between the structure learned and the original contact matrix of the graph (Figure 10.3) is remarkable. This example shows that EDAs are able to recover problem information from a statistical analysis of the data.

Table 10.2 shows a comparison between the results achieved by Tree-EDA and EDA-Loopy-P for all the instances. Similarly, a comparison between the results achieved by Tree-EDA^r and EDA-Loopy for all the instances can be seen in Table 10.3.

pdb	Tree-EDA				EDA-Loopy-P			
	$-f$	S	$-f$	$\bar{g}$	$-f$	S	$-f$	$\bar{g}$
1afp	1493.46	1	1458.46	29.00	1498.58	1	1466.51	28.00
1apq	1426.39	1	1367.53	32.00	1442.43	1	1402.05	28.00
1bba	800.31	7	775.36	25.00	800.31	13	782.85	21.08
1bh4	758.44	3	734.41	21.00	759.42	2	740.41	18.50
1bkv-A	425.23	25	411.48	16.88	425.23	48	418.46	14.60
1bqf-A	479.03	5	458.60	18.80	479.03	11	462.94	16.18
1btd-A	703.33	1	678.32	22.00	732.45	2	692.44	19.00
1ciq-B	452.90	45	449.99	16.00	452.90	62	450.67	14.35
1clv-I	926.78	1	892.70	22.00	933.72	2	904.03	20.00
1dec	947.68	1	907.02	50.00	947.59	1	919.05	50.00
1dfn-A	695.52	1	668.45	19.00	699.27	1	674.69	21.00
1eiu	907.32	6	898.07	24.50	908.19	1	900.74	26.00
1gnf	1167.75	1	1116.09	28.00	1170.04	2	1139.54	23.50
1gps	1323.88	1	1284.45	32.00	1328.38	1	1295.21	25.00
1iva	1300.78	1	1282.35	28.00	1301.56	4	1291.08	25.00
1ktx	1002.84	1	941.52	24.00	1008.66	1	951.30	22.00
1mct-I	766.62	1	745.84	19.00	771.84	2	755.05	19.00
1mea	691.47	2	664.44	19.50	691.47	4	674.63	19.00
1mhu	907.43	1	880.05	22.00	907.43	4	885.60	19.50
1myn	1289.52	1	1230.85	29.00	1301.34	1	1255.70	24.00
1pnh	687.83	1	669.68	20.00	779.23	1	680.18	28.00
1pyc	1101.69	1	1050.11	25.00	1123.14	2	1072.40	25.00
1qdp	1104.75	1	1060.80	32.00	1114.81	1	1076.30	26.00
1qfn-B	338.55	28	330.84	15.89	338.55	41	332.73	14.24
1qk6-A	816.63	1	780.12	24.00	817.42	1	792.76	21.00
1res	1038.58	1	981.14	26.00	1070.32	1	1013.49	25.00
1roo	975.74	32	969.57	21.22	980.24	1	970.27	21.00
1sh1	1405.55	1	1348.21	30.00	1442.51	1	1371.17	23.00
1sp2	673.46	1	608.36	22.00	673.46	3	640.12	18.67
1ter	528.31	4	503.82	16.00	528.31	20	507.81	14.55
1tmz-A	609.82	1	573.30	22.00	609.82	1	579.25	18.00
1zwd	658.84	1	621.14	24.00	662.34	2	637.07	19.00
2mrb	806.87	1	784.90	22.00	808.44	3	796.49	19.33
2pta	882.57	1	837.70	23.00	905.66	1	854.50	21.00
3ins-A	446.54	2	418.86	17.00	446.54	7	423.78	15.43
3znf	679.72	1	668.07	20.00	682.03	12	669.87	17.42
8tfv-A	368.23	80	367.04	13.85	368.23	83	366.91	12.73
1aoj-A	1252.12	1	1163.01	38.00	1256.80	2	1219.55	29.00
1b7d-A	1750.34	1	1666.33	40.00	1821.99	1	1713.48	30.00
1bbr-H	3615.33	1	3458.34	50.00	3900.68	1	3710.49	50.00
1bf0	1665.05	1	1596.53	34.00	1707.06	1	1629.43	33.00
1cdq	2122.12	1	2043.20	43.00	2171.36	1	2080.39	42.00
1doy	3324.07	1	3266.49	50.00	3350.96	1	3289.37	48.00
1ehd-A	3243.84	1	3159.73	50.00	3287.48	1	3210.14	46.00
1eo0-A	2242.45	1	2070.38	50.00	2349.48	1	2142.50	48.00
1fxr-A	1924.04	1	1831.44	38.00	1987.13	1	1876.89	34.00
1gam-A	2356.16	1	2266.02	47.00	2490.80	1	2353.47	48.00
1gat-A	1495.89	1	1401.05	36.00	1510.78	1	1441.40	33.00
1hd0-A	2282.86	1	2179.34	47.00	2320.12	1	2225.02	36.00
1if1-A	2984.11	1	2840.86	50.00	3041.68	1	2961.29	46.00
1imp	2478.64	1	2374.79	50.00	2505.63	1	2430.92	45.00
1ivl-A	3338.91	1	3195.89	50.00	3450.78	1	3349.20	50.00
1kst	2166.32	1	2080.83	42.00	2190.66	1	2114.19	36.00
1nra	1888.56	1	1806.72	40.00	1910.99	2	1843.35	33.50
1pba	2154.58	1	2033.58	50.00	2183.43	1	2090.56	41.00
1qd9-A	3929.16	1	3762.24	50.00	3993.59	1	3871.68	50.00
1vfy-A	1857.82	1	1797.28	40.00	1880.49	1	1828.34	34.00
1whf	2443.19	1	2293.68	50.00	2470.85	1	2382.65	43.00
2hgf	2923.62	1	2848.07	50.00	3017.59	1	2917.62	50.00
2r63	1829.48	1	1678.30	45.00	1890.86	1	1731.83	32.00
4mt2	1864.31	1	1809.06	35.00	1884.13	1	1799.89	32.00
5cro	1527.11	1	1449.72	35.00	1587.78	1	1441.56	31.00

Table 10.2: Results of the Tree-EDA and EDA-Loopy-P

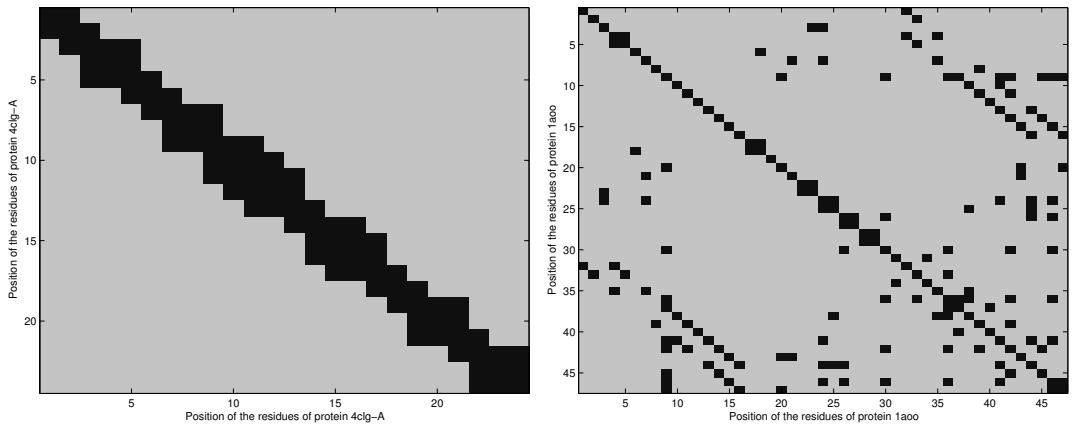


Figure 10.5: Most frequent interactions found by Tree-EDA in the first generation while finding the evolutionary niches of proteins 4clg-A and 1aoo.

An analysis of Table 10.2 shows that EDA-Loopy-P finds the same -if not- better solutions as Tree-EDA in all instances except instance 1dec for which Tree-EDA finds a solution with energy -947.68 and EDA-Loopy-P one with energy -947.59. The average energy found by EDA-Loopy-P is better in all the examples. These experiments confirm that the addition of the inference steps improves the results achieved by Tree-EDA.

Table 10.3 compares the results achieved by Tree-EDA^r and EDA-Loopy-P. In this case, the addition of the inference step does not improve the best solution found for most of the instances. Although the average of the best solution energies is higher than the one calculated from the solutions found by Tree-EDA^r, EDA-Loopy-P is able to improve the best solutions found by Tree-EDA^r for some instances. Furthermore, a comparison between Table 10.2 and Table 10.3 underlines that the use of problem information by means of constraining the search of the tree structures to the set of edges that exist in the contact graph can improve the results of the EDA. Results achieved by Tree-EDA^r are better than those achieved with Tree-EDA.

10.7.5 Approximation of the entropy

The objective of the following example is to show the way in which the most probable configurations can provide useful information for protein design. In this case, we calculate the entropy of the univariate distributions of the most probable configurations found. This analysis reveals which residues are less sensitive to mutations. There are residues in which changes do not provoke an important increase in the energy of the configuration. Figure 10.6 shows the entropy of the residues of protein 1dfn-A calculated from the 1000 most probable configurations of the Boltzmann distribution calculated from the energy

pdb	Tree-EDA^r				EDA-Loopy			
	$-f$	S	$-f$	$\bar{g}$	$-f$	S	$-f$	$\bar{g}$
1afp	1494.83	1	1468.78	27.00	1499.04	2	1467.41	29.00
1apq	1442.43	2	1405.37	27.00	1442.43	1	1406.07	27.00
1bba	800.31	20	788.67	20.80	800.31	19	784.38	20.74
1bh4	759.42	4	743.75	18.00	759.42	4	739.60	18.25
1bkv-A	425.23	56	419.71	14.89	425.23	51	418.76	14.84
1bqf-A	479.03	18	466.56	16.39	479.03	20	463.98	16.50
1btd-A	726.62	1	694.10	21.00	726.96	1	692.15	20.00
1ciq-B	452.90	62	451.01	14.60	452.90	60	451.04	14.32
1clv-I	933.72	4	913.21	19.00	933.72	1	904.66	19.00
1dec	955.11	3	924.31	50.00	955.11	2	923.32	50.00
1dfn-A	703.26	2	676.09	17.00	703.26	1	674.55	18.00
1eiu	908.19	1	903.53	22.00	907.32	18	900.90	21.17
1gnf	1172.37	1	1147.62	23.00	1172.46	1	1139.93	27.00
1gps	1329.07	1	1301.12	26.00	1328.38	3	1299.87	26.33
1iva	1301.56	3	1294.18	25.67	1301.56	3	1292.03	26.00
1ktx	1016.46	1	960.75	25.00	1015.53	1	959.02	23.00
1mct-I	771.84	2	757.00	18.50	771.84	4	754.55	17.50
1mea	691.47	7	679.30	18.43	691.47	3	678.31	18.33
1mhu	907.43	4	889.83	18.50	907.43	3	888.71	18.67
1myn	1299.16	1	1263.31	26.00	1298.22	1	1255.04	23.00
1pnh	784.83	1	691.89	21.00	776.76	1	684.77	22.00
1pyc	1123.14	1	1083.41	25.00	1123.14	1	1078.01	22.00
1qdp	1114.81	2	1083.88	22.50	1117.14	1	1080.18	25.00
1qfn-B	338.55	65	335.45	14.63	338.55	52	334.43	14.67
1qk6-A	817.42	2	797.95	20.00	817.42	1	794.58	21.00
1res	1069.92	1	1026.72	25.00	1066.21	1	1017.28	26.00
1roo	977.36	1	971.67	22.00	979.55	1	971.23	18.00
1sh1	1436.10	1	1386.09	24.00	1444.09	1	1384.37	28.00
1sp2	673.46	18	652.37	19.17	673.46	6	645.90	19.83
1ter	528.31	26	513.96	14.69	528.31	23	511.21	14.83
1tmz-A	609.82	8	589.70	18.00	609.82	5	584.88	18.00
1zwd	663.07	2	642.17	20.00	663.38	2	638.83	20.50
2mrb	808.44	4	797.63	18.00	808.44	3	793.25	18.67
2pta	903.44	1	867.30	22.00	903.44	1	858.28	21.00
3ins-A	446.54	12	428.22	14.83	446.54	9	424.92	15.33
3znf	682.03	7	670.64	18.00	682.03	6	667.79	17.50
8tfv-A	368.23	85	367.42	12.80	368.23	83	366.99	12.89
1aoj-A	1258.01	1	1226.78	28.00	1256.80	2	1220.54	30.00
1b7d-A	1822.28	1	1743.25	32.00	1808.39	1	1732.37	31.00
1bbr-H	3947.21	1	3798.36	50.00	3962.91	1	3761.01	50.00
1bf0	1711.79	1	1643.32	35.00	1696.70	1	1635.60	31.00
1cdq	2192.00	1	2100.08	38.00	2184.83	1	2091.44	39.00
1doy	3373.96	1	3325.90	48.00	3361.89	1	3309.27	50.00
1ehd-A	3282.18	1	3236.33	44.00	3283.69	1	3219.38	44.00
1eo0-A	2389.50	1	2227.81	41.00	2362.38	1	2204.66	44.00
1fxr-A	1988.22	1	1910.87	31.00	1973.55	1	1891.82	35.00
1gam-A	2465.97	1	2386.30	39.00	2460.10	1	2351.36	39.00
1gat-A	1513.75	1	1457.24	30.00	1511.62	1	1447.34	34.00
1hd0-A	2331.39	1	2252.34	38.00	2317.08	1	2239.32	34.00
1if1-A	3051.07	1	2993.47	49.00	3047.62	1	2980.32	46.00
1imp	2513.36	1	2458.20	40.00	2500.96	1	2434.47	41.00
1ivl-A	3470.71	1	3407.85	50.00	3474.05	1	3377.17	50.00
1kst	2203.51	1	2138.72	42.00	2194.82	1	2129.36	36.00
1nra	1914.91	1	1863.79	34.00	1914.21	1	1850.92	32.00
1pba	2199.66	1	2112.74	41.00	2208.69	1	2111.45	41.00
1qd9-A	4026.28	1	3952.83	50.00	4012.66	1	3914.47	50.00
1vfy-A	1912.55	1	1844.27	32.00	1896.07	1	1836.68	32.00
1whf	2487.02	1	2424.10	41.00	2480.45	1	2403.61	43.00
2hgf	3046.69	1	2960.59	50.00	3062.54	1	2940.58	47.00
2r63	1912.22	1	1802.58	38.00	1923.97	1	1778.93	38.00
4mt2	1884.13	2	1849.47	30.50	1884.13	1	1845.03	32.00
5cro	1605.88	1	1516.00	32.00	1588.82	1	1500.03	32.00

Table 10.3: Results of the Tree-EDA^r and EDA-Loopy

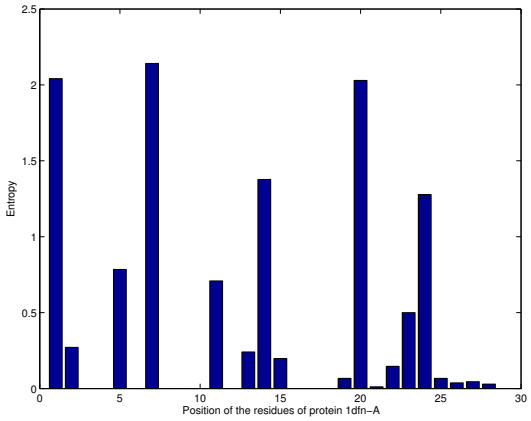


Figure 10.6: Entropy of the residues calculated from the 1000 most probable configurations of protein 1dfn-A.

function using loopy-BP. Differences in the entropy of the different residues can be clearly appreciated.

10.8 Conclusions

This chapter has studied the application of graphical models to protein design. We have started from the review of current work on probabilistic protein design. We have proposed a number of ways in which graphical models can be used to improve current applications of probabilistic modeling to problems that arise in protein design. Two main alternatives have been identified for the application of graphical models. The first one considers the use of graphical models to model the problems by associating a probability distribution with an energy function. Once the probability model has been defined, exact and approximate inference algorithms can be applied. We have shown in the chapter that the topological analysis of the contact graph can support clues about the feasibility of applying exact inference.

The second alternative implies the use of graphical models inserted in the directed search of low energy configurations. Using EDAs, graphical models participate in the modeling of the potential function and lead the search to promising areas of the search space. We have introduced a way to combine traditional learning methods employed in the context of EDAs with inference based techniques. Although the use of inference techniques have been proposed for the side chain prediction problem, we have shown that the application of these techniques goes beyond the scope of protein structure prediction. Furthermore, both inference based optimization methods and EDAs have shown different performances

in different situations. The combined used of both methods seems to be a good global strategy.

Finally, this chapter has addressed the problem of finding the evolutionary niche of a protein. We have re-formulated the problem as that of finding the most probable configurations of the protein and evaluated the convenience of inference methods and EDAs to sample the space of low energy configurations.

There are several research trends to continue the work described in this chapter. One needed step is to extend the analysis of the use of inference techniques and EDAs to other potential functions. For all the functions considered in our experiments, the score attached to the native sequence was always higher than that achieved by many other configurations. This is usually due to the existence of configurations formed by the same residues that have low energy and can be addressed by modifying the value of the energy function (145; 146).

Since our main research goal was to evaluate the performance of the algorithms to sample the space of low energy configuration disregarding the structural characteristics of these configurations, we did not pay attention to this issue. Nevertheless, the use of other potential functions could expand the results. Additionally, investigating other potential functions would allow to know whether the deceptive results of inference algorithms are due to the topology of the graph or to the particular features of the $TE13$ function.

The inference problems could be faced in two steps by using the binary patterning of polar and non-polar amino acids (230). This method first establishes a constraint to the number of residue types, and later on, a more detailed search for sequences. This approach is based on the important role played by the hydrophobicity pattern of proteins in the native structure achieved, as has been analyzed in Chapter 8. Binary patterning of polar and non-polar amino acids could be useful to organize the search for solutions by first searching for the most probable configurations in a reduced alphabet, and later conducting a more detailed search.

There are other possible uses of graphical models. They could be the basis of classification algorithms to be employed to discriminate not only between native and decoy structures, but also between native and other sequences. This could serve as an alternative approach to the use of energy potentials. Knowledge based potentials are obtained from the statistical analysis of known protein structures and/or optimization of the bias of native structures against their decoys. The limited accuracy of statistical potentials has been attributed to the fact that the procedure ignores basic physical characteristics of proteins (247).

By means of manipulating the prior probabilities (e.g. residue propensities calculated from sequences biased towards the formation of secondary structures), graphical models can incorporate different types of information to the protein design process.

Another possibility is, given a protein sequence, to obtain from the probabilistic model probable structural constraints to be satisfied by the protein structure. These constraints could be expressed in terms of the most probable pairwise distances between the residues.

One of the limitations of current models used for protein design (e.g. rotamer libraries) is that most of the models consider a fixed backbone (74). This does not allow the true flexibility that would afford more optimal sequences, and more robust predictions. Probabilistic modeling should enable a combined representation of backbone flexibility and residue variability. Indeed, this seems to be a complex task.

Part IV

Conclusions

11 Conclusions

This chapter presents general conclusions of the thesis. More specific conclusions have been presented at the end of the corresponding chapters.

Throughout the thesis, we have shown that Kikuchi approximations satisfy a number of properties that make them convenient for applications where estimation of probability distributions is required. We have shown Kikuchi that approximations can be learned from data by means of score+search techniques. Further study of possible metrics to be used for learning is needed.

The thesis has provided new evidence about the use of region based approximations in optimization by means of Kikuchi approximations of the distribution, and by means of generalized belief propagation algorithms. We have proposed a novel way to insert abductive inference in the context of estimation distribution algorithms. The approach combines learning the probabilistic model from data with the application of the max-marginal loopy propagation algorithm for sampling new solutions. The proposal has been set in the general context of possible learning and sampling alternatives for EDAs.

A number of problems from protein modeling have been addressed. We have applied different variants of EDAs to the HP protein folding problem, the protein side chain placement problem, and the problem of finding solutions from the evolutionary niche of a protein given an energy function. The results achieved show that EDAs are useful not only as optimization algorithms. They can support knowledge about the problem structure and be used to mimic certain features of the problem considered.

The thesis points out to the need to widen the scope of studies of probabilistic graphical models to analyze the suitability of region based decompositions for the estimation of probability distributions. It shows that research from computational biology can take advantage of algorithms that employ the efficient knowledge representation tools which probabilistic graphical models are.

12 Bibliography

[1] F. Abascal and A. Valencia. Automatic annotation of protein function based on family identification. *Proteins: Structure, Function and Genetics*, 53:683–692, 2003.

[2] S. Acid and L. M. de Campos. Approximations of causal networks by polytrees: An empirical study. In B. Bouchor-Meunier, R. R. Yager, and L. A. Zadeh, editors, *Advances in Intelligence Computing*, volume 945 of *Lectures Notes in Computer Science*, pages 149–158. Springer Verlag, 1995.

[3] S. M. Aji and R. J. McEliece. The generalized distributive law. *IEEE Transactions on Information Theory*, 46(2):325–343, 2000.

[4] S. M. Aji and R. J. McEliece. The generalized distributive law and free energy minimization. In *Proceedings of the 39th Allerton Conference on Communication, Control, and Computing*, Allerton, Illinois, 2001.

[5] S. M. Aji and M. Yildirim. *Mathematical Systems Theory in Biology, Communications, Computation and Finance Series: The IMA Volumes in Mathematics and its Applications, Vol. 134*, chapter Belief propagation on partially ordered sets, pages 275–300. Springer Verlag, 2003.

[6] H. Akaike. A new look at the statistical identification model. *IEEE Transactions on Automatic Control*, pages 716–723, 1974.

[7] B. Al-Lazikani, J. Jung, Z. Xiang, and B. Honig. Protein structure prediction. *Current Opinion in Chemical Biology*, 5(1):51–56, 2001.

[8] B. S. Anderson and A. W. Moore. ADtrees for fast counting and for fast learning of association rules. In *Knowledge Discovery from Databases 98*, pages 134–138, 1998.

[9] A. Andreatta and C. Ribeiro. Heuristics for the phylogeny problem. *Journal of Heuristics*, 8:429–447, 2002.

[10] P. Attard, O. G. Jepps, and S. Marcelja. Information content of signals using correlation function expansions of the entropy. *Physical Review E*, 56:4052–4067, 1997.

[11] J. H. Badsberg and F. M. Malvestuto. An implementation of the iterative proportional fitting procedure by propagation trees. *Computational Statistics and Data Analysis*, 37(3):297–322, 2003.

[12] M. Bain. *Learning Logical Exceptions in Chess*. PhD thesis, University of Strathclyde, 1994.

[13] D. Baker. A surprising simplicity to protein folding. *Nature*, 405:39–42, 2000.

[14] D. Baker. Protein structure prediction and structural genomics. *Science*, 294:93–96, 2001.

[15] S. Baluja. Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Technical Report CMU-CS-94-163, Carnegie Mellon University, Pittsburgh, PA, 1994.

[16] S. Baluja and S. Davies. Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space. In *Proceedings of the 14th International Conference on Machine Learning*, pages 30–38. Morgan Kaufmann, 1997.

[17] U. Bastolla, H. Frauenkron, E. Gerstner, P. Grassberger, and W. Nadler. Testing a new Monte Carlo algorithm for protein folding. *Proteins: Structure, Function, and Genetics*, 32:52–66, 1998.

[18] N. Belacel, P. Hansen, and N. Mladenović. Fuzzy J-means: a new heuristic for fuzzy clustering. *Pattern Recognition*, 35(10):2193–2200, 2002.

[19] E. Bengoetxea. *Inexact Graph Matching Using Estimation of Distribution Algorithms*. PhD thesis, Ecole Nationale Supérieure des Télécommunications, 2003.

[20] E. Bengoetxea, T. Miquélez, P. Larrañaga, and J. A. Lozano. *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, chapter Experimental results in function optimization with EDAs in continuous domain, pages 177–190. Kluwer Academic Publishers, Boston/Dordrecht/London, 2002.

[21] B. Berger and T. Leight. Protein folding in the hydrophobic-hydrophilic (HP) model is NP-complete. *Journal of Computational Biology*, 5(1):27–40, 1998.

[22] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. The protein data bank. *Nucleid Acid Research*, 28:235–242, 2000.

[23] J. Besag. Markov chain Monte Carlo for statistical inference. Technical Report Working paper No. 9, Center for Statistical and Social Science, University of Washington, September 2000.

[24] H. Bethe. Statistical theory of superlattices. *Proceedings of the Royal Society of London*, 150(871):552–575, 1935.

[25] J. Bilmes. Dynamic Bayesian multinets. In C. Boutilier and M. Goldszmidt, editors, *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 38–45. Morgan Kaufmann Publishers, 2000.

[26] J. Binder, D. Koller, S. Russell, and K. Kanazawa. Adaptive probabilistic networks with hidden variables. *Machine Learning*, 29(2-3):213–244, 1997.

[27] R. Blanco. *Learning Bayesian Networks from Data with Factorisation and Classification Purposes. Applications in Biomedicine*. PhD thesis, University of Basque Country, Donostia, Spain, 2005.

[28] J. Blazewicz, P. Lukasiak, and M. Milostan. Application of tabu search strategy for finding low energy structure of protein. *Artificial Intelligence in Medicine*, 35:135–145, 2005.

[29] R. Bonneau, I. Ruczinski, J. Tsai, and D. Baker. Contact order and ab initio protein structure prediction. *Protein Science*, 11:1937–1944, 2002.

[30] P. A. Bosman and D. Thierens. Linkage information processing in distribution estimation algorithms. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-1999*, volume I, pages 60–67, Orlando, FL, 1999. Morgan Kaufmann Publishers, San Francisco, CA.

[31] P. A. Bosman and D. Thierens. Expanding from discrete to continuous estimation of distribution algorithms: The IDEA. In *Parallel Problem Solving from Nature - PPSN VI 6th International Conference*, Paris, France, September 16-20 2000. Springer Verlag. LNCS 1917.

[32] P. A. Bosman and D. Thierens. Multi-objective optimization with diversity preserving mixture-based iterated density estimation evolutionary algorithms. *International Journal of Approximate Reasoning*, 31(3):259–289, 2002.

[33] P. A. Bosman and D. Thierens. Permutation optimization by iterated estimation of random keys marginal product factorizations. In J. J. Merelo, P. Adamidis, H. G. Beyer, J. L. Fernandez-Villacañas, and H. P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN VII*, volume 2439 of *Lecture Notes in Computer Science*, pages 331–340, Granada, Spain, 2002. Springer Verlag.

[34] J. M. Bower and H. Bolouri, editors. *Computational Modeling of Genetic and Biochemical Networks*. MIT Press, 2004.

[35] A. Braunstein, M. Mézard, and R. Zecchina. Survey propagation: An algorithm for satisfiability. *Random Structures and Algorithms*, 27(2):201–226, 2005.

[36] A. Braunstein and R. Zecchina. Survey and belief propagation on random k-sat. *Lecture Notes in Computer Science*, 2919:519–528, 2004.

[37] J. Brimberg, P. Hansen, N. Mladenović, and É. Taillard. Improvements and comparison of heuristics for solving the multisource weber problem. *Operations Research*, 48(3):444–460, 2000.

[38] C. Bron and J. Kerbosch. Algorithm 457—finding all cliques of an undirected graph. *Communications of the ACM*, 16(6):575–577, 1973.

[39] B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. J. States, S. Swaminathan, and M. Karplus. CHARMM: A program for macromolecular energy minimization, and dynamics calculations. *Journal of Computational Chemistry*, 4(2):187 – 217, 1983.

[40] W. Buntine. A guide to the literature on learning graphical models. *IEEE Transactions on Knowledge and Data Engineering*, 8:195–210, 1996.

[41] C. J. Butz, Q. Hu, and X. D. Yang. Critical remarks on the maximal prime decomposition of Bayesian networks. In G. Wang, Q. Liu, Y. Yao, and A. Skowron, editors, *9th International Conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing (RSFDGrC03)*, volume 2639 of *Lecture Notes in Computer Science*, pages 682–685. Springer Verlag, 2003.

[42] A. A. Canutescu, A. A. Shelenkov, and R. L. Dunbrack. A graph-theory algorithm for rapid protein side-chain prediction. *Protein Science*, 12:2001–2014, 2003.

[43] E. Castillo, J. M. Gutierrez, and A. S. Hadi. *Expert Systems and Probabilistic Network Models*. Springer-Verlag, 1997.

[44] H. Cejtin, J. Edler, A. Gottlieb, R. Helling, H. Li, J. Philbin, N. Wingreen, and C. Tang. Fast tree search for enumeration of a lattice model of protein folding. *Journal of Chemical Physics*, 116:121–144, 2002.

[45] H. S. Chan and E. Bornberg-Bauer. Perspectives on protein evolution from simple exact models. *Applied Bioinformatics*, 1(3):121–144, 2002.

[46] V. Chandru, A. Dattasharma, and V. S. A. Kumar. The algorithmics of folding proteins on lattices. *Discrete Applied Mathematics*, 127(1):145–161, 2003.

[47] L. L. Chavez, J. N. Onuchic, and C. Clementi. Quantifying the roughness on the free energy landscape: Entropic bottlenecks and protein folding rates. *Journal of American Chemical Society*, 126(27):8426–8432, 2004.

[48] D. M. Chickering, D. Geiger, and D. Heckerman. Learning Bayesian networks is NP-hard. Technical Report MSR-TR-94-17, Microsoft Research, Redmond, WA, 1994.

[49] C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467, 1968.

[50] K. J. Cios, H. Mamitsuka, T. Nagashima, and R. Tadeusiewicz. Computational intelligence in solving bioinformatics problems. *Artificial Intelligence in Medicine*, 35:1–8, 2005.

[51] G. F. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

[52] C. Cotta. Protein structure prediction using evolutionary algorithms hybridized with backtracking. In J. Mira and J. R. Alvarez, editors, *Artificial Neural Nets Problem Solving Methods*, volume 2687 of *Lecture Notes in Computer Science*, pages 321–328. Springer Verlag, 2003.

[53] T. Cover and J. Thomas. *Elements of Information Theory.* Wiley & Sons, New York, 1991.

[54] R. Cowell. Sampling without replacement in junction trees. Technical Report Statistical Research Paper 15, City University, London, 1997.

[55] P. Crescenzi, D. Goldman, C. H. Papadimitriou, A. Piccolboni, and M. Yannakakis. On the complexity of protein folding. *Journal of Computational Biology*, 5(3):423–466, 1998.

[56] C. Crick and A. Pfeffer. Loopy belief propagation as a basis for communication in sensor networks. In *Proceedings of the 19th Annual Conference on Uncertainty in Artificial Intelligence (UAI-2003)*, pages 159–166. Morgan Kaufmann Publishers, 2003.

[57] Y. Cui, W. H. Wong, E. Bornberg-Bauer, and H. S. Chan. Recombinatoric exploration of novel folded structures: A heteropolymer-based model of protein evolutionary landscapes. *Proceedings of the National Academy of Sciences*, 99(2):809–814, 2002.

[58] V. Cutello, G. Morelli, G. Nicosia, and M. Pavone. Immune algorithms with aging operators for the string folding problem and the protein folding problem. In G. R. Raidl and J. Gottlieb, editors, *Proceedings of the 5th European Conference on Computation in Combinatorial Optimization, EvoCop-2005*, volume 3448 of *Lecture Notes in Computer Science*, pages 80–90, Lausanne, Switzerland, 2005. Springer.

[59] V. Cutello, G. Nicosia, and M. Pavone. An immune algorithm with hyper-macromutations for the Dill's 2d hydrophobic-hydrophilic model. In *Proceedings of the 2004 Congress on Evolutionary Computation CEC-2004*, volume 1, pages 1074–1080, Portland, Oregon, 2004. IEEE Press.

[60] T. Davidović, P. Hansen, and N. Mladenović. Permutation–based genetic, tabu and variable neighborhood search heuristics for multiprocessor scheduling with communications delays. Technical Report G–2004–19, Les Cahiers du GERAD, 2004.

[61] A. P. Dawid. Applications of a general propagation algorithm for probabilistic expert systems. *Statistics and Computing*, (2):25–36, 1992.

[62] J. S. De Bonet, C. L. Isbell, and P. Viola. MIMIC: Finding optima by estimating probability densities. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, page 424. The MIT Press, Cambridge, 1997.

[63] W. F. De Grado, C. M. Summa, V. Pavone, F. Nastri, and A. Lombardi. De novo design and structural characterization of proteins and metalloproteins. *Annual Review of Biochemistry*, 68(1):779–819, 1999.

[64] M. De Maeyer, J. Desmet, and I. Lasters. The dead-end elimination theorem: Mathematical aspects, implementation, optimization, evaluation, and performance. *Methods in Molecular Biology*, 143:265–304, 2000.

[65] J. Desmet, M. De Maeyer, B. Hazes, and I. Lasters. The dead-end elimination theorem and its use in protein side-chain positioning. *Nature*, 356:539–542, 1992.

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

[67] M. Dorigo and T. Stützle. *Ant Colony Optimization*. MIT press, 2004.

[68] S. Droste, T. Jansen, and I. Wegener. Perhaps not a free lunch but at least a free appetizer. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-1999*, pages 833–839, San Francisco, CA, 1999. Morgan Kaufmann Publishers, Inc.

[69] S. Duarte-Flores and J. E. Smith. Study of fitness landscapes for the HP model of protein structure prediction. In R. Sarker, R. Reynolds, H. Abbass, K. C. Tan, B. McKay, D. Essam, and T. Gedeon, editors, *Proceedings of the 2003 Congress on Evolutionary Computation CEC-2003*, pages 2338–2345. IEEE Press, 2003.

[70] R. L. Dunbrack. Rotamer libraries in the 21st century. *Current Opinion in Structural Biology*, 12:431–440, 2002.

[71] R. L. Dunbrack and F. E. Cohen. Bayesian statistical analysis of protein side-chain rotamer preferences. *Protein Science*, 6(8):1661–1681, 1997.

[72] R. Etxeberria and P. Larrañaga. Global optimization using Bayesian networks. In *Proceedings of the Second Symposium on Artificial Intelligence (CIMAF-99)*, pages 151–173, Habana, Cuba, March 1999.

[73] J. D. Farmer, N. H. Packard, and A. S. Perelson. The immune system, adaptation and machine learning. *Physica D*, 22:187–204, 1986.

[74] C. A. Floudas, J. L. Klepeis, J. D. Lambris, and D. Morikis. De novo protein design: An interplay of global optimization, mixed-integer optimization, and experiments. In *Proceedings of Foundations of Computer Aided Process Design 2004*, pages 133–146, 2004.

[75] H. Frauenkron, U. Bastolla, E. Gerstner, P. Grassberger, and W. Nadler. New Monte Carlo algorithm for protein folding. *Physical Review Letters*, 80(4):3149–3153, 1998.

[76] N. Friedman and M. Goldszmidt. Building classifiers using Bayesian networks. In *AAAI/IAAI, Vol. 2*, pages 1277–1284, 1996.

[77] Y. Gao and J. C. Culberson. Space complexity of estimation of distribution algorithms. *Evolutionary Computation*, 13(1):125–143, 2005.

[78] M. Glick, A. Rayan, and A. Goldblum. A stochastic algorithm for global optimization for best populations: A test case of side chains in proteins. *Proceedings of the National Academy of Sciences*, 99(2):703–708, 2002.

[79] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 5:533–549, 1986.

[80] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, MA, 1989.

[81] C. González, J. A. Lozano, and P. Larrañaga. Analyzing the PBIL algorithm by means of discrete dynamical systems. *Complex Systems*, 12(4):465–479, 2001.

[82] C. González, J. A. Lozano, and P. Larrañaga. Mathematical modeling of UMDAc algorithm with tournament selection. Behaviour on linear and quadratic functions. *International Journal of Approximate Reasoning*, 31(4):313–340, 2002.

[83] P. Grassberger. Sequential Monte Carlo methods for protein folding. In D. Wolf, G. Münster, and M. Kremer, editors, *Proceedings of the NIC Symposium 2004*, volume 20 of *NIC series*, pages 1–10. John von Neumann-Institut für Computing (NIC), 2004.

[84] V. D. Grouba, A. V. Zorin, and I. A. Sevastianov. The superposition approximation: A critical review. *International Journal of Modern Physics B*, 18(1):1–44, 2004.

[85] P. Hansen and N. Mladenović. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130:449–467, 2001.

[86] U. H. E. Hansmann and Y. Okamoto. New Monte Carlo algorithms for protein folding. *Current Opinion in Structural Biology*, 9:177–181, 1999.

[87] G. Harik. Linkage learning via probabilistic modeling in the EcGA. IlliGAL Report 99010, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL, 1999.

[88] G. R. Harik, F. G. Lobo, and D. E. Goldberg. The compact genetic algorithm. *IEEE Transactions on Evolutionary Computation*, 3(4):287–297, 1999.

[89] W. E. Hart and S. C. Istrail. Fast protein folding in the hydrophobic-hydrophilic model within three-eights of optimal. *Journal of Computational Biology*, 3(1):53–96, 1996.

[90] D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243, 1995.

[91] M. Henrion. Propagating uncertainty in Bayesian networks by probabilistic logic sampling. *Uncertainty in Artificial Intelligence*, 2:317–324, 1988.

[92] T. Heskes. Stable fixed points of loopy belief propagation are minima of the Bethe free energy. In *Advances in Neural Information Processing Systems, (NIPS-2002)*, volume 14, pages 343–350. MIT Press, 2003.

[93] J. D. Hirst. The evolutionary landscape of functional model proteins. *Protein Engineering*, 12:721–726, 1999.

[94] J. H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, MI, 1975.

[95] R. Höns. *Estimation of Distribution Algorithms and Minimum Relative Entropy.* PhD thesis, University of Bonn, Bonn, Germany, 2006.

[96] R. Höns, R. Santana, P. Larrañaga, and J. A. Lozano. Optimization by max-propagation using Kikuchi approximations. Submitted for publication, 2006.

[97] M. D. T. Hoque, M. Chetty, and L. S. Dooley. A new guided genetic algorithm for 2d hydrophobic-hydrophilic model to predict protein folding. In *Proceedings of the 2005 Congress on Evolutionary Computation CEC-2005*, pages 259–266, Edinburgh, U.K., 2005. IEEE Press.

[98] H.-P. Hsu, V. Mehra, and P. Grassberger. Structure optimization in an off-lattice protein model. *Physical Review E*, 68(2):4 pages, 2003.

[99] H.-P. Hsu, V. Mehra, W. Nadler, and P. Grassberger. Growth algorithms for lattice heteropolymers at low temperatures. *Journal of Chemical Physics*, 118(1):444–451, 2003.

[100] A. Jakulin and I. Bratko. Testing the significance of attribute interactions. In *Proceedings of the 21th Conference on Machine Learning (ICML-2004)*, pages 409–416, Banff, Canada, 2004. ACM Press.

[101] A. Jakulin, I. Rish, and I. Bratko. Kikuchi-Bayes: Factorized models for approximate classification in closed form. Technical Report RC23314 (WO408-175), IBM, August 2004.

[102] D. T. Jones. De *novo* protein design using pairwise potentials and a genetic algorithm. *Protein Science*, 3:567–574, 1994.

[103] G. Jones, P. Willett, and R. C. Glen. Molecular recognition of receptor sites using a genetic algorithm with a description of desolvation. *Journal of Molecular Biology*, 245(1):43–53, 1995.

[104] G. Jones, P. Willett, R. C. Glen, A. R. L. Leach, and R. Taylor. Development and validation of a genetic algorithm for flexible docking. *Journal of Molecular Biology*, 267(3):727–748, 1997.

[105] W. Just. Computational complexity of multiple sequence alignment with sp-score. *Journal of Computational Biology*, 8(6):615–623, 2001.

[106] L. Kallel, B. Naudts, and R. Reeves. Properties of fitness functions and search landscapes. In L. Kallel, B. Naudts, and A. Rogers, editors, *Theoretical Aspects of Evolutionary Computing*, pages 177–208. Springer Verlag, 2000.

[107] M. Khimasia and P. Coveney. Protein structure prediction as a hard optimization problem: The genetic algorithm approach. *Molecular Simulation*, 19:205–226, 1997.

[108] R. Kikuchi. A theory of cooperative phenomena. *Physical Review*, 81(6):988–1003, 1951.

[109] S. Kirkpatrick, C. D. J. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, May 1983.

[110] J. G. Kirkwood and E. M. Boggs. The radial distribution function in liquids. *Journal of Chemical Physics*, 10:394–402, 1942.

[111] P. Koehl and M. Delarue. Application of a self-consistent mean field theory to predict protein side-chains conformation and estimate their conformational entropy. *Journal of Molecular Biology*, 239:249–275, 1994.

[112] P. Koehl and M. Delarue. Building protein lattice models using self consistent mean field theory. *Journal of Chemical Physics*, 108:9540–9549, 1998.

[113] R. König and T. Dandekar. Improving genetic algorithms for protein folding simulations by systematic crossover. *Biosystems*, 50:17–25, 1999.

[114] V. Kovačević, M. Čangalović, M. Ašić, D. Dražić, and L. Ivanović. Tabu search methodology in global optimization. *Computers and Mathematics with Applications*, 37(125–133), 1999.

[115] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* The MIT Press, Cambridge, MA, 1992.

[116] C. M. Kraemer-Pecore, A. M. Wollacott, and J. R. Desjarlais. Computational protein design. *Current Opinion in Chemical Biology*, 5:690–695, 2001.

[117] N. Krasnogor, B. Blackburne, E. K. Burke, and J. D. Hirst. Algorithms for protein structure prediction. In J. J. Merelo, P. Adamidis, H. G. Beyer, J. L. Fernandez-Villacañas, and H. P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN VII*, volume 2439 of *Lecture Notes in Computer Science*, pages 769–778, Granada, Spain, 2002. Springer Verlag.

[118] N. Krasnogor, W. E. Hart, J. Smith, and D. A. Pelta. Protein structure prediction with evolutionary algorithms. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 2, pages 1596–1601, Orlando, Florida, USA, 1999. Morgan Kaufmann.

[119] F. R. Kschischang, B. J. Frey, and H. A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.

[120] R. Kuang, J. Weston, W. S. Noble, and C. Leslie. Motif-based protein ranking by network propagation. *Bioinformatics*, 21(19):3711 – 3718, 2005.

[121] G. B. Lamont and L. D. Merkle. Toward effective polypeptide structure prediction with parallel fast messy genetic algorithms. In G. B. Fogel and D. W. Corne, editors, *Evolutionary Computation in Bioinformatics*, pages 137–162. Morgan Kaufmann, 2002.

[122] P. Larrañaga. *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, chapter An introduction to probabilistic graphical models, pages 25–54. Kluwer Academic Publishers, Boston/Dordrecht/London, 2002.

[123] P. Larrañaga, B. Calvo, R. Santana, C. Bielza, J. Galdiano, I. Inza, J. A. Lozano, R. Armañanzas, G. Santafé, A. Pérez, and V. Robles. Machine learning in bioinformatics. *Briefings in Bioinformatics*, 2006. In press.

[124] P. Larrañaga, R. Etxeberria, J. A. Lozano, and J. M. Peña. Optimization by learning and simulation of Bayesian and Gaussian networks. Technical Report EHU-KZAA-IK-4/99, Department of Computer Science and Artificial Intelligence, University of the Basque Country, December 1999.

[125] P. Larrañaga and J. A. Lozano, editors. *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, Boston/Dordrecht/London, 2002.

[126] S. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems (with discussion). *Journal of the Royal Statistical Society, Series B*, 50:157–224, 1988.

[127] S. L. Lauritzen. *Graphical Models*. Oxford:Clarendon Press, 1996.

[128] G. A. Lazar, J. R. Desjarlais, and T. M. Handel. De *novo* protein design of the hydrophobic core of ubiquitin. *Protein Science*, 6:1167–1178, 1997.

[129] C. Lee and S. Subbiah. Prediction of protein side-chain conformation by packing optimization. *Journal of Molecular Biology*, 217:373–388, 1991.

[130] M. Leone and A. Pagnani. Predicting protein functions with message passing algorithms. *Bioinformatics*, 21(2):239–247, 2005.

[131] N. Lesh, M. Mitzenmacher, and S. Whitesides. A complete and effective move set for simplified protein folding. In *Proceedings of the Seventh Annual International Conference on Research in Computational Molecular Biology*, pages 188 – 195, 2003.

[132] N. Lesh, M. Mitzenmacher, and S. Whitesides. A complete and effective move set for simplified protein folding. Technical Report TR-2003-03, Mitsubishi Electric Research Laboratories, February 2003.

[133] S. Liang and N. V. Grishin. Side-chain modeling with an optimized scoring function. *Protein Science*, 11:322–331, 2002.

[134] S. Liang and W. H. Wong. Evolutionary Monte Carlo for protein folding simulation. *Journal of Chemical Physics*, 115:3374–3380, 2001.

[135] Z. Liu, W. Li, S. Liang, Y. Han, and L. Lai. Beyond rotamer library: Genetic algorithm combined with disturbing mutation process for upbuilding protein side-chains. *Proteins: Structure, Function, and Genetics*, 50:49–62, 2003.

[136] L. L. Looger and H. W. Hellinga. Generalized dead-end elimination algorithms make large-scale protein side-chain structure prediction tractable: Implications for protein design and structural genomics. *Journal of Molecular Biology*, 307(1):429–445, 2001.

[137] L. Lozada and R. Santana. UMDA dynamics for a class of parametric functions. Technical Report ICIMAF 2003-239, Institute of Cybernetics, Mathematics and Physics, Havana, Cuba, September 2003.

[138] J. A. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea, editors. *Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms.* Springer-Verlag, 2006.

[139] R. M. MacCallum. Striped sheets and protein contact prediction. *Bioinformatics*, 20(8):i224 – i231, 2004.

[140] D. Mackay. *Learning in Graphical Models*, chapter Introduction to Monte Carlo methods, pages 175–204. MIT Press, 1998.

[141] H. Matsuda. Physical nature of higher-order mutual information: intrinsic correlations and frustration. *Physical Review E*, 62(3):3096–3102, 2000.

[142] M. Meila. *Learning Mixtures of Trees.* PhD thesis, Massachusetts Institute of Technology, 1999.

[143] J. Meller, M. Wagner, and R. Elber. Maximum feasibility guideline to the design and analysis of protein folding potentials. *Journal of Computational Chemistry*, 23:111–118, 2002.

[144] N. Metropolis, A. Rosenbluth, A. Teller, and E. Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1091, 1953.

[145] L. Meyerguz, C. Grasso, J. Kleinberg, and R. Elber. Computational analysis of sequence selection mechanisms. *Structure*, 12(4):547–557, 2004.

[146] L. Meyerguz, D. Kempe, J. Kleinberg, and R. Elber. The evolutionary capacity of protein structures. In *Proceedings of the Eighth Annual International Conference on Research in Computational Molecular Biology*, pages 290–297, San Diego, California, 2004. Morgan Kaufmann Publishers, San Francisco, CA.

[147] T. Minka. *A Family of Algorithms for Approximate Bayesian Inference*. PhD thesis, Massachusetts Institute of Technology, Massachusetts, 2001.

[148] T. Miquélez, E. Bengoetxea, and P. Larrañaga. Evolutionary computation based on Bayesian classifiers. *International Journal of Applied Mathematics and Computer Science*, 14(3):101–115, 2004.

[149] N. Mladenović. A variable neighborhood algorithm – a new metaheuristics for combinatorial optimization. In *Abstracts of Papers Presented at Optimization Days. Montréal*, page 112, 1995.

[150] N. Mladenović and P. Hansen. Variable neighborhood search. *Computers and Operation Research*, 24:1097–1100, 1997.

[151] D. A. Moffet and M. H. Hecht. De novo proteins from combinatorial libraries. *Chemical Reviews*, 101(10):3191–3203, 2001.

[152] G. L. Moore and C. D. Maranas. Identifying residue-residue clashes in protein hybrids by using a second-order mean-field approach. *Proceedings of the National Academy of Sciences*, 100(9):5091–5096, 2003.

[153] T. Morita. Cluster variation method for non-uniform Ising and Heisenberg models and spin-pair correlation function. *Progress of Theoretical Physics*, 85(2):243–255, 1991.

[154] T. Morita. Formal structure of the cluster variation method. *Progress of Theoretical Physics Supplements*, 115:27–39, 1994.

[155] S. Muggleton. Inductive logic programming. In S. Muggleton, editor, *Inductive logic programming*, pages 3–27. Academic Press, London, 1992.

[156] H. Mühlenbein and R. Höns. The estimation of distributions and the minimum relative entropy principle. *Evolutionary Computation*, 13(1):1–27, 2005.

[157] H. Mühlenbein and T. Mahnig. Evolutionary computation and beyond. In Y. Uesaka, P. Kanerva, and H. Asoh, editors, *Foundations of Real-World Intelligence*, pages 123–188. CSLI Publications, Stanford, California, 2001.

[158] H. Mühlenbein and T. Mahnig. Evolutionary synthesis of Bayesian networks for optimization. In M. Patel, V. Honavar, and K. Balakrishnan, editors, *Advances*

in Evolutionary Synthesis of Intelligent Agents, pages 429–455. MIT Press, Cambridge, Mass., 2001.

[159] H. Mühlenbein and T. Mahnig. Evolutionary optimization and the estimation of search distributions with applications to graph bipartitioning. *International Journal on Approximate Reasoning*, 31(3):157–192, 2002.

[160] H. Mühlenbein, T. Mahnig, and A. Ochoa. Schemata, distributions and graphical models in evolutionary optimization. *Journal of Heuristics*, 5(2):213–247, 1999.

[161] H. Mühlenbein and G. Paaß. From recombination of genes to the estimation of distributions I. Binary parameters. In H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN IV*, pages 178–187, Berlin, 1996. Springer Verlag. LNCS 1141.

[162] H. Mühlenbein and J. Zimmermann. Size of neighborhood more important than temperature for stochastic local search. In *Proceedings of the 2000 Congress on Evolutionary Computation CEC-2000*, pages 1017–1024. IEEE Press, 2000.

[163] B. Naudts and L. Kallel. A comparison of predictive measures of problem difficulty in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 4(1):1–15, 2000.

[164] D. Nilsson. An efficient algorithm for finding the M most probable configurations in probabilistic expert systems. *Statistics and Computing*, 2:159–173, 1998.

[165] M. Norman and P. Moscato. A competitive and cooperative approach to complex combinatorial search. Technical Report Caltech Concurrent Computation Program, Report. 790, California Institute of Technology, Pasadena, California, USA, 1989.

[166] J. Ocenasek. Entropy-based convergence measurement in discrete estimation of distribution algorithms. In J. A. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea, editors, *Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms*, pages 39–50. Springer-Verlag, 2006. In Press.

[167] A. Ochoa, H. Mühlenbein, and M. Soto. Factorized Distribution Algorithms using Bayesian networks bounded complexity. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2000*, pages 212–215, 2000.

[168] A. Ochoa, H. Mühlenbein, and M. R. Soto. A Factorized Distribution Algorithm using single connected Bayesian networks. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN VI 6th International Conference*, Paris, France, September 16-20 2000. Springer Verlag. LNCS 1917.

[169] A. Ochoa and M. R. Soto. Linking entropy to estimation of distribution algorithms. In J. A. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea, editors, *Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms*, pages 1–38. Springer-Verlag, 2006. In press.

[170] K. G. Olesen and A. L. Madsen. Maximal prime subgraph decomposition of Bayesian networks. *IEEE Transactions on Systems, Man and Cybernetics: Part B*, 32(1):21–31, 2000.

[171] J. N. Onuchic, H. Nymeyer, A. E. Garcia, J. Chahine, and N. D. Socci. The energy landscape theory of protein folding: Insights into folding mechanisms and scenarios. *Advances in Protein Chemistry*, 53:87–152, 2000.

[172] J. N. Onuchic and P. G. Wolynes. Theory of protein folding. *Current Opinion in Structural Biology*, 14:70–75, 2004.

[173] P. Pakzad and V. Anantharam. Estimation and marginalization using Kikuchi based methods. Technical Report M04-21, Faculty of Electrical Engineering and Computer Science. University of California Berkeley, 2003.

[174] V. S. Pande, A. Y. Grosberg, T. Tanaka, and D. S. Rokhsar. Protein folding pathways: Is a 'new view' needed? *Current Opinion in Structural Biology*, 8(1):68–79, 1998.

[175] R. V. Pappu, G. R. Marshall, and J. W. Ponder. A potential smoothing algorithm accurately predicts transmembrane helix packing. *Nature Structural Biology*, 6:50–55, 1999.

[176] S. Park, H. Kono, W. Wang, E. T. Boder, and J. G. Saven. Progress in the development and application of computational methods for probabilistic protein design. *Computers and Chemical Engineering*, 29:407–421, 2005.

[177] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, California, 1988.

[178] M. Pelikan. *Hierarchical Bayesian Optimization Algorithm. Toward a New Generation of Evolutionary Algorithms*. Studies in Fuzziness and Soft Computing. Springer, 2005.

[179] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz. BOA: The Bayesian optimization algorithm. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-1999*, volume I, pages 525–532, Orlando, FL, 1999. Morgan Kaufmann Publishers, San Francisco, CA.

[180] M. Pelikan and H. Mühlenbein. The bivariate marginal distribution algorithm. In R. Roy, T. Furuhashi, and P. Chawdhry, editors, *Advances in Soft Computing - Engineering Design and Manufacturing*, pages 521–535, London, 1999. Springer-Verlag.

[181] D. Pelta and N. Krasnogor. Multimeme algorithms using fuzzy logic based memes for protein structure prediction. In W. H. William, N. Krasnogor, and J. E. Smith, editors, *Recent Advances in Memetic Algorithms*, Studies in Fuzziness and Soft Computing, pages 49–64. Springer, 2004.

[182] N. A. Pierce and E. Winfree. Protein design is NP-hard. *Protein Engineering*, 15(10):779–782, 2002.

[183] N. Pokala and T. M. Handel. Review: Protein design–where we were, where we are, where we're going. *Journal of Structural Biology*, 134:269–281, 2001.

[184] J. W. Ponder and F. M. Richard. Tertiary templates for proteins. Use of packing criteria in the enumeration of allowed sequence for different structure classes. *Journal of Molecular Biology*, 193:775–791, 1987.

[185] V. Robles, P. de Miguel, and P. Larrañaga. Solving the traveling salesman problem with EDAs. In P. Larrañaga and J. A. Lozano, editors, *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, pages 227–238. Kluwer Academic Publishers, Boston/Dordrecht/London, 2002.

[186] V. Robles, J. M. Peña, M. S. Pérez, and V. Herves. GA-EDA: A new hybrid cooperative search evolutionary algorithm. In J. A. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea, editors, *Towards a New Evolutionary Computation. Advances in Estimation of Distribution Algorithms*, pages 187–220. Springer Verlag, 2006.

[187] I. Rodríguez, J. M. Moreno, and J. A. Moreno. Variable neighborhood tabu search and its application to the median cycle problem. *European Journal of Operations Research*, 151(2):365–378, 2003.

[188] C. A. Rohl, C. E. M. Strauss, K. Misura, and D. Baker. Protein structure prediction using Rosetta. *Methods in Enzymology*, 383:66–93, 2004.

[189] R. Santana. A Markov network based factorized distribution algorithm for optimization. In *Proceedings of the 14th European Conference on Machine Learning (ECML-PKDD 2003)*, volume 2837 of *Lecture Notes in Artificial Intelligence*, pages 337–348, Dubrovnik, Croatia, 2003. Springer-Verlag.

[190] R. Santana. Estimation of distribution algorithms with Kikuchi approximations. *Evolutionary Computation*, 13(1):67–97, 2005.

[191] R. Santana, E. P. de León, and A. Ochoa. The edge incident model. In *Proceedings of the Second Symposium on Artificial Intelligence (CIMAF-99)*, pages 352–359, Habana, Cuba, March 1999.

[192] R. Santana, P. Larrañaga, and J. A. Lozano. Protein folding in 2-dimensional lattices with estimation of distribution algorithms. In *Proceedings of the First International Symposium on Biological and Medical Data Analysis*, volume 3337 of *Lecture Notes in Computer Science*, pages 388–398, Barcelona, Spain, 2004. Springer Verlag.

[193] R. Santana, P. Larrañaga, and J. A. Lozano. Interactions and dependencies in estimation of distribution algorithms. In *Proceedings of the 2005 Congress on Evolutionary Computation CEC-2005*, pages 1418–1425, Edinburgh, U.K., 2005. IEEE Press.

[194] R. Santana, P. Larrañaga, and J. A. Lozano. Combining variable neighborhood search and estimation of distribution algorithms in the protein side chain placement problem. 2006. Submitted for publication.

[195] R. Santana and H. Mühlenbein. Blocked stochastic sampling versus Estimation of Distribution Algorithms. In *Proceedings of the 2002 Congress on Evolutionary Computation CEC-2002*, volume 2, pages 1390–1395. IEEE press, 2002.

[196] R. Santana, A. Ochoa, and M. R. Soto. The mixture of trees factorized distribution algorithm. In L. Spector, E. Goodman, A. Wu, W. Langdon, H. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon, and E. Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2001*, pages 543–550, San Francisco, CA, 2001. Morgan Kaufmann Publishers.

[197] K. Sastry, M. Pelikan, and D. Goldberg. Efficiency enhancement of genetic algorithms via building-block-wise fitness estimation. In *Proceedings of the 2004 Congress on Evolutionary Computation CEC-2004*, pages 720–727, Portland, Oregon, 2004. IEEE Press.

[198] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 7(2):461–464, 1978.

[199] H.-P. Schwefel. *Numerical Optimization of Computer Models*. John Wiley & Sons, 1981.

[200] R. P. Sear. Highly specific protein-protein interactions, evolution and negative design. *Physical Biology*, 1(3):166–172, 2004.

[201] S. Shakya, J. McCall, and D. Brown. Using a Markov network model in a univariate EDA: An empirical cost-benefit analysis. In *Proceedings of Genetic and Evolutionary Computation Conference GECCO-2005*, pages 727–734, Washington, D.C., USA, 2005. ACM.

[202] P. S. Shenkin, H. Farid, and J. S. Fetrow. Prediction and evaluation of side-chain conformations for protein backbone structures. *Proteins: Structure, Function, and Genetics*, 26(3):323–352, 1998.

[203] A. Shmygelska, R. A. Hernández, and H. H. Hoos. An ant colony optimization algorithm for the 2D HP protein folding problem. In *Proceedings of the Third International Workshop on Ant Algorithms*, pages 40–53. Springer Verlag, 2002.

[204] A. Shmygelska and H. H. Hoos. An improved ant colony optimization algorithm for the 2D HP protein folding problem. In Y. Xiang and B. Chaib-draa, editors, *Advances in Artificial Intelligence*, volume 2671 of *Lecture Notes in Computer Science*, pages 400–417. Springer Verlag, 2003.

[205] D. Shortle. Composites of local structure propensities: Evidence for local encoding of long-range structure. *Protein Science*, 11(1):18–26, 2002.

[206] J. Smith. The co-evolution of memetic algorithms for protein structure prediction. In W. H. William, N. Krasnogor, and J. E. Smith, editors, *Recent Advances in Memetic Algorithms*, Studies in Fuzziness and Soft Computing, pages 105–128. Springer, 2004.

[207] M. R. Soto. *A Single Connected Factorized Distribution Algorithm and Its Cost of Evaluation*. PhD thesis, University of Havana, Havana, Cuba, July 2003. In Spanish.

[208] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction and Search*, volume 81 of *Lecture Notes in Statistics*. Springer-Verlag, New York, 1993.

[209] P. Spirtes and C. Meek. Learning Bayesian networks with discrete variables from data. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, pages 294–299, San Francisco, 1995. Morgan Kaufmann.

[210] B. Steipe. Protein design concepts. In P. V. R. Schleyer, N. L. Allinger, T. Clark, J. Gasteiger, P. A. Kollman, H. F. Schaefer III, and P. R. Schreiner, editors, *The Encyclopedia of Computational Chemistry*, pages 2168–2185. John Wiley & Sons, Chichester, 1998.

[211] F. H. Stillinger and T. A. Weber. Nonlinear optimization simplified by hypersurface deformation. *Journal of Statistical Physics*, 52:1429–1445, 1988.

[212] M. Stone. Cross-validation choice and assessment of statistical procedures. *Journal Royal of Statistical Society*, 36:111–147, 1974.

[213] B. K. Sy. A recurrence local computation approach towards ordering composite beliefs in Bayesian belief networks. *International Journal Approximated Reasoning*, 8:17–50, 1993.

[214] R. Tarjan. Decomposition by clique separators. *Discrete Mathematics*, 55:221–232, 1985.

[215] S. Tatikonda and M. I. Jordan. Loopy belief propagation and Gibbs measures. In *Proceedings of the 18th Annual Conference on Uncertainty in Artificial Intelligence (UAI-2002)*, pages 493–500. Morgan Kaufmann Publishers, 2002.

[216] Y. W. Teh and M. Welling. On improving the efficiency of the iterative proportional fitting procedure. In *Proceedings of the International Workshop on Artificial Intelligence and Statistics*, volume 9, 2003.

[217] D. Tobi and R. Elber. Distance-dependent, pair potential for protein folding: Results from linear optimization. *Proteins*, 41(1):40–46, 2000.

[218] A. E. Torda. Protein sequence optimisation - theory, practice and fundamental impossibility. *Soft Materials*, 2:1–10, 2004.

[219] M. Toussaint. Factorial representations to generate arbitrary search distributions. In *Genetic and Evolutionary Computation Conference GECCO-2005, Workshop Program*, pages 339–345, Washington, D.C., USA, 2005. ACM Press.

[220] P. Tuffery, C. Etchebest, S. Hazout, and R. Lavery. A new approach to the rapid determination of protein side chain conformations. *Journal of Biomolecular Structure Dynamics*, 8:1267–1289, 1991.

[221] R. Unger and J. Moult. Genetic algorithms for protein folding simulations. *Journal of Molecular Biology*, 231:75–81, 1993.

[222] M. Vasquez. Modeling side-chain conformation. *Current Opinion in Structural Biology*, 6(2):217–221, 1996.

[223] C. Venclovas, A. Zemla, K. Fidelis, and J. Moult. Comparison of performance in successive casp experiments. *Proteins: Structure, Function, and Genetics*, 5:163–170, 2001.

[224] C. A. Voigt, D. B. Gordon, and S. L. Mayo. Trading accuracy for speed: A quantitative comparison of search algorithms in protein sequence design. *Journal of Molecular Biology*, 299(3):799–803, 2000.

[225] M. J. Wainwright, T. Jaakkola, and A. S. Willsky. Tree-based reparameterization framework for analysis of belief propagation and related algorithms. Technical Report LIDS P-2510, Laboratory for Information and Decision Systems, MIT, 2001.

[226] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. Technical Report 649, Department of Statistics, University of California, Berkeley, September 2003.

[227] D. J. Wales and H. A. Scheraga. Global optimization of clusters, crystals, and biomolecules. *Science*, 285(5432):1368 – 1372, 1999.

[228] L. Wang and T. Jiang. On the complexity of multiple sequence alignment. *Journal of Computational Biology*, 1(4):337–348, 1994.

[229] A. L. Watters and D. Baker. Searching for folded proteins in vitro and in silico. *European Journal of Biochemistry*, 271(9):1615–1622, 2004.

[230] Y. Wei, T. Liu, S. L. Sazinsky, D. A. Moffet, I. Pelczer, and M. H. Hecht. Stably folded de novo proteins from a designed combinatorial library. *Protein Science*, 12:92–102, 2003.

[231] S. J. Weiner, P. A. Kollman, D. T. Nguyen, and D. A. Case. An all atom force field for simulations of proteins and nucleic acids. *Journal of Computational Chemistry*, 7:230–252, 1986.

[232] M. Welling. On the choice of regions for generalized belief propagation. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence (UAI-2004)*, pages 585–592, Banff, Canada, 2004. Morgan Kaufmann Publishers.

[233] L. Wernisch, S. Hery, and S. Wodak. Automatic protein design with all atom force-fields by exact and heuristic optimization. *Journal of Molecular Biology*, 301(3):713–736, 2000.

[234] J. Whittaker. *Graphical Models in Applied Multivariate Statistics*. Wiley Series in Probability and Mathematical Statistics, New York, 1991.

[235] D. H. Wolpert and W. G. Macready. No free lunch theorem for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, April 1997.

[236] Y. Xiang, S. K. M. Wong, and N. Cercone. A microscopic study of minimum entropy search in learning decomposable Markov networks. *Machine Learning*, 26(1):65–92, 1997.

[237] E. P. Xing and M. I. Jordan. Graph partition strategies for generalized mean field inference. Technical Report CSD-03-1274, Division of Computer Science, University of California, Berkeley, 2003.

[238] E. P. Xing, M. I. Jordan, and S. Russell. Graph partition strategies for generalized mean field inference. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence (UAI-2004)*, pages 602–610, Banff, Canada, 2004. Morgan Kaufmann Publishers.

[239] J.-M. Yang, C.-H. Tsai, M.-J. Hwang, H.-K. Tsai, J.-K. Hwang, and C.-Y. Kao. GEM: A Gaussian evolutionary method for predicting protein side-chain conformations. *Protein Science*, 11:1897–1907, 2002.

[240] C. Yanover and Y. Weiss. Approximate inference and protein-folding. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 1457–1464. MIT Press, Cambridge, MA, 2003.

[241] C. Yanover and Y. Weiss. Approximate inference for side-chain prediction. Submitted for publication, 2004.

[242] C. Yanover and Y. Weiss. Finding the M most probable configurations using loopy belief propagation. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

[243] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. Technical Report TR-2001-22, Mitsubishi Electric Research Laboratories, November 2001.

[244] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Constructing free energy approximations and generalized belief propagation algorithms. Technical Report TR-2002-35, Mitsubishi Electric Research Laboratories, August 2002.

[245] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Constructing free energy approximations and generalized belief propagation algorithms. Technical Report TR-2004-040, Mitsubishi Electric Research Laboratories, May 2004.

[246] A. Yuille. A double-loop algorithm to minimize the Bethe and Kikuchi free energies. *Neural Computation*, 14(6):1691–1722, 2001.

[247] C. Zhang, S. Liu, H. Zhou, and Y. Zhou. An accurate, residue-level, pair potential of mean force for folding and binding based on the distance-scaled, ideal-gas reference state. *Protein Science*, 13(2):400–411, 2004.

[248] J. Zhang, R. Chen, and J. Liang. Empirical potential function for simplified protein models: Combining contact and local sequence-structure descriptors. *Protein*, 2006. In press.

[249] Q. Zhang. On stability of fixed points of limit models of univariate marginal distribution algorithm and factorized distribution algorithm. *IEEE Transactions on Evolutionary Computation*, 8(1):80–93, 2004.

[250] Y. Zhang, D. Kihara, and J. Skolnick. Local energy landscape flattening: Parallel hyperbolic Monte Carlo sampling of protein folding. *Proteins: Structure, Function, and Genetics*, 48:192–201, 2002.

[251] A. Zhou, Q. Zhang, Y. Jin, and E. P. K. Tsang. A model-based evolutionary algorithm for bi-objective optimization. In *Proceedings of the 2005 Congress on Evolutionary Computation CEC-2005*, pages 2568–2575, Edinburgh, U.K., 2005. IEEE Press.

[252] J. Zhu, Q. Zhu, Y. Shi, and H. Liu. How well can we predict native contacts in proteins based on decoy structures and their energies? *Proteins: Structure, Function, and Genetics*, 52(4):598–608, 2003.

[253] J. Zou and J. G. Saven. Using self-consistent fields to bias Monte Carlo methods with applications to designing and sampling protein sequences. *The Journal of Chemical Physics*, 118(8):3843–3854, 2003.

[254] M. Zviling, H. Leonov, and I. T. Arkin. Genetic algorithm-based optimization of hydrophobicity tables. *Bioinformatics*, 21(6):2651 – 2656, 2005.