Estimation of Distribution Algorithms and Minimum Relative Entropy

Dissertation
zur Erlangung des Doktorgrades (Dr. rer. nat.)
der Mathematisch-Naturwissenschaftlichen Fakultät
der Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von
Robin Höns
aus
Bonn

Bonn, Mai 2005

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Rheinischen Friedrich-Wilhelms-Universität Bonn. Diese Dissertation ist auf dem Hochschulschriftenserver der ULB Bonn http://hss.ulb.uni-bonn.de/diss_online elektronisch publiziert.
Erscheinungsjahr: 2006

1. Referent: Prof. Dr. S. Wrobel
2. Referent: Prof. Dr. J. K. Anlauf
Tag der Promotion: 15. Mai 2006

Abstract

In the field of optimization using probabilistic models of the search space, this thesis identifies and elaborates several advancements in which the principles of maximum entropy and minimum relative entropy from information theory are used to estimate a probability distribution.

The probability distribution within the search space is represented by a graphical model (factorization, Bayesian network or junction tree). An estimation of distribution algorithm (*EDA*) is an evolutionary optimization algorithm which uses a graphical model to sample a population within the search space and then estimates a new graphical model from the selected individuals of the population.

- So far, the Factorized Distribution Algorithm (*FDA*) builds a factorization or Bayesian network from a given additive structure of the objective function to be optimized using a greedy algorithm which only considers a subset of the variable dependencies. Important connections can be lost by this method. This thesis presents a heuristic *subfunction merge* algorithm which is able to consider all dependencies between the variables (as long as the marginal distributions of the model do not become too large).

  On a 2-D grid structure, this algorithm builds a pentavariate factorization which allows to solve the deceptive grid benchmark problem with a much smaller population size than the conventional factorization. Especially for small population sizes, calculating large marginal distributions from smaller ones using Maximum Entropy and iterative proportional fitting leads to a further improvement.

- The second topic is the generalization of graphical models to loopy structures. Using the Bethe-Kikuchi approximation, the loopy graphical model (region graph) can learn the Boltzmann distribution of an objective function by a generalized belief propagation algorithm (GBP). It minimizes the free energy, a notion adopted from statistical physics which is equivalent to the relative entropy to the Boltzmann distribution.

  Previous attempts to combine the Kikuchi approximation with *EDA* have relied on an expensive Gibbs sampling procedure for generating a population from this loopy probabilistic model. In this thesis a combination with a factorization is presented which allows more efficient sampling. The free energy is generalized to incorporate the inverse temperature β. The factorization building algorithm mentioned above can be employed here, too.

The dynamics of GBP is investigated, and the method is applied on Ising spin glass ground state search. Small instances (7×7) are solved without difficulty. Larger instances (10×10 and 15×15) do not converge to the true optimum with large β, but sampling from the factorization can find the optimum with about 1000-10000 sampling attempts, depending on the instance. If GBP does not converge, it can be replaced by a concave-convex procedure which guarantees convergence.

- Third, if no probabilistic structure is given for the objective function, a Bayesian network can be learned to capture the dependencies in the population. The relative entropy between the population-induced distribution and the Bayesian network distribution is equivalent to the log-likelihood of the model. The log-likelihood has been generalized to the BIC/MDL score which reduces overfitting by punishing complicated structure of the Bayesian network. A previous information theoretic analysis of BIC/MDL in the context of *EDA* is continued, and empiric evidence is given that the method is able to learn the correct structure of an objective function, given a sufficiently large population.

- Finally, a way to reduce the search space of *EDA* is presented by combining it with a local search heuristics. The Kernighan Lin hillclimber, known originally for the traveling salesman problem and graph bipartitioning, is generalized to arbitrary binary problems. It can be applied in a stand-alone manner, as an iterative 1+1 search algorithm, or combined with *EDA*. On the MAXSAT problem it performs in a similar scale to the specialized SAT solver Walksat. An analysis of the Kernighan Lin local optima indicates that the combination with an *EDA* is favorable.

The thesis shows how evolutionary optimization can be improved using interdisciplinary results from information theory, statistics, probability calculus and statistical physics. The principles of information theory for estimating probability distributions are applicable in many areas. *EDA*s are a good application because an improved estimation affects directly the optimization success.

Contents

List of Algorithms

Acknowledgements

First of all, I would like to thank warmly my supervisor, Dr. Heinz Mühlenbein. Dear Heinz, I will always remember our fruitful, friendly and sometimes funny discussions, which have elevated me very much and inspired my scientific work. Your support and encouragement was indispensable for the completion of this thesis.

I thank very much my supervisors at the university of Bonn, Prof. Wrobel and Prof. Anlauf, for their help, hints and suggestions which helped to improve my work. Many thanks to Prof. Rolf Klein and Prof. Albeverio for their kind agreement to join the doctorate commission.

I also owe very much (if not all) to my family, who have helped me in more ways than I could conceive. It is wonderful to know that my parents and my sister are always at my side.

Many thanks to my friends who in the last years have helped me, coached me, or just gave me the courage and power to continue my way, particularly to Susanne Böttger, Christina Fleischer, Markus Gröschl, Dieter Lucht, Andrea and Thilo Mahnig and Roberto Santana, as well as to one person who prefers to stay unnamed and one person whose name slipped my mind. I am especially thankful to Thilo and Roberto for reading previous versions of this thesis and giving valuable hints.

Many thanks to the German academic exchange service (DAAD) for supporting my stay in San Sebastián with a doctorand scholarship. I thank very much Pedro Larrañaga, José Antonio Lozano and those who encircled me daily in the despacho, clockwise: Borxa, Ramón, Rosa, Guzman, Rubén ("sidiga"), Aritz and Roberto. Thanks to you all for the warm reception, the scientific help, and the friendly and cordial atmosphere which reigns in your group.

Last but definitely not least, I thank Raquel Córdova from all of my heart. Mi chiquita, gracias por tu amor, tu paciencia, el apoyo que me das, tu alegría y tu bondad. Ya sé que no siempre era facil conmigo, que estaba muy atareado, que te dejaba sola para irme a España; pero aun cuando estamos separados, siempre te siento conmigo, y por eso te digo gracias.

Esta tesis es para ti.

1. Introduction

Finding the optimum of a function f is an important problem in computer science. In the class of population-based optimization algorithms, Estimation of Distribution Algorithms (*EDA*) [MP96, LL01] constitute a new and promising approach which proved its fitness on many difficult optimization problems [LL01, MM02, PG03].

This thesis presents several improvements of *EDA* which profit from an interdisciplinary research influenced by statistics, information theory and statistical physics.

EDA can be divided into three classes: Algorithms which use a pre-defined probabilistic structure [MP96, BC95], algorithms which exploit a known structure of the problem [MMO99, Bal02], and algorithms which determine a probabilistic structure from the population itself [BIV97, PM99, EL99, PGC99, MM99].

An efficient algorithm which exploits a given probabilistic structure is the Factorized Distribution Algorithm (*FDA*). In *FDA* [MMO99] an additive structure on the function is turned into a probability structure by choosing heuristically a subset of the connections and disregarding the rest. Important connections can be lost by this method. Therefore the question is: Is it possible to include more or even all of the connections? This thesis presents an algorithm which does this by merging marginal distributions, thus capturing more variable dependencies. The larger marginals can be critical to estimate particularly from small populations. In this case the principle of maximum entropy from information theory helps to improve the estimate.

FDA uses factorizations, equivalent to Bayesian networks. These are cycle-free, but of limited expressive power. Is it possible to extend the regarded model class to loopy models? Loopy probabilistic models can be identified with the Kikuchi approximation, which has been adopted from statistical physics [AM01, MY02, YFW01, YFW04]. *EDA* relies on sampling points from probabilistic models. A previous attempt to use the Kikuchi approximation for *EDA* [San05] employs a Gibbs sampling procedure which is computationally expensive. This thesis presents a way to combine the Kikuchi approximation with a factorization, in order to profit from the loopy structure, but nevertheless be able to sample points and use this for optimization.

The class of *EDA* which determine a probabilistic structure from the population are another application of the information theory principles. Minimum relative entropy is the basis of the minimum description length (MDL) score [Ris78, Sch78, Grü98] used for learning a structure from population data [FG99]. Previous analysis of structure learning *EDA*s [PG03, PGOT03] has relied on the benchmark method: The algorithm is applied on a benchmark function, and the optimization success is measured. Can information theory be used to analyze in detail the algorithm's dynamics and results? This thesis performs an empiric analysis which gives evidence that the method is able to recognize a structure, given a sufficiently large population size. A strong correlation

between the mutual information of variables and their connection in *EDA* is shown. An analysis of the MDL space shows how the parameters influence the learning dynamics.

In order to reduce the search space of *EDA* and thus make them applicable for complex problems, they can be combined with a local hillclimber. Several works on *EDA* have used a simple one-bit hillclimber [PG03, San04]. Is it reasonable to combine *EDA* with a more sophisticated hillclimber? For special problems (graph bipartitioning [KL70] and the traveling salesman problem [LK73]), the Kernighan Lin hillclimber turned out to be very efficient, also in combination with *EDA* [MM02]. Can this hillclimber be generalized for arbitrary binary problems, in order to be more applicable in *EDA* (or also alone)? This thesis presents this generalization and performs an analysis of its space of local optima, in order to assess how reasonable is the combination with *EDA*.

This introduction now first describes in detail the background of the thesis and the work which it improves. Population-based optimization, *EDA*s and some theoretical work about them are described. Then in Sect. 1.2 the contributions of this thesis are explained. Finally, Sect. 1.3 outlines the structure of the thesis.

1.1. Background of the Thesis

1.1.1. Optimization

The objective of this thesis is optimization. This is an important problem in computer science. There are many instances of this problem, like maximizing the number of satisfied clauses for a Boolean formula, minimizing the cost of a traveling salesman tour, minimizing the number of edges crossing a graph bipartitioning, and many others.

Such problems can be very complex, so that the help of computers is required to solve them. Many real-life applications can be given, from chip design over timetable management to aircraft design.

We code optimization problems mathematically. Let $\mathcal{D}$ be the domain of possible solution of the problem. For each $x \in \mathcal{D}$ there is given a single, unique function value $f(x) \in \mathbb{R}$ which is to be maximized or minimized. Furthermore, we assume the domain $\mathcal{D}$ to be finite and discrete. In this case, every $x \in \mathcal{D}$ can be coded as a binary vector, so usually we assume $\mathcal{D} = \{0,1\}^n$ and $\mathbf{x} \in \mathcal{D}$ to be an n-dimensional binary vector.

1.1.2. Population-based Optimization

There are many methods for solving such optimization problems. The simplest is to calculate the function value of all $\mathbf{x} \in \mathcal{D}$, but this is usually unfeasible because the domain size $|\mathcal{D}| = 2^n$ is too large.

The starting point of this thesis is population-based optimization. This paradigm works with a set of *individuals* $\mathfrak{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \subseteq \mathcal{D}$, the *population*. Historically, population-based optimization began with the *genetic algorithm* (*GA*) [Hol75, Gol89]. Its basic idea is to mimic the force of evolution[1], which leads to well-adapted, thus

[1] This approach was already proposed by Turing in 1948 and then pursued by several researchers independently. [Fog98, HKS04]

optimized solutions. This origin has affected the terminology of population-based optimization, which is imbued with biological terms.

Starting with a random population $\mathfrak{X}_0$, the *GA* first computes the *fitness* $f(\mathbf{x})$ of all $\mathbf{x} \in \mathfrak{X}_0$. Then it *selects* the best individuals $\hat{\mathfrak{X}}_0 \subseteq \mathfrak{X}_0$ of the population. These selected individuals are coupled and *recombined*: For a couple of parents $\mathbf{x}_i, \mathbf{x}_j \in \hat{\mathfrak{X}}_0$, a child vector is constructed by taking some of its bits (*alleles*) from $\mathbf{x}_i$ and the others from $\mathbf{x}_j$. The children form the next *generation* $\mathfrak{X}_1$.

This scheme is iterated, which leads to a sequence $\mathfrak{X}_t$ $(t = 0, 1, 2, \ldots)$ of generations. Whereas $\mathfrak{X}_0$ is spread all over the search space $\mathcal{D}$, subsequent generations are more and more concentrated on regions with high fitness.

The basic assumption is that points of high fitness are concentrated in the search space, so that "good points lead to better points". Analogously, the highest mountains on Earth all lie in the same region. Therefore this property is also called the "Himalaya effect". And just like by searching between Annapurna and Kanchenjunga we would find Mount Everest, combining good solutions for an optimization problem might lead to a better or even optimal solution.

1.1.3. Estimation of Distribution Algorithms

Evolutionary optimization has been applied successfully in many real-world applications. The *GA* is simple to implement, there are few parameters to be chosen; yet it shows good performance. For many years, the *GA* community believed to have found the "optimal" search paradigm.

There are several recombination/crossover techniques. But it is an old result from population genetics that asymptotically they are all equivalent [Gei44]. They approximate sampling the new population from the *simple product distribution*

$$p(\mathbf{x}) = \prod_{i=1}^{n} p_i(x_i) \ . \tag{1.1}$$

This distribution, in which all bits are independent of each other, is also called *linkage equilibrium* or *Robbins' proportion* [Rob18].

Mühlenbein and Paaß [MP96] have therefore used directly (1.1) for generating the next population, instead of some recombination scheme which is just a biased approximation of the same formula.

This founded the new track of Estimation of Distribution Algorithms (*EDA*). Remarkable is the new perspective: Instead of mimicking biological operators, a probabilistic model $p_t(\mathbf{x})$ in the space $\mathcal{D}$ is built using the selected population. Then this model is used to generate new individuals for the next generation.

If the distribution (1.1) is used, this means first estimating the univariate marginal probabilities $p_i(x_i)$ from the selected population $\hat{\mathfrak{X}}_t$ and then drawing samples from these distributions independently, so that the new generation $\mathfrak{X}_{t+1}$ is distributed according to (1.1). This algorithm is called *Univariate Marginal Distribution Algorithm* (*UMDA*) [MP96].

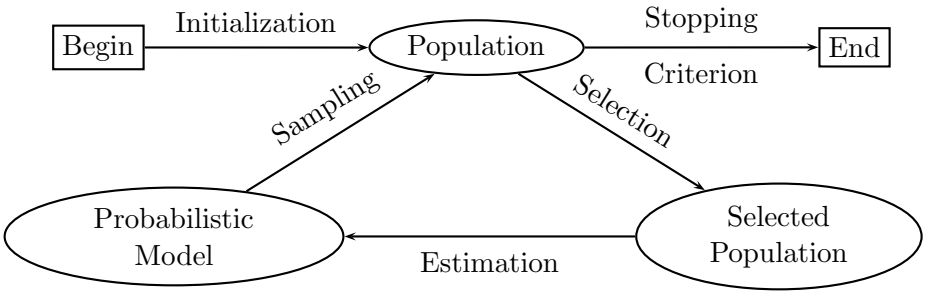

Figure 1.1.: The basic loop of Estimation of Distribution Algorithms

It is at this point that the research becomes interdisciplinary. The problem to estimate a probability distribution from data is well-known from statistics. In this dissertation, results of statistics, belief networks, information theory and statistical physics are used to understand and further develop *EDA*.

The first observation that has been made is that *UMDA* cannot handle problems in which some variables are strongly correlated. These variables have to be changed together in order to obtain an improvement. This can be accomplished by replacing $p_i(x_i)$ in (1.1) by conditional distributions $p_i(x_i|x_j, x_k, \ldots)$. Such a probabilistic model is called a *factorization* of the distribution $p(\mathbf{x})$. It can be used to incorporate *a priori* knowledge about the structure of the objective function f (like "x_i is dependent on x_j and x_k") into the model. This has been done in the Factorized Distribution Algorithm (*FDA*) [MMO99, MM99].

Factorizations are strongly related to Bayesian networks [Lau96]. These and other graphical models for describing probability distributions have been used successfully in the context of *EDA*, too. If there is no *a priori* knowledge given, the structure of the distribution can also be *learned* from the selected population itself. There are several methods for learning Bayesian networks which have been developed in the field of graphical models [Jor99] and were applied in *EDA*, too [EL99, PGC99, MM99].

1.1.4. Boltzmann Distributions and Graphical Models

In the theoretical analysis of *EDA* the Boltzmann distribution

$$p_{\beta,f}(x) := \frac{e^{\beta f(x)}}{\sum_{y \in \mathcal{D}} e^{\beta f(y)}} \tag{1.2}$$

plays a vital role. It is very useful for optimization, and it has been applied for this purpose e. g. in simulated annealing [KGV83].

Its favorable properties are the following. The points of maximal probability are the maxima of $f(x)$. The parameter $\beta \geq 0$ controls how sharply peaked the distribution is. For $\beta = 0$ it is the uniform distribution. The higher β is chosen, the more the distribution concentrates on the maxima.

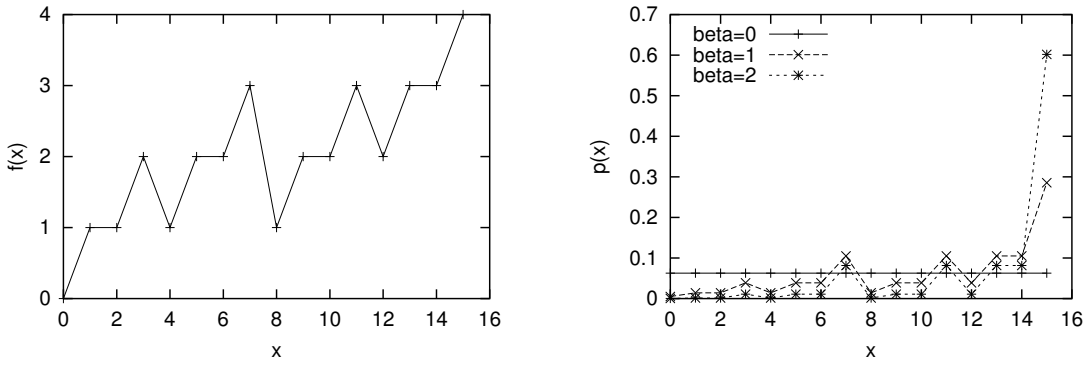

Figure 1.2.: An example fitness function $f(x)$ (OneMax-4) and its Boltzmann distribution for various β.

An example of this can be seen in Fig. 1.2. The left figure depicts the function $f(x)$. In this simple example it is the function Onemax-4, the number of ones in the binary representation of $x \in \{0, 1, \ldots, 15\}$. The right side shows several Boltzmann distributions for this function. It can be seen that the probability of the maximum $x = 15$ grows with β.

These properties of the Boltzmann distribution induce the method of optimization by sampling from Boltzmann distributions. However, in general it is not possible to sample efficiently from a Boltzmann distribution because the sum in (1.2) cannot be computed efficiently. But there are other schemes which achieve this indirectly. For example, simulated annealing [KGV83] uses a random walker which is Boltzmann-distributed over time average. The parameter β is increased over time.

In *EDA*, it can be shown that the populations of the generations are Boltzmann-distributed. The initial population is uniformly distributed, so we start with $\beta = 0$. Selection can be understood as an operator to increase β and produce a more sharply peaked distribution. This can be proven rigorously for the Boltzmann selection scheme, where in [MM01a] an algorithm for controlling β was presented. For other selection schemes it can only be shown approximatively and empirically [Mah01].

The connection between Boltzmann distributions and graphical models has been shown in a *factorization theorem* [MMO99]. If the function $f(\mathbf{x})$ is additively decomposable (that means, $f(\mathbf{x})$ can be written as the sum of some subfunctions which are defined on subvectors of $\mathbf{x}$), then the Boltzmann distribution is decomposable, too. Moreover, it adheres to a factorization which allows to sample points for the next generation. But the proof is only valid if the graphical model is chordal or – equivalently – fulfills the *running intersection property* (RIP). In this case, as is shown in [MMO99], it is indeed possible to sample efficiently from the Boltzmann distribution, and the optimum of $f(\mathbf{x})$ can be found using *FDA*.

This proof comes with a few grains of salt. First, it contains no statement about the required population size. It may be that for a sufficient estimate of the distribution the

population must be exceedingly large. Second, the proof does not hold when the RIP is violated, as is the case in many important optimization problems. Building a model fulfilling the RIP is too expensive even in sparsely connected problems [GC05].

But empirically it has been shown that *EDA* are good at solving difficult optimization problems [MM99]. For example, in [MM02] they have been successfully applied to the graph bipartitioning problem.

Today, *EDA* is a growing field of evolutionary computation. In 2005 for the first time the GECCO conference (Genetic and Evolutionary Computation Conference) as well as the CEC (Congress on Evolutionary Computation) will feature a track on *EDA*. Examples of *EDA*s and their applications can be found in [LL01].

1.2. Contribution of This Thesis

1.2.1. Maximum Entropy and Minimum Relative Entropy

As was already stated, the basic principle of *EDA* is to build a probability distribution from the selected population and use this distribution to sample points for the next population. The problem of estimating distributions arises in many different disciplines, in different guises, using a different language, and with different goals and applications.

In statistical physics and information theory, the principle for choosing a distribution is the *principle of maximum entropy* (MaxEnt) [Jay57, Jay78]. In the space of possible distributions, adhering to a set of constraints, the distribution should be chosen which maximizes the entropy, because this distribution is the only one which does not assume any additional information which by hypothesis we do not have [Jay57].

A generalization of this principle is the *principle of minimum relative entropy* (MinRel). In this case there is additionally given an objective distribution q to which we want to be as near as possible. The principle states that in the space of possible distributions, adhering to the constraints, one should choose the distribution which minimizes the relative entropy (also called the Kullback-Leibler divergence) to the distribution q.

MaxEnt and MinRel distributions can be computed using algorithms such as Iterative Proportional Fitting (IPF) [DS40] or Generalized Iterative Scaling (GIS) [DR72]. These algorithms can be combined with graphical models, too [JP95, Mey98].

MaxEnt and MinRel have recently found interest in the field of *EDA*. In [OHSM03] we have used MaxEnt for a special class of graphical models, the *polytrees* (singly connected Bayesian networks). This work is briefly revised in Sect. 5.2 and then generalized.

Another attempt to incorporate MaxEnt into *EDA* was made in [WPS+04]. This work has not yet proceeded very far. In the article some basic theorems about MaxEnt and graphical models are proven, without using the notation of probability calculus. Theorem 4.7 of this thesis generalizes their results and gives an elegant proof.

This thesis presents several ways of applying MaxEnt and MinRel within the context of *EDA* and optimization by estimating Boltzmann distributions. In the next sections we identify three areas in which *EDA* can be developed or improved using the MaxEnt or MinRel principle.

1.2.2. Improved Factorizations by Merging

Not all dependency structures can be used to generate new points. Circular dependencies are not allowed. For example, in *FDA* the factorization

$$p(\mathbf{x}) = p(x_2|x_1)p(x_3|x_2)p(x_1|x_3) \tag{1.3}$$

is not allowed. This structure is not a valid distribution, and sampling is not possible either.

So far in *FDA* a factorization was built by leaving out dependencies. In [MM99, Mah01] an algorithm is given which chooses a valid subset of the dependencies. In the above example, this algorithm might choose

$$p(\mathbf{x}) = p(x_1)p(x_2|x_1)p(x_3|x_2) \tag{1.4}$$

and drop the dependency between x_1 and x_3.

But there is another possibility. Dependency sets can be merged. In our example,

$$p(\mathbf{x}) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2) \tag{1.5}$$

is a factorization system which accounts for all dependencies. This thesis presents a heuristic algorithm which merges dependency sets in order to capture all dependencies in the data, while keeping the size of the variable sets as small as possible.

This new algorithm is applied on a 2-D grid structure. The result is a *pentavariate* factorization which is not symmetric, but is shown to be superior to the tetravariate factorization of the grid given in [Mah01].

A factorization with merge contains larger marginal probabilities. In the small example above, all the dependencies are of order 2, but the merged factorization (1.5) contains a 3-variate probability. The larger a marginal distribution is, the more noisy is its estimate from a population. Therefore particularly for small populations it is favorable to construct these distributions from smaller ones using the principle of maximum entropy. This idea leads to the new Maximum Entropy Factorized Distribution Algorithm (*MEFDA*).

1.2.3. Bethe-Kikuchi Approximation

There are possibilities to use loopy models like (1.3) in *EDA*. In statistical physics, the Bethe-Kikuchi approximation is a method to generalize graphical models to loopy structures.

On graphical models, there exist belief propagation algorithms [Pea88] which distribute knowledge or evidence throughout the model. They can be used to calculate efficiently a Boltzmann distribution on the graphical model.

On loopy structures there is a counterpart of this, a *generalized belief propagation* (GBP) algorithm which minimizes the *free energy* of the loopy graphical model, a notion known from statistical physics. If the model is not loopy, GBP is equivalent to conventional belief propagation. These algorithms have recently been developed from

the computer science perspective [AM01, MY02, YFW04]. We show that minimizing the free energy is equivalent to minimizing the relative entropy to the Boltzmann distribution [MH05].

The first attempt to use the Kikuchi method in *EDA* was done by Santana in [San04, San05]. But, as already mentioned, it is not possible to sample from these approximative distributions like it is done in *FDA*. Santana employs the Gibbs sampling procedure, an iterative scheme to draw samples from a Kikuchi distribution. The problem of this method is that it might take a large number of steps (a large *mixing time*) to produce a good sample from the distribution.

This thesis follows another track. It introduces a way to combine the Kikuchi approximation method with an *FDA* factorization which makes it possible to sample values efficiently. The algorithm can be combined with the subfunction merge algorithm of Sect. 1.2.2, which leads to a further improvement.

In the previous work [AM01, MY02, YFW04], the Boltzmann parameter β is not regarded. The reason is that they do not use GBP for optimization; in [YFW04] the optimization task is explicitly excluded. For the optimization objective β plays an important role. Therefore we generalize the free energy by varying β and investigate the effect on GBP.

The resulting algorithm, the *BKDA*, takes a step away from *EDA*, because it does not use a previous population to build a graphical model. Instead, it uses GBP to directly approximate the Boltzmann distribution, and then draws samples from it.

The performance and the dynamic properties of the GBP algorithm are investigated using a number of benchmark functions, like difficult instances of the Ising spin glass problem on a 2-D grid.

It is found that in some cases GBP has difficulties to converge. For these cases, Yuille [Yui02] has developed a Concave Convex Procedure (CCCP). The free energy is split in a concave and a convex part, and these are minimized in a double-loop algorithm. This procedure is more expensive than GBP, but guaranteed to converge to an extreme point of the free energy. It can readily replace GBP within *BKDA*.

1.2.4. Structure Learning from Data – LFDA

The third application of MaxEnt and MinRel in the context of *EDA* is learning a graphical model from the selected population. *FDA* assumes that the structure of the graphical model is given in advance. But in many cases there is no information about this structure.

A probability structure can be *learned* from data [Jor99]. When the best points are selected according to their fitness, is can be assumed that the selected set contains information about the dependencies within the data. So a graphical model can be built to capture the dependencies in the selected population. This has been applied in *EDA* in several variants [EL99, PGC99, MM99].

If the principle of minimizing the relative entropy between the empirical distribution of the selected population and the distribution induced by the graphical model is followed, the resulting measure is equivalent to maximizing the log-likelihood of the model.

Using this measure alone would result in severe overfitting; far too many edges would be added to the graphical model, learning the noise in the population distribution. Therefore an additional term is included which favors simple structures. This measure is motivated by information theory, it has been derived independently in [Sch78] as the Bayesian Information Criterion (BIC) score and in [Ris78] as the Minimal Description Length (MDL) score.

Learning a graphical model using this score was incorporated into *FDA* in [MM99]; this yields the Learning Factorized Distribution Algorithm (*LFDA*).

Recently Mühlenbein and Höns [MH05] have shown that the Bayesian network learning algorithm connects especially variables which have a high mutual information. This thesis continues the work and investigates further the properties of *LFDA*.

It is demonstrated that for large population size, the optimum of this scoring metric is the true dependency graph of the problem, so the learning algorithm reproduces the correct structure.

1.2.5. Kernighan Lin Hillclimber

An advantage of *EDA* is that it can be combined easily with a local optimizer. On each generated individual a hillclimber is applied: In a neighborhood of the point, the hillclimber searches for better points, until a local optimum is reached. A local optimum is a point which is optimal within its neighborhood. Effectively the *EDA* only works on the space of local optima of the hillclimber. So this combination reduces the search space of the *EDA*.

The Kernighan Lin hillclimber (*KLH*) is a local search heuristic, originally introduced for graph bipartitioning [KL70] and the traveling salesman problem [LK73]. The graph bipartitioning *KLH* was successfully combined with *EDA* in [Mah01, MM02].

This thesis presents a generalization of this very efficient hillclimber to arbitrary binary vector problems, which can easily be combined with *EDA*.

Furthermore, an iterative version of *KLH* is introduced: In the local optimum found by *KLH*, some random bit flips are performed, and *KLH* is applied again to this new starting point. This scheme results in the Iterated Mutation Kernighan Lin Hillclimber (*IMKLH*), which is a simple yet effective optimization algorithm; for many benchmark problems this algorithm is very successful.

The space of the local optima is analyzed for paradigmatic instances of the MAXSAT problem, in order to investigate whether and why the combination with *EDA* is fruitful. *KLH* is shown to perform in a similar scale as Walksat, a well-known specialized SAT solver algorithm.

Then, the combination of various *EDA*s with *KLH* is investigated. It is shown that only with help of a hillclimber like *KLH* the MAXSAT problem becomes tractable for *EDA*.

The hybrid algorithm of *EDA* and *KLH* is also applied on the Ising spin glass problem and on the Kaufmann (n, k) problem, in comparison with the other techniques.

1.2.6. Summary

This thesis is highly interdisciplinary. It employs results from information theory, statistical physics, probability calculus and statistics in order to understand and develop *EDA*. The methods, the principles of MaxEnt and MinRel and estimation of distributions from data are useful in many other fields, too. For example, in [MH02b, MH02a] we used these methods for stochastic analysis of cellular automata. *EDA* is a good benchmark for these methods because the optimization success is a good quality indicator.

We summarize the main contributions of this thesis. The theoretical contributions are the following:

- Maximum entropy and minimum relative entropy are identified as basic principles for estimating a distribution, and they are applied for several tasks within the context of *EDA*.
- Various graphical models for probability distributions (Bayesian networks, Markov networks, junction trees, polytrees) are presented and their relationships are clarified.
- The free energy of a loopy graphical model (as defined in statistical physics) is identified with the relative energy to the Boltzmann distribution. For optimization purpose it is generalized by incorporating the Boltzmann parameter β.
- It is shown empirically that for large population sizes, the learning algorithm of *LFDA* finds the correct dependency structure of the objective function.
- The space of local optima of *KLH* is investigated for paradigmatic 3-SAT instances. It is shown that the optima are clustered within the search space and that the combination of *EDA* and *KLH* is sensible.

Algorithmically, we obtain the following contributions:

- A heuristic algorithm is presented for building a factorization system from an additively decomposable function by merging dependency sets. This algorithm regards all dependencies between the variables. A paradigmatic example is the 2-D grid structure, for which a pentavariate factorization is introduced.
- The marginal distributions on the merged dependency sets can be constructed using the MaxEnt principle, which results in a less noisy estimate. This gives the Maximum Entropy Factorized Distribution Algorithm (*MEFDA*).
- Generalized belief propagation (GBP) on loopy graphical models is combined with an *FDA* factorization, resulting in the Bethe-Kikuchi Distribution Algorithm (*BKDA*). If GBP does not converge, it can be replaced by the CCCP algorithm.
- The Kernighan Lin Hillclimber (*KLH*) is formulated for general bit vector problems. An iterated version of this hillclimber is presented which is a simple but powerful optimization procedure. *KLH* can also be employed in order to reduce the search space of an *EDA*.

The dissertation is based on information theory. It demonstrates that the empirical estimation of distributions is an important task in many fields, how it can be solved by the principles of maximum entropy and minimum relative entropy, and how this can be exploited for optimization in the context of *EDA*.

1.3. Outline of the thesis

This thesis is organized as follows. In the following chapter, *EDA* is introduced, starting from the genetic algorithm paradigm, introducing *UMDA* and *FDA* as presented in [Mah01]. Chapter 3 presents graphical models as a tool to code dependencies between variables and compute probability distributions efficiently. The relationship between the different graphical models is clarified. Then in Chapter 4, basic notions of information theory and probability theory are described, along with the principle of maximum entropy.

The following chapters contain the main contributions of this thesis. Chapter 5 presents manipulations of a factorization graph in order to capture more dependencies in *FDA*, and the application of the maximum entropy principle, resulting in the new algorithm *MEFDA*. Chapter 6 explores loopy belief propagation and the Kikuchi approximation. It describes the setting of statistical physics, then derives generalized belief propagation (GBP) on the region graph (a loopy graphical model), and combines this with a factorization, yielding the *BKDA*. Then its behavior on several benchmark problems is investigated. CCCP is described as a more stable alternative to GBP. Chapter 7 describes how to learn a graphical model from the selected population and analyzes the result of the learning algorithm *LFDA*. The Kernighan Lin hillclimber is introduced in Chapter 8, along with an analysis of its local optima and its results on the MAXSAT, Ising and Kaufmann problem.

Chapter 9 concludes the thesis with a summary of the results and the outlook.

2. Estimation of Distribution Algorithms

2.1. The Optimization Objective

Estimation of Distribution Algorithms (*EDA*) are evolutionary algorithms. The objective of *evolutionary computation* is optimization: Given a function $f : \mathcal{D} \to \mathbb{R}$ – where $\mathcal{D}$ is a discrete[1] domain –, find $\mathbf{x}_{\max}$ with

$$f(\mathbf{x}_{\max}) = \max_{\mathbf{x} \in \mathcal{D}} f(\mathbf{x}) =: f_{\max} \tag{2.1}$$

This maximum is not necessarily unique. The function f is also called the *fitness function*.

Unless otherwise noted, we consider the domain $\mathcal{D} = \{0, 1\}^n$. It contains binary vectors of size n. Whenever necessary, we denote the variables with upper-case letters, $\mathbf{X} = (X_1, X_2, \ldots, X_n)$, and assignments by lower-case letters, $\mathbf{x} = (x_1, x_2, \ldots, x_n)$.

In this case, optimization generally requires an exponential effort in n. Without information about the function f, all values $f(\mathbf{x})$ must be computed in order to find the global optimum. But in practice heuristics are used to approximate the optimum. Among the most well-known heuristics are gradient ascent, linear programming, simulated annealing and population-based methods.

Population-based heuristics manage a collection of possible solutions $\mathfrak{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, the so-called *population*. N is the *population size*, not to be confused with n, the dimension of the problem. Prominent examples of population-based algorithms are the *genetic algorithms*, using selection, crossover and mutation. In the beginning, these were considered "optimal" by their advocates. Holland [Hol75] devised the *schema theorem*, stating that genetic algorithms in general examine subspaces of the search space with relatively high fitness.

Indeed, genetic algorithms are successful in many practical applications. However, the optimality of genetic algorithms was refuted when Goldberg [Gol87, Gol89] constructed *deceptive functions* which lead the genetic algorithms away from the global optimum, towards a local optimum. Then the *no free lunch* theorem [WM95] was formulated, stating that over the average of all possible fitness functions, all heuristics perform equally well [Cul98]. For each method that performs well on a number of tasks, there can be constructed just as many fitness functions on which it performs badly. Therefore, there is no "holy grail" optimization technique. The idea that the schema theorem provides us with an "optimal" optimization algorithm has been found to be a fallacy [Müh91, Alt95, MM01b].

[1] There are also continuous (e. g. Gaussian) models [LL01], but this thesis treats only discrete problems.

The *no free lunch* theorem has radically changed the direction of research in optimization. Earlier, researchers were looking for "the best optimization algorithm". Now they must investigate in which cases a given search algorithm performs well, or which algorithm is appropriate for a given problem.

2.2. Genetic Algorithms

There are several variants of the *Simple Genetic Algorithm* (*SGA*). An example is the following scheme:

Algorithm 2.1: Simple Genetic Algorithm

```
1  Create an initial population of size N
2  do {
3      Select N̂ ≤ N individuals according to fitness
4      Create a new population from the selected using recombination and
       mutation
5  } until stopping condition fulfilled
```

We will now consider selection, recombination and mutation in detail.

2.2.1. Selection

Selection is the only place in the algorithm in which the fitness of the individuals is considered. From the N individuals, $\hat{N}$ are chosen. They form the selected population $\hat{\mathfrak{X}} \subset \mathfrak{X}$. It is generally allowed to choose an individual repeatedly.

Now we will introduce the most common selection methods. For comparison, an example is depicted in Table 2.1. There, an example population of size $N = 10$ is shown, with the fitness function OneMax-5

$$f(\mathbf{x}) = \sum_{i=1}^{5} x_i \ , \tag{2.2}$$

the number of 1 bits. The selection probabilities are given in the table.

Proportional Selection: An individual $\mathbf{x}_i$ is selected for recombination with probability $p(\mathbf{x}_i) = \frac{f(\mathbf{x}_i)}{\sum_j f(\mathbf{x}_j)} \propto f(\mathbf{x}_i)$. A drawback of this method is that the selection pressure decreases with time: When all individuals are near the optimum, small fitness gains will not matter much. Also, it is not invariant under translation of the fitness function. Using $g(\mathbf{x}) := f(\mathbf{x}) + 1000$ will decrease the selection pressure substantially. (For the example in Table 2.1, we would get selection probabilities between 0.09978 and 0.10018, which is almost no difference.) However, it is invariant under multiplication with a constant.

$\mathbf{x}$	$f(\mathbf{x})$	Prop	Exp $\beta = 1$	Trunc $\tau = 0.3$	Tourn $T = 2$
10111	4	0.182	0.295	0.333	0.18
01111	4	0.182	0.295	0.333	0.18
10110	3	0.136	0.109	0.111*	0.13
00111	3	0.136	0.109	0.111*	0.13
11010	3	0.136	0.109	0.111*	0.13
00101	2	0.091	0.040	0	0.09
00001	1	0.045	0.015	0	0.05
01000	1	0.045	0.015	0	0.05
10000	1	0.045	0.015	0	0.05
00000	0	0	0.005	0	0.01
Sum:	22	1	1	1	1

Table 2.1.: Example population for OneMax-5 with selection probability for various selection methods. $N = 10$, $n = 5$. Precisely, this is the probability that a random member of $\hat{\mathfrak{X}}$ is equal to $\mathbf{x}$. For truncation selection, of the three individuals marked with "*" only one can be in the selected population, so they are not independent.

Exponential Selection: The selection probability is chosen proportionally to $e^{\beta f(\mathbf{x})}$. Due to the exponential growth, the selection pressure will stay high over time. This method is invariant under translation $f(\mathbf{x}) + c$, but generally not under multiplication. The parameter $\beta > 0$ is called the *inverse temperature*. It is not easy to choose a reasonable value. There exist, like in simulated annealing [KGV83], various methods for varying β within time, so-called *cooling schedules*.

Truncation Selection: Choosing a truncation threshold τ, the best $\hat{N} = \tau N$ individuals are chosen for recombination. Parents are chosen uniformly within this set. This simple method is invariant under monotonous transformations of the fitness function.

In the example table, note that there are three individuals with equal fitness 3 (marked with a "*"), so each of these will be selected with probability $1/3$. The selected set has a size of $\hat{N} = \tau N = 3$, so we get a probability of $1/9$; but in this place please note that unlike the other methods, which select all individuals independently, this one chooses one of the three "*" individuals and then stays with this one and further on disregards the two others.

Tournament Selection: Given a tournament size $T \geq 2$, T individuals are drawn randomly from the population (allowing repetition). The best one of these individuals will be selected.[2] This procedure is repeated $\hat{N}$ times. A bigger tournament size T increases the selection pressure.

[2] For equal fitness, we can just arbitrarily choose the first drawn individual. This will not create any bias.

The probability to select an individual $\mathbf{x}_i \in \mathfrak{X}$ can be calculated as follows: Let the population consist of

- N_+ individuals with better fitness than $\mathbf{x}_i$,
- $N_=$ with equal fitness (including $\mathbf{x}_i$ itself) and
- N_- individuals with worse fitness.

Then for $T = 2$ the probability to choose $\mathbf{x}_i$ for a tournament is

$$p(\text{Tourn}(\mathbf{x}_i)) = \frac{2N - 1}{N^2} \ , \tag{2.3}$$

because there are N^2 possible pairs of individuals, of which $2N - 1$ contain $\mathbf{x}_i$. (The situation is comparable to rolling two dies: There are 36 possible outcomes, of which 11 contain a six.) Remember that repetition is allowed, so $\mathbf{x}_i$ might compete against itself.

Of the $2N - 1$ combinations, there are $2N_- + N_=$ in which $\mathbf{x}_i$ wins against its competitor, so the probability that $\mathbf{x}_i$ is finally selected is

$$p(\mathbf{x}_i \text{ selected}) = \frac{2N_- + N_=}{N^2} \tag{2.4}$$

2.2.2. Recombination

Recombination (crossover) produces new individuals from the selected set. This operator is inspired from sexual reproduction in biology. There the genome of a child is a mixture of the genes of the parents. Recombination of bit strings chooses a pair of (usually) two parents and then produces a child bit string from them.

The first commonly used recombination technique was **one-point crossover**. An index k is randomly chosen between 1 and n. $x_1, \ldots, x_{k-1}$ are copied from the first parent, $x_k, \ldots, x_n$ from the second. A variant of this is **two-point crossover**, which chooses two indices, $1 \leq k < l \leq n$. Then $x_k, \ldots, x_{l-1}$ are copied from the first parent and the other bits from the second parent.

The idea behind these recombination techniques is that neighbored bits are often connected in some way to produce a high fitness. Certainly, in general there is no reason to assume this – as the *no free lunch* theorem states –, but in many real-world applications this is nevertheless the case; the bits can be arranged in such an ordering. However, this method to incorporate connections between the bits is fairly crude and poorly motivated. Meanwhile more sophisticated techniques have been developed, which do not depend on neighborhood of bits. This will be presented in section 2.4.

If there is no reason to assume connections between neighboring bits, **uniform crossover** can be used. Here, for every bit of the child one of the parents is chosen randomly, independent from the other bits.

Even one step further, away from the genetic paradigm, is the idea of **gene pool recombination** [Sys93, MV96], which does not use two parents for recombination, but the whole set of selected individuals. This is already the first step ahead, away from mimicking genetics and breeding, towards optimization by *estimation of distributions*.

2.2.3. Mutation

Mutation randomly flips bits from the children, thus increasing diversity within the population. Mutation is applied with a probability μ, called the **mutation rate**. A good rule of thumb is to choose $\mu = 1/n$ [Müh91, Müh92].

Mutation can be regarded as a local random search in the neighborhood of the children in the solution space.

2.2.4. Elitism

A very common technique in evolutionary computation is to copy the best individual of the population to the next generation, without any change. In many cases this has been found to improve performance. This technique is called **elitism**. Sometimes more than one individual is kept.

Elitism is not contained in the biological foundation of genetic algorithms (until today, the best individuals are not cloned to be contained in the next generation again). But from the computer science point of view, it is not desirable to throw away the best individual.

2.2.5. Stopping Conditions

In algorithm 2.1, the loop is performed until a "stopping condition is fulfilled". There are several possibilities for when to stop the computation:

Fixation: If all individuals are the same, the algorithm has converged and we can expect that there will be no further improvement so we can stop there.

For detecting fixation, it should be considered that in elitist runs the best individual can differ from all the others, and that there can be slight variance due to mutation. One possible definition is that for all bits, the fraction of individuals in which the bit differs from the value of the majority is lower than the mutation rate.

No progress: The maximal fitness (the fitness of the best individual) has not improved over the last Δ_t generations.

Plateau: Let $\bar{f}(t)$ be the average fitness of the population at generation t. Let there be given a fitness horizon Δ_t. The run is on a **plateau** if the average fitness is not higher than the average of the Δ_t last generations:

$$\bar{f}(t) \leq \frac{1}{\Delta_t} \sum_{\tau=t-\Delta_t+1}^{t} \bar{f}(\tau) \tag{2.5}$$

Maximum found: If the maximal fitness value is known, we can stop the run when a maximum is found.

Maximum percentage: If more stable information about the required generations is desired, we can stop when a given percentage (e. g. 10 % or 30 %) of the population has maximal fitness. This can be very different from the first hit of a maximum. E. g. in elitist runs, it might occur that the best individual sticks with the maximum, but the rest of the population converges to a suboptimal value.

On the other hand, in real-world problems it is more interesting when the first maximum is found, and one does not care about the percentage. Also, in our experiments we usually found that the standard deviation of the first hit time is rather small, so this stopping condition is not very important.

2.3. Populations and Distributions. The UMDA

Evolutionary Algorithms have been further developed to the *Estimation of Distribution Algorithms* (*EDA*). These replace the crossover and mutation operators by a probability model within the search space. In general, a probability distribution over $\mathcal{D}$ is of exponential size. For performance reasons, it is necessary to use distributions which can be stored and sampled from polynomially.

In 1996, Mühlenbein and Paaß [MP96] have proposed the *Univariate Marginal Distributions Algorithm* (UMDA), which approximates the simple genetic algorithm. It consists of a genetic algorithm with gene pool recombination and without mutation.

Algorithm 2.2: *UMDA* – Univariate Marginal Distribution Algorithm

```
1  Create an initial population of size N. t ⇐ 0.
2  do {
3      Select N̂ ≤ N individuals according to fitness
4      for i = 1 to n do {
5          Calculate marginal distribution p_i(x_i, t) from the selected individuals
6      }
7      Generate new individuals using p(x, t+1) = ∏_{i=1}^{n} p_i(x_i, t)
8      t ⇐ t + 1
9  } until stopping condition fulfilled
```

Remarkable is the new view of the genes. Choosing the bits from the parents can also be seen as sampling from a probability distribution. *UMDA* uses a very simple distribution: the product of the distributions for the single bits.

$$p(\mathbf{x}, t+1) = \prod_{i=1}^{n} p_i(x_i, t) \tag{2.6}$$

This assumes no dependencies between the bits.

It has been shown by Geiringer [Gei44] that this distribution is the limit of any reasonable recombination/crossover scheme for large populations. In population genetics, this

independent product distribution is also called *linkage equilibrium* or *Robbins' proportion* [Rob18].

Therefore *UMDA* can be seen as the limit case of the genetic algorithm. If many recombination steps are performed without selection, the result is equivalent to uniform crossover with gene pool recombination. The local effects of 1-point or 2-point crossover will disappear. Certainly this is a theoretical result only. It doesn't mean that these crossover techniques are equivalent in practice. But it is impossible to say whether a bit combination of high fitness will persist; in particular, this depends on their distance in the vector. (Neighboring bits are more probable to stay together because it is less likely that the crossover point will be between them.) One problem of *GA* is that it is very difficult to quantify and thus analyze these effects. *UMDA* is based on probability theory, and its behavior can be analyzed mathematically.

This has been done by Mühlenbein and Mahnig [MM00]. The mathematical analysis of *UMDA* extends the perception in the following way: Let there be given a fitness function

$$f : \{0,1\}^n \to \mathbb{R} \tag{2.7}$$
$$f : x_1, \ldots, x_n \mapsto f(x_1, \ldots, x_n) \tag{2.8}$$

This discrete space is now transformed into a continuous model. $x_1, \ldots, x_n$ are replaced by the univariate marginal distributions $p_1, \ldots, p_n$, i. e. the probabilities of the bits to be 1. This leads to

$$\bar{f} : [0,1]^n \to \mathbb{R} \tag{2.9}$$
$$\bar{f} : p_1, \ldots, p_n \mapsto \bar{f}(p_1, \ldots, p_n) \tag{2.10}$$

where $\bar{f}$ is the **expected fitness** of an individual sampled with the given probabilities.

$$\bar{f}(\mathbf{p}) = \sum_{\mathbf{x}} \mathbf{p}(\mathbf{x}) f(\mathbf{x}) = \sum_{\mathbf{x}} f(\mathbf{x}) \prod_i p_i(x_i) \tag{2.11}$$

Whereas f is only defined for the corners of the n-dimensional unit hypercube, $\bar{f}$ extends the domain to the interior of this hypercube. Of course, in the corners the two functions coincide.

It turns out that there are no local maxima in the interior of the hypercube, but only at the corners. *UMDA* (with fitness proportional selection) performs a gradient ascent in this space, until it converges to one of the corners. When *UMDA* reaches a corner, all probabilities are 0 or 1; then all individuals of the population are identical. This phenomenon is called **fixation**.

Since *UMDA* performs nothing but a gradient ascent, it cannot traverse valleys of the fitness landscape. Also, it does not regard couplings between the variables, since in the product distribution $p(\mathbf{x}, t+1) = \prod_{i=1}^{n} p_i(x_i, t)$ all variables are assumed to be independent.

2.4. Incorporating Dependencies. The FDA

The product distribution used by *UMDA* is too simple for fitness functions that contain strong couplings between the variables. Thus, it can be deceived and led into the wrong direction. Therefore we should consider more powerful distributions to sample from.

Such a distribution $\tilde{p}$ is determined by parameters $q(t)$ that are varied during the run. The initial population is distributed after $\tilde{p}(\mathbf{x}, q(t=0))$, usually the uniform distribution. Also, there should be a parameter estimation scheme $\tilde{P}$ for obtaining the new parameters $q(t)$ using the set of selected individuals.

For example, for *UMDA* $q(t)$ is the set of univariate marginal probabilities $p_i(x_i, t), i = 1, \ldots, n$. The parameter estimator $\tilde{P}$ calculates these as the marginal frequencies in the population.

In general, this leads to the conceptual algorithm *PDA* (Parametric Distribution Algorithm, algorithm 2.3).

Algorithm 2.3: *PDA* – Parametric Distribution Algorithm

1 Create an initial population $\mathfrak{X}$ of size N using $\tilde{p}(\mathbf{x}, q(0))$ (uniform distribution), $t \Leftarrow 1$.
2 **do** {
3 Select $\hat{N} \leq N$ points $\hat{\mathfrak{X}}$.
4 Estimate $q(t) = \tilde{P}(\hat{\mathfrak{X}}, t)$ from the selected points.
5 Generate new points using $\tilde{p}(\mathbf{x}, q(t))$.
6 $t \Leftarrow t + 1$.
7 } **until** stopping condition fulfilled

For UMDA, $\tilde{p}$ is the product distribution (2.6). A more sophisticated choice for $\tilde{p}$ is the Boltzmann distribution, presented in the following section.

2.4.1. The Boltzmann Distribution

Definition 2.1. Given a function $f(x)$ and parameter $\beta \geq 0$, the **Boltzmann distribution** (or Gibbs distribution) is defined as

$$p_{\beta,f}(x) := \frac{e^{\beta f(x)}}{\sum_y e^{\beta f(y)}} = \frac{e^{\beta f(x)}}{Z_f(\beta)} \tag{2.12}$$

β is called the inverse temperature. This stems from the analogon of thermodynamics, where the Boltzmann distribution is used with the temperature $T = \frac{1}{\beta}$.

$Z_f(\beta)$ is called the partition function. It is needed for normalization. If the indices β or f are clear from the context, they can be omitted.

The Boltzmann distribution has several properties which recommend it for optimization. First we note the following:

Lemma 2.1. *A Boltzmann distribution has the following properties:*

1. *For $\beta = 0$, $p_{\beta,f}$ is the uniform distribution, independent from f.*
2. *For $\beta > 0$, $f(x_1) < f(x_2) \Longrightarrow p_{\beta,f}(x_1) < p_{\beta,f}(x_2)$.*
3. *For $g(x) = f(x) + c$, $p_{\beta,g} = p_{\beta,f}$.*
4. *For $g(x) = c \cdot f(x)$, $p_{\beta,g} = p_{c\cdot\beta,f}$.*
5. *$p_{\beta,f}(x_1)/p_{\beta,f}(x_2) = e^{\beta(f(x_1)-f(x_2))}$.*

Proposition 2.2. *For $\beta \to \infty$, $p_{\beta,f}$ tends to the uniform distribution on the set of the maxima of f.*

Proof. Let $\mathcal{M}$ be the set of maxima, $x_m \in \mathcal{M}$ be a maximum and $x_l \notin \mathcal{M}$, so $f(x_l) < f(x_m)$. Then

$$p_{\beta,f}(x_l) = \frac{e^{\beta f(x_l)}}{\sum_y e^{\beta f(y)}} \leq \frac{e^{\beta f(x_l)}}{e^{\beta f(x_m)}} = e^{-\beta(f(x_m)-f(x_l))} \tag{2.13}$$

With increasing β, this goes to zero exponentially.

For two maxima x_1, x_2, there is always $p_{\beta,f}(x_1) = p_{\beta,f}(x_2)$. Therefore

$$\lim_{\beta\to\infty} p_{\beta,f}(x) = \begin{cases} \frac{1}{|\mathcal{M}|} & \Longleftrightarrow x \in \mathcal{M} \\ 0 & \Longleftrightarrow x \notin \mathcal{M} \end{cases} \tag{2.14}$$

□

This result is particularly promising. If we could sample efficiently from a Boltzmann distribution with arbitrary β, optimization would be an easy task. Unfortunately, the calculation of the partition function needs a summation of exponentially many terms. Nevertheless, it is very valuable in optimization, as will be shown in the following section.

2.4.2. Boltzmann Selection

Definition 2.2. Let there be given a distribution p and a selection pressure γ. **Boltzmann selection** [MT93] calculates the distribution for the selected points according to

$$p^s(x) = \frac{p(x)e^{\gamma f(x)}}{\sum_y p(y)e^{\gamma f(y)}} \tag{2.15}$$

Boltzmann selection is similar to the exponential selection scheme in genetic algorithms (see section 2.2.1). In fact, exponential selection is a special case of this, where $p(x)$ is the frequency of x in the population, and $p^s(x)$ is the probability to choose x for the selected population.

In that context the focus was on a population from which to select points, instead of a distribution p. But instead of selecting points with p^s to build a probabilistic model, you can also build the model directly using p^s. Soon we are going to present a feasible way to do this. But first, we will motivate Boltzmann selection by the following propositions.

Proposition 2.3. *Starting from an initial distribution $p_0(x)$, applying Boltzmann selection twice, with parameters β and γ, results in the distribution*

$$p^{ss}(x) = \frac{p_0(x)e^{(\beta+\gamma)f(x)}}{\sum_y p_0(y)e^{(\beta+\gamma)f(y)}} \tag{2.16}$$

Proof. Let $Z_0(\beta) = \sum_y p_0(y)e^{\beta f(y)}$ be the partition function after the first selection. After the second selection we obtain

$$p^{ss}(x) = \frac{p_0(x)e^{\beta f(x)}}{Z_0(\beta)} \frac{e^{\gamma f(x)}}{\sum_y \frac{p_0(y)e^{\beta f(y)}}{Z_0(\beta)} e^{\gamma f(y)}} \tag{2.17}$$

$$= \frac{p_0(x)e^{(\beta+\gamma)f(x)}}{\sum_y p_0(y)e^{(\beta+\gamma)f(y)}} \tag{2.18}$$

□

Corollary 2.4. *Applying Boltzmann selection with parameter γ on a Boltzmann distribution $p_{\beta,f}(x)$ results in the Boltzmann distribution*

$$p^s(x) = p_{\beta+\gamma,f}(x) \tag{2.19}$$

Proof. In the above proof, set p_0 to be the uniform distribution. □

This can also be iterated.

Corollary 2.5. *Applying Boltzmann selection on the uniform distribution iteratively with parameters $\beta_1, \ldots, \beta_T$ results in a sequence of Boltzmann distributions $p_{\bar{\beta}_t,f}$ ($t = 1, \ldots, T$) with*

$$\bar{\beta}_t = \sum_{\tau=1}^{t} \beta_\tau \tag{2.20}$$

2.4.3. The BEDA

The results of the previous section lead to an algorithm which looks for the optimum by sampling from Boltzmann distributions with increasing β.

The *BEDA* (Boltzmann Estimated Distribution Algorithm) is a conceptual algorithm only. It needs an exponential time to calculate the distributions. Therefore, it is not useful, since in exponential time one could as well search for the optimum by brute force.

The algorithm draws samples from increasingly peaked Boltzmann distributions. Since the Boltzmann distribution concentrates on the maxima for increasing β, we say that *BEDA* is **focusing** on the maxima, and the probability to sample a maximum tends to 1 (provided that the probability of the maximum is greater than zero in the starting distribution).

The distributions that are estimated from the populations contain an exponential number of parameters. Therefore, in order to estimate with sufficient accuracy, a very

Algorithm 2.4: *BEDA* – Boltzmann Estimated Distribution Algorithm

1 $t \Leftarrow 0$. Generate N individuals using the uniform distribution $p(x, 0)$ with $\beta_0 = 0$.
2 **do** {
3 Estimate $p(x, t)$ from the population
4 For a given $\beta_t > 0$, calculate
$$p^s(x,t) = \frac{p(x,t)e^{\beta_t f(x)}}{\sum_y p(y,t)e^{\beta_t f(y)}}. \tag{2.21}$$
5 Sample N new points from the distribution $p^s(x, t)$.
6 $t \Leftarrow t + 1$.
7 } **until** stopping condition fulfilled

large population is needed. If the population size is too small, the maximum might be lost in an early step, and then its probability will be zero for the rest of the run.

So, due to the exponential effort and the big sampling error, *BEDA* is not applicable. But it can be turned into a useful algorithm when the Boltzmann distribution can be split in parts of polynomial size. This is the case if there is an additive structure on the fitness function.

2.4.4. Additively Decomposable Functions and Factorization Systems

Definition 2.3. Let there be given a function $f : \{0,1\}^n \to \mathbb{R}$. An **additive decomposition** $\mathfrak{S}$ of f is a system of index sets $s_1, \ldots, s_m \subseteq \{1, \ldots, n\}$ and a set of functions $f_1(\mathbf{x}_{s_1}), \ldots, f_m(\mathbf{x}_{s_m})$, where $\mathbf{x}_{s_i}$ denotes the subvector of $\mathbf{x} \in \{0,1\}^n$ indexed by s_i, such that

$$f(\mathbf{x}) = \sum_{i=1}^{m} f_i(\mathbf{x}_{s_i}) \tag{2.22}$$

If there exists an additive decomposition of f, we call f an **additively decomposable function (ADF)**.

For an additive decomposition, we define the following:

$$\begin{aligned} d_i &:= \textstyle\bigcup_{j=1}^{i} s_j && \text{(histories)} \\ b_i &:= s_i \setminus d_{i-1} && \text{(residuals)} \\ c_i &:= s_i \cap d_{i-1} && \text{(separators)} \end{aligned} \tag{2.23}$$

with $d_0 := \emptyset$. This means that, given an ordering of the sets $\mathfrak{S} = (s_1, \ldots, s_m)$, we imagine the variables being added to a large pot in this order. Whenever adding a new s_i, the history d_i is the set of variables that are now in the pot, the residuals b_i are the variables that are new, and the separators c_i are the variables in s_i that have already been added before.

Definition 2.4. An additive decomposition $\mathfrak{S}$ is a **factorization system**, if

$$d_m = \{1, \ldots, n\} \ , \tag{2.24}$$

so that in the end all variables are contained, and

$$\forall i \in \{1, \ldots, m\}: \quad b_i \neq \emptyset \ . \tag{2.25}$$

If (2.25) is not fulfilled, we can merge some subsets (and add their f_i).

Example 2.1 (Bivariate circle). Suppose that the fitness function has a circular structure:

$$f(\mathbf{x}) = \sum_{i=1}^{n-1} f_i(x_i, x_{i+1}) + f_n(x_n, x_1) \tag{2.26}$$

The additive decomposition $\{\{1,2\},\{2,3\},\ldots,\{n-1,n\},\{n,1\}\}$ is not a factorization system, because the very last subset will have $b_m = \emptyset$. But we can merge the last two subsets and set

$$\tilde{f}_{n-1}(x_{n-1}, x_n, x_1) = f_{n-1}(x_{n-1}, x_n) + f_n(x_n, x_1) \tag{2.27}$$

and

$$f(\mathbf{x}) = \sum_{i=1}^{n-2} f_i(x_i, x_{i+1}) + \tilde{f}_{n-1}(x_{n-1}, x_n, x_1) \tag{2.28}$$

This induces the valid factorization system $\{\{1,2\},\{2,3\},\ldots,\{n-1,n,1\}\}$.

Now, let there be a probability distribution $p(\mathbf{x})$. Using the factorization system, we can define the distribution

$$\tilde{p}(\mathbf{x}) = \prod_{i=1}^{m} p(\mathbf{x}_{b_i} | \mathbf{x}_{c_i}) \tag{2.29}$$

It consists of the conditional distributions

$$p(\mathbf{x}_{b_i} | \mathbf{x}_{c_i}) = \frac{p(\mathbf{x}_{c_i}, \mathbf{x}_{b_i})}{\sum_{\tilde{\mathbf{x}}_{b_i}} p(\mathbf{x}_{c_i}, \tilde{\mathbf{x}}_{b_i})} \ . \tag{2.30}$$

Lemma 2.6. *$\tilde{p}(\mathbf{x})$ has the following properties:*

1. *$\tilde{p}(\mathbf{x})$ is a probability distribution.*
2. *For all i, $\tilde{p}(\mathbf{x}_{b_i} | \mathbf{x}_{c_i}) = p(\mathbf{x}_{b_i} | \mathbf{x}_{c_i})$.*

Proof. It is obvious that for all $\mathbf{x}$, $\tilde{p}(\mathbf{x}) \geq 0$. So for it to be a probability distribution, we must show that it sums up to 1. The sum over all $\mathbf{x}$ can be split up in a sum over

all $\mathbf{x}_{b_i}$. The factors that to not depend on $\mathbf{x}_{b_i}$ can be moved outside the sum, and then the sum over all $\mathbf{x}_{b_i}$ is 1, since $p(\mathbf{x}_{b_i}|\mathbf{x}_{c_i})$ is a probability distribution.

$$\sum_{\mathbf{x}} \tilde{p}(\mathbf{x}) = \sum_{\mathbf{x}} \prod_{i=1}^{m} p(\mathbf{x}_{b_i}|\mathbf{x}_{c_i}) \tag{2.31}$$

$$= \sum_{\mathbf{x}_{b_1}} p(\mathbf{x}_{b_1}) \sum_{\mathbf{x}_{b_2}} p(\mathbf{x}_{b_2}|\mathbf{x}_{c_2}) \cdots \underbrace{\sum_{\mathbf{x}_{b_m}} p(\mathbf{x}_{b_m}|\mathbf{x}_{c_m})}_{=1} \tag{2.32}$$

$$= \cdots = 1 \tag{2.33}$$

For the second part, we split up $\bar{s}_i$ into the antecessors $a_i := d_i \setminus s_i = d_{i-1} \setminus c_i$ and the successors (followers) $f_i := \{1, \ldots, n\} \setminus d_i = \cup_{j=i+1}^{m} b_j$. Then $\{1, \ldots, n\}$ is the disjunct union $a_i \dot{\cup} b_i \dot{\cup} c_i \dot{\cup} f_i$, and

$$\tilde{p}(\mathbf{x}_{b_i}|\mathbf{x}_{c_i}) = \frac{\tilde{p}(\mathbf{x}_{b_i}, \mathbf{x}_{c_i})}{\tilde{p}(\mathbf{x}_{c_i})} \tag{2.34}$$

$$= \frac{\sum_{\tilde{\mathbf{x}}_{a_i}, \tilde{\mathbf{x}}_{f_i}} \tilde{p}(\tilde{\mathbf{x}}_{a_i}, \mathbf{x}_{b_i}, \mathbf{x}_{c_i}, \tilde{\mathbf{x}}_{f_i})}{\sum_{\tilde{\mathbf{x}}_{a_i}, \tilde{\mathbf{x}}_{b_i}, \tilde{\mathbf{x}}_{f_i}} \tilde{p}(\tilde{\mathbf{x}}_{a_i}, \tilde{\mathbf{x}}_{b_i}, \mathbf{x}_{c_i}, \tilde{\mathbf{x}}_{f_i})} \tag{2.35}$$

$$= \frac{\sum_{\tilde{\mathbf{x}}_{a_i}} \prod_{j=1}^{i-1} p(\tilde{\mathbf{x}}_{b_j}|\mathbf{x}_{c_j \cap c_i}, \tilde{\mathbf{x}}_{c_j \setminus c_i}) p(\mathbf{x}_{b_i}|\mathbf{x}_{c_i}) \sum_{\tilde{\mathbf{x}}_{f_i}} \prod_{j=i+1}^{m} p(\tilde{\mathbf{x}}_{b_j}|\mathbf{x}_{c_j \cap s_i}, \tilde{\mathbf{x}}_{c_j \setminus s_i})}{\sum_{\tilde{\mathbf{x}}_{a_i}} \prod_{j=1}^{i-1} p(\tilde{\mathbf{x}}_{b_j}|\mathbf{x}_{c_j \cap c_i}, \tilde{\mathbf{x}}_{c_j \setminus c_i}) \sum_{\tilde{\mathbf{x}}_{b_i}} p(\tilde{\mathbf{x}}_{b_i}|\mathbf{x}_{c_i}) \sum_{\tilde{\mathbf{x}}_{f_i}} \prod_{j=1}^{i-1} p(\tilde{\mathbf{x}}_{b_j}|\mathbf{x}_{c_j \cap c_i}, \tilde{\mathbf{x}}_{c_j \setminus c_i})} \tag{2.36}$$

Here, $\tilde{\mathbf{x}}$ denotes summation variables. Note that the c_j are split up in the variables that are summed over (so they are adorned with a tilde), and those that are not.

Like in the first part of the lemma, the sum over all $\tilde{\mathbf{x}}_{f_i}$ is equal to 1 and therefore drops out. The same holds for the sum over $\tilde{\mathbf{x}}_{b_i}$ in the denominator. The sum over $\tilde{\mathbf{x}}_{a_i}$ cancels out, so we are left with

$$\tilde{p}(\mathbf{x}_{b_i}|\mathbf{x}_{c_i}) = p(\mathbf{x}_{b_i}|\mathbf{x}_{c_i}) \tag{2.37}$$

□

This lemma can be used to sample points from $\tilde{p}(\mathbf{x})$. For this, we proceed through all the subsets and sample values for $\mathbf{x}_{b_i}$ from $p(\mathbf{x}_{b_i}|\mathbf{x}_{c_i})$, given the values for $\mathbf{x}_{c_i}$ that have already been sampled. This is possible because by construction there are no cycles.

In the literature, this sampling method has been called *Probabilistic Logic Sampling* (PLS) [Hen88][3].

However, note that in general

$$\tilde{p}(\mathbf{x}) \neq p(\mathbf{x}) \ , \tag{2.38}$$

and also $\tilde{p}(\mathbf{x}_{b_i}, \mathbf{x}_{c_i}) \neq p(\mathbf{x}_{b_i}, \mathbf{x}_{c_i})$. For this to be valid, two requirements must be fulfilled:

[3] This name seems unfortunate, because it has nothing to do with probabilistic logic [Nil86], the generalization of logic under uncertainty.

1. The probability distribution must be compatible with the factorization system.
2. The *running intersection property* must be fulfilled.

We will explain these two requirements in the following sections.

2.4.5. Boltzmann Distributions and Factorization Systems

To make a connection between a factorization system and a probability distribution, we want to achieve that the dependencies of the variables in the probability distribution accord with the structure of the factorization system. A thorough discussion of what we mean by the dependencies of the variables will be given in section 3 about graphical models.

For the moment, we recall our problem: We are given a fitness function f which we know to be additively decomposable. Then a good choice of the probability distribution is the Boltzmann distribution.

$$p(\mathbf{x}) = \frac{e^{\beta f(\mathbf{x})}}{Z} \tag{2.39}$$

$$= \frac{1}{Z} \prod_{i=1}^{m} e^{\beta f_i(\mathbf{x}_{s_i})} \tag{2.40}$$

For an additively decomposable function, the Boltzmann distribution is multiplicatively decomposable. Furthermore, for each multiplicatively decomposable distribution, we can define an ADF for which it is the Boltzmann distribution.

We will soon show that it is possible to sample from this distribution efficiently. Thus we will turn *BEDA* into a feasible algorithm.

2.4.6. The Running Intersection Property

Definition 2.5. A factorization system fulfills the **running intersection property (RIP)** [Lau96], if

$$\forall i \geq 2 \quad \exists j < i : \quad c_i \subseteq s_j \tag{2.41}$$

This means that all the variable sets on which new variables are conditioned must be contained in a previous subset. It may be difficult to understand the implications of this requirement. First we give an example.

Example 2.2. Let the factorization system be $\mathfrak{S} = (\{1,2\},\{1,3\},\{2,3,4\})$. It does not fulfill the RIP. The probability distribution is

$$\tilde{p}(\mathbf{x}) = \tilde{p}(x_1, x_2)\tilde{p}(x_3|x_1)\tilde{p}(x_4|x_2, x_3) \tag{2.42}$$

Suppose now that we learn the marginal probabilities using the following data set:

$$\{0011, 1100, 0101, 0010, 1110\}$$

When sampling a new point using lemma 2.6, we begin with x_1, x_2. With probability $2/5$, we obtain $0, 0$. For x_3, given that $x_1 = 0$, the probability for another 0 is $1/3$. Supposing that we have sampled these values, we are now in trouble trying to sample x_4: The combination $(x_2, x_3) = (0, 0)$ does not appear in the data!

This sort of undefined conditional probabilities can only occur when the RIP is violated. Pragmatically, in absence of any information, we then sample from the uniform distribution. This can be justified within the theory of Bayesian prior distributions (see section 4.3.3).

This undefined probability is certainly an extreme case. But also otherwise, when the RIP is violated the marginal distributions of $\tilde{p}$ are different from those which were used to construct it. An example for this is presented in Sect. 5.2.3.

On the other hand, when the RIP is fulfilled, our task is much easier. We even have the following proposition:

Proposition 2.7. *Given an ADF $f(\mathbf{x})$, whose factorization satisfies the RIP, the maximum of $f(\mathbf{x})$ can be found in polynomial time (provided that the size of the subsets is at most logarithmic in n) by dynamic programming.*

Proof. We describe an algorithm for finding the maximum of f. Analogous algorithms can be found in [BB72, Jor99].

The running intersection property states that for every $i \geq 2$ there is a $j < i$ with $c_i \subseteq s_j$. We call j the parent of i, $j = \pi(i)$, and i a child of j. The set of children is defined as $C(j) = \{i > j | j = \pi(i)\}$. If there is more than one possible j, we choose the parent arbitrarily. This forms a tree from the factorization.

Within the tree, we can now pass messages from the children up to the parents. Let $j = \pi(i)$, then we define

$$m_{ij}(\mathbf{x}_{c_i}) := \max_{\mathbf{x}_{b_i}} \left(f_i(\mathbf{x}_{s_i}) + \sum_{k \in C(i)} m_{ki}(\mathbf{x}_{c_k}) \right) \tag{2.43}$$

Having computed all the messages, the maximum can be computed using the root node s_1:

$$\max f(\mathbf{x}) = \max_{\mathbf{x}_{s_1}} \left(f_1(\mathbf{x}_{s_1}) + \sum_{k \in C(1)} m_{k1}(\mathbf{x}_{c_k}) \right) \tag{2.44}$$

The maximum configuration $\mathbf{x}_{\max} = \operatorname{argmax} f(\mathbf{x})$ can be computed by traversing from the root down to the leaves. On the way the variables are fixed upon the maximum values. So, given that the values of the parent node j have already been fixed (and thus also the values of $\mathbf{x}_{c_i}$), the residual variables of the child i are chosen as

$$\mathbf{x}_{b_i} = \operatorname*{argmax}_{\mathbf{x}_{b_i}} \left(f_i(\mathbf{x}_{s_i}) + \sum_{k \in C(i)} m_{ki}(\mathbf{x}_{c_k}) \right) \tag{2.45}$$

If there is more than one possibility with equal maximal probability, one of them can be chosen arbitrarily.

If there is more than one root node (with $c_i = \emptyset$), the maximum is just the sum of the maxima for all the roots. The running intersection property guarantees that the trees from the different roots are disjunct.

Considering the complexity of the algorithm, we note that the messages are of size $2^{|c_i|}$, and the maximization over all assignments of $\mathbf{x}_{b_i}$ requires to compute $2^{|b_i|}$ values. This is polynomial if all $|s_i|$ are of order $O(\log n)$. □

So, when the RIP is satisfied, the optimization task is easy. In the next section we will show that in this case the Boltzmann distribution can be factorized, too.

2.4.7. The Factorization Theorem

For the proof of the factorization theorem, we first need the following lemma:

Lemma 2.8. *Let $X_\alpha, X_\beta, X_\gamma$ be three disjunct sets of variables, and let p be a probability distribution that satisfies*

$$p(x_\alpha, x_\beta, x_\gamma) = g(x_\alpha, x_\gamma)h(x_\beta, x_\gamma) \tag{2.46}$$

for some functions g, h. Then the following equations hold:

$$p(x_\alpha, x_\beta, x_\gamma) = p(x_\alpha, x_\gamma)p(x_\beta|x_\gamma) \tag{2.47}$$
$$p(x_\alpha, x_\gamma) = g(x_\alpha, x_\gamma)\tilde{h}_\gamma(x_\gamma) \tag{2.48}$$
$$p(x_\alpha, x_\beta|x_\gamma) = p(x_\alpha|x_\gamma)p(x_\beta|x_\gamma) \tag{2.49}$$

Proof. First we calculate the following marginal distributions:

$$p(x_\alpha, x_\gamma) = g(x_\alpha, x_\gamma) \sum_{x_\beta} h(x_\beta, x_\gamma) \tag{2.50}$$
$$p(x_\beta, x_\gamma) = h(x_\beta, x_\gamma) \sum_{x_\alpha} g(x_\alpha, x_\gamma) \tag{2.51}$$
$$p(x_\beta|x_\gamma) = \frac{h(x_\beta, x_\gamma) \sum_{x_\alpha} g(x_\alpha, x_\gamma)}{\sum_{\tilde{x}_\beta} h(\tilde{x}_\beta, x_\gamma) \sum_{x_\alpha} g(x_\alpha, x_\gamma)} = \frac{h(x_\beta, x_\gamma)}{\sum_{\tilde{x}_\beta} h(\tilde{x}_\beta, x_\gamma)} \tag{2.52}$$

Then (2.47) follows from (2.50) and (2.52). Equation (2.49) is derived by dividing (2.47) by $p(x_\gamma)$. Finally (2.48) follows from (2.50) with $\tilde{h}_\gamma(x_\gamma) = \sum_{x_\beta} h(x_\beta, x_\gamma)$. □

Remark 2.1. If a probability distribution satisfies (2.46), we call the variable sets X_α and X_β **conditionally independent given** X_γ. We write $X_\alpha \perp\!\!\!\perp X_\beta | X_\gamma$. This definition will be resumed in the theory of graphical models in Chapter 3.

Now we can prove the essential theorem.

Theorem 2.9 (Factorization Theorem). *Let* $p(\mathbf{x}) = \frac{e^{\beta f(\mathbf{x})}}{Z}$ *be a Boltzmann distribution for the ADF*

$$f(\mathbf{x}) = \sum_{i=1}^{m} f_i(\mathbf{x}_{s_i}) \tag{2.53}$$

If the factorization system fulfills the running intersection property (2.41)*, then*

$$p(\mathbf{x}) = \prod_{i=1}^{m} p(\mathbf{x}_{b_i}|\mathbf{x}_{c_i}) \tag{2.54}$$

Proof. The proof follows [Mah01], via inverse induction over i, from m down to 1.

- We begin the induction with $i = m$. For this case it is obvious that

$$p(\mathbf{x}) = p(\mathbf{x}_{d_m}) \tag{2.55}$$
$$p(\mathbf{x}_{d_m}) = F_{s_1}^m(\mathbf{x}_{s_1}) \cdots F_{s_m}^m(\mathbf{x}_{s_m}) \tag{2.56}$$

  Equation (2.56) results from (2.40) with $F_{s_1}^m(\mathbf{x}_{s_1}) = e^{\beta f_1(\mathbf{x}_{s_1})}/Z$ and $F_{s_j}^m(\mathbf{x}_{s_j}) = e^{\beta f_j(\mathbf{x}_{s_j})}$ for $j > 1$.

- The induction claim is that for i

$$p(\mathbf{x}) = p(\mathbf{x}_{d_i})p(\mathbf{x}_{b_{i+1}}|\mathbf{x}_{c_{i+1}}) \cdots p(\mathbf{x}_{b_m}|\mathbf{x}_{c_m}) \tag{2.57}$$
$$p(\mathbf{x}_{d_i}) = F_{s_1}^i(\mathbf{x}_{s_1}) \cdots F_{s_i}^i(\mathbf{x}_{s_i}) \tag{2.58}$$

  Then we must show that this holds also for $i - 1$, i. e.

$$p(\mathbf{x}) = p(\mathbf{x}_{d_{i-1}})p(\mathbf{x}_{b_i}|\mathbf{x}_{c_i}) \cdots p(\mathbf{x}_{b_m}|\mathbf{x}_{c_m}) \tag{2.59}$$
$$p(\mathbf{x}_{d_{i-1}}) = F_{s_1}^{i-1}(\mathbf{x}_{s_1}) \cdots F_{s_{i-1}}^{i-1}(\mathbf{x}_{s_{i-1}}) \tag{2.60}$$

  for some functions F^{i-1}.

- We begin by splitting up (2.58)

$$p(\mathbf{x}_{d_i}) = \underbrace{F_{s_1}^i(\mathbf{x}_{s_1}) \cdots F_{s_{i-1}}^i(\mathbf{x}_{s_{i-1}})}_{g(x)} \underbrace{F_{s_i}^i(\mathbf{x}_{s_i})}_{h(x)} \tag{2.61}$$

  Then we can apply Lemma 2.8 with the partition $\alpha = d_{i-1} \setminus c_i$, $\beta = b_i$ and $\gamma = c_i$, yielding

$$(2.47) \Longrightarrow \qquad p(\mathbf{x}_{d_i}) = p(\mathbf{x}_{d_{i-1}})p(\mathbf{x}_{b_i}|\mathbf{x}_{c_i}) \tag{2.62}$$
$$(2.48) \Longrightarrow \qquad p(\mathbf{x}_{d_{i-1}}) = F_{s_1}^i(\mathbf{x}_{s_1}) \cdots F_{s_{i-1}}^i(\mathbf{x}_{s_{i-1}})\tilde{h}_{c_i}(\mathbf{x}_{c_i}) \tag{2.63}$$

  The RIP states that there must be a $j < i$ with $c_i \subseteq s_j$. So we can define the functions F^{i-1} as

$$F_{s_k}^{i-1}(\mathbf{x}_{s_k}) := \begin{cases} F_{s_k}^i(\mathbf{x}_{s_k}) & \text{for } k \neq j \\ F_{s_k}^i(\mathbf{x}_{s_k})\tilde{h}_{c_i}(\mathbf{x}_{c_i}) & \text{for } k = j \end{cases} \tag{2.64}$$

incorporating $\tilde{h}$ in the subfunction that contains all the variables on which it depends. This definition guarantees that

$$p(\mathbf{x}_{d_{i-1}}) = F_{s_1}^{i-1}(\mathbf{x}_{s_1}) \cdots F_{s_{i-1}}^{i-1}(\mathbf{x}_{s_{i-1}}) \tag{2.65}$$

This proves (2.60). By inserting (2.62) in (2.57), we also obtain (2.59). So, the induction step is proven.

(2.54) follows from (2.59) with $i = 1$. □

2.4.8. FDA

Suppose that we know a factorization system for the function f. Then we can use the factorization theorem to turn *BEDA* into a feasible optimization algorithm. This algorithm [MM98, MMO99] is called *FDA* (Factorized Distribution Algorithm).

Algorithm 2.5: *FDA* – Factorized Distribution Algorithm

1. Calculate b_i and c_i from the decomposition of the function.
2. $t \Leftarrow 1$. Generate an initial population with N individuals from the uniform distribution.
3. **do** {
4. Perform selection
5. Estimate the conditional probabilities $p(\mathbf{x}_{b_i}|\mathbf{x}_{c_i}, t)$ from the selected points.
6. Generate new points according to $p(\mathbf{x}, t+1) = \prod_{i=1}^{m} p(\mathbf{x}_{b_i}|\mathbf{x}_{c_i}, t)$.
7. $t \Leftarrow t + 1$.
8. } **until** stopping criterion reached

BEDA uses Boltzmann selection. So *FDA* with Boltzmann selection also calculates a series of Boltzmann distributions with increasing β, and it inherits the focusing property (see Sect. 2.4.2). But it can also be used with any other selection scheme. The most common choices are truncation selection and tournament selection (see Sect. 2.2.1).

Although the factorization theorem is only valid when the RIP is satisfied, *FDA* can also be applied when it is violated. Lemma 2.6 assures that we sample from a proper distribution. In fact, this is the case in which *FDA* is most interesting, because when the RIP is fulfilled, dynamic programming (see Prop. 2.7) is very efficient in finding an optimum.

A difficulty of *FDA* is the choice of the population size. If it is too small, the estimate of $p(\mathbf{x}_{b_i}|\mathbf{x}_{c_i}, t)$ will be bad.

The algorithm is stochastic. It requires to estimate the marginal distributions with sufficient precision. The precision of the estimation depends especially on the population size.

2.4.9. From additive decomposition to a factorization system

FDA needs a factorization of f. Given an ADF f, we need two things to turn the additive decomposition into a factorization system:

1. An ordering of the subsets and
2. a way to handle $b_i = \emptyset$ (see Sect. 2.4.4).

Given an additive decomposition $\mathfrak{S} = \{s_1, \ldots, s_m\}$, it is often not possible to order the s_i in a factorization system, because this results in a $b_i = \emptyset$; that is, a set whose variables are all already contained in previous sets. Take for example the decomposition $(\{1,2\},\{1,3\},\{2,3\})$. No matter which ordering we choose, the third set will be fully contained in the union of the previous sets.

In this case, it is in general not possible to sample from the system in a consistent way. One possibility to solve this problem is to choose only a subset of the s_i and disregard the others; in our example, we can use the factorization $(\{1,2\},\{1,3\})$. To choose a subset $\tilde{\mathfrak{S}} = \{\tilde{s}_j | 1 \leq j \leq \tilde{m}\} \subseteq \mathfrak{S}$ which is a factorization system, it is desirable to use as many connections between the variables as possible. This is implemented in the heuristics of Alg. 2.6.

Algorithm 2.6: Subfunction Choice

1. Choose the first set $\tilde{s}_1$ randomly among the s_i.
2. Then, choose among the s_i the set that has the maximal overlap with $\tilde{d}_{j-1}$ (the history of variables added so far). If there is more than one possible set, choose the smallest one.
3. Add this set as $\tilde{s}_j$. $j \Leftarrow j+1$
4. Repeat this until the whole set $\{1, \ldots, n\}$ is covered.

The number of added connections is the size of the overlap $|\tilde{c}_j| = |\tilde{s}_j \cap \tilde{d}_{j-1}|$. This is maximized by the algorithm. Furthermore, we minimize $|\tilde{b}_j|$, the number of new variables, so that the total number of incorporated subfunctions $|\tilde{\mathfrak{S}}|$ is as large as possible.

This algorithm was already presented in [Mah01], but the version given here contains a slight improvement: In [Mah01], the overlap $|\tilde{c}_j|$ was maximized, too, but in case of a tie, $|\tilde{b}_j|$ was maximized.

To motivate the decision to minimize rather than maximize $|\tilde{b}_j|$, take for example the decomposition

$$(\{1,2\},\{2,3\},\{3,4\},\{2,4,5\}) \quad . \tag{2.66}$$

The algorithm presented by Mahnig would first add $\{1,2\}$. Then there are two sets with an overlap of one variable: $\{2,3\}$ and $\{2,4,5\}$. Mahnig would add the larger one, so $\{2,4,5\}$ is added to $\tilde{\mathfrak{S}}$ next. Finally, there are the sets $\{2,3\}$ and $\{3,4\}$ left, both having an overlap of 1 and a size of 2, so one of them would be added arbitrarily, say $\{2,3\}$. The complete factorization system is

$$\tilde{\mathfrak{S}} = (\{1,2\},\{2,4,5\},\{2,3\}) \quad , \tag{2.67}$$

dropping the dependency $\{3, 4\}$. The probability distribution for this factorization system is

$$p(\mathbf{x}) = p(x_1, x_2)p(x_4, x_5|x_2)p(x_3|x_2) \ . \tag{2.68}$$

On the other hand, if $|\tilde{b}_j|$ is minimized, the second added set is $\{2, 3\}$, followed by $\{3, 4\}$ and finally $\{2, 4, 5\}$. This results in the factorization system

$$\tilde{\mathfrak{S}} = (\{1, 2\}, \{2, 3\}, \{3, 4\}, \{2, 4, 5\}) \ , \tag{2.69}$$

which yields the distribution

$$p(\mathbf{x}) = p(x_1, x_2)p(x_3|x_2)p(x_4|x_3)p(x_5|x_2, x_4) \ . \tag{2.70}$$

This distribution is more accurate than (2.68). It contains all dependencies between the variables.

Sect. 5.1.1 presents a further improvement, which instead of choosing a subset of the subfunctions merges them together. This allows to consider more dependencies or even all of them.

2.5. Summary

In this chapter, we presented population-based optimization techniques, first of all the genetic algorithm. Then we introduced *EDA*, which use probability distributions for optimization. Crossover in the genetic algorithm is replaced by building a probabilistic model and sampling data from it. The simple product distribution is used in the *UMDA*. Then, dependencies between the variables are exploited in the *FDA*, which uses a factorized probability distribution for optimization of additively decomposable functions.

The role of the Boltzmann distribution as a useful probability distribution for *EDA* is illuminated. It can also be used for selection (Boltzmann selection).

A special class of factorizations are those which fulfill the running intersection property. For these, optimization is a simple task. But the *FDA* works also when the RIP is violated.

Finally, a way to build a factorization system from an additively decomposable function is described which chooses a subset of the variable connections.

3. Graphical Models

The basic idea of *EDA* is to use the selected population for building a probabilistic model, and then sample from this model the next generation of individuals.

The most well-known probabilistic models are the graphical models, which are introduced in this chapter. We will introduce particularly *Markov networks*, *Bayesian networks*, and *junction trees*, and show their connection with factorization systems, presented in the previous chapter.

But first, we note that the main incentive of using graphical models is

- to code dependencies or independencies of variables and
- to exploit this information for the efficient calculation of or sampling from distributions.

So, we start the chapter with some considerations about independence.

Then, in Sect. 3.2, we introduce Markov and Bayesian networks, and in Sect. 3.3 the junction tree. Finally, we explore the relationship between factorization systems and junction trees, explain how to build a junction tree and present inference in junction trees.

3.1. Independence and Conditional Independence

The notion of independence is well-known from probability calculus:

Definition 3.1. Two random variables A and B are **independent**, if for all values a, b of A, B we have $p(A = a, B = b) = p(A = a)p(B = b)$. We write $A \perp\!\!\!\perp B$.

So, if we know that two variables A, B are independent, we can replace $p(A, B)$ – having 3 degrees of freedom – by $p(A)$ and $p(B)$ with one degree of freedom each, thus saving one parameter.

However, only the information which variables are independent will not bring us very far. For example, if we have three variables which are pairwise independent, it may be that all three are dependent, i.e.

$$(A \perp\!\!\!\perp B) \wedge (A \perp\!\!\!\perp C) \wedge (B \perp\!\!\!\perp C) \not\Rightarrow p(a, b, c) = p(a)p(b)p(c) \tag{3.1}$$

Example 3.1. This example is taken from [Mah01] (p. 88), correcting an obvious mistake in the probability table.

Let the probability distribution for three binary variables A, B, C be given by the following table:

abc	000	001	010	011	100	101	110	111
$p(a,b,c)$	$1/4$	0	0	$1/4$	0	$1/4$	$1/4$	0

(3.2)

Each variable has probability $1/2$ of being 1, and each pair of variables is uniformly distributed with probability $1/4$ for each pair of values. But the complete distribution is not uniform. The three variables are dependent.

For decomposing probability distributions, the following notion is more useful.

Definition 3.2. Two probability variables A and B are **conditionally independent**, given a third variable C, if

$$p(A, B|C) = p(A|C)p(B|C) \tag{3.3}$$

We write $A \perp\!\!\!\perp B|C$. The same notion is used for A, B, C being sets of probability variables.

Lemma 3.1. *If $A \perp\!\!\!\perp B|C$,*

$$p(A|B, C) = p(A|C) \tag{3.4}$$

Proof. Divide (3.3) by $p(B|C)$. □

More properties of conditional independency have already been given in Lemma 2.8.

Note that neither does independence imply conditional independence nor vice versa.

Example 3.2 ([Bré99], p. 12). Suppose that there are two factories that produce watches. Factory A produces expensive watches with a failure probability of 0.01. Factory B produces cheap watches with a failure probability of 0.2. Suppose that you receive a box full of watches, coming all from the same factory.

You take two watches from the box and check whether they work. Given the information that the watches come from factory A (or B), the events that the two watches work are conditionally independent, because all watches are produced independently.

However, the events W_1 = "first watch works" and W_2 = "second watch works" are dependent. If the probability that they come from factory A is 0.5, the probability $p(W)$ that a watch from the box works is

$$p(W) = p(W|A)p(A) + p(W|B)p(B) = 0.895 \tag{3.5}$$

But the probability that two watches W_1, W_2 from the box work is

$$\begin{aligned} p(W_1, W_2) &= p(W_1, W_2|A)p(A) + p(W_1, W_2|B)p(B) && (3.6)\\ &= p(W_1|A)p(W_2|A)p(A) + p(W_1|B)p(W_2|B)p(B) && (3.7)\\ &= 0.81005 && (3.8)\\ &\neq p(W_1)p(W_2) && (3.9) \end{aligned}$$

So we see that $W_1 \perp\!\!\!\perp W_2|A \not\Rightarrow W_1 \perp\!\!\!\perp W_2$.

Example 3.3. Let there be a house equipped with an alarm which can be triggered by a burglar or an earthquake. Suppose that an earthquake has the probability $p(E) = 0.001$. Suppose further that a burglary has the probability of $p(B) = 0.01$. For the alarm to go off, we assume the following probabilities: $p(A|B,E) = 0.95$, $p(A|\bar{B},E) = 0.9$, $p(A|B,\bar{E}) = 0.8$, $p(A|\bar{B},\bar{E}) = 0$. ($\bar{B}$ denotes the event "no burglary", $\bar{E}$ "no earthquake".) The events "earthquake" and "burglary" are independent.

This information suffices to calculate the complete table of probability for the three events, which is given in Table 3.1.

Alarm	Burglar	Earthquake	Probability
no	no	no	0.98901
no	no	yes	0.000099
no	yes	no	0.001998
no	yes	yes	0.0000005
yes	no	no	0
yes	no	yes	0.000891
yes	yes	no	0.007992
yes	yes	yes	0.0000095
		Sum:	1

Table 3.1.: The complete probability table for the alarm example

Using this table, we can calculate all probabilities in which we are interested. For example, supposing that the alarm went off, the probability of a burglary is

$$p(B|A) = \frac{0.007992 + 0.0000095}{0.000891 + 0.007992 + 0.0000095} = 0.899803 \tag{3.10}$$

whereas an earthquake has the probability

$$p(E|A) = \frac{0.000891 + 0.0000095}{0.000891 + 0.007992 + 0.0000095} = 0.101265 \tag{3.11}$$

However, the probability of an earthquake *and* a burglary in the case of an alarm is

$$p(B,E|A) = \frac{0.0000095}{0.000891 + 0.007992 + 0.0000095} = 0.001068 \tag{3.12}$$

$$\neq p(B|A)p(E|A) = 0.091188 \tag{3.13}$$

So we see that the two events *earthquake* and *burglary*, although independent, are conditionally dependent, given *alarm*, or: $B \perp\!\!\!\perp E \not\Rightarrow B \perp\!\!\!\perp E|A$.

3.2. Graphical Models

The idea of graphical models is to encode conditional dependencies graphically. This provides us with tools to calculate probabilities like in Ex. 3.3 efficiently and automatically, without having to calculate an exponentially large table like Table 3.1. Another goal is that whenever new information (evidence) enters, it can be included into the model, and the probabilities can automatically be updated.

3.2.1. Markov Networks

The first straightforward idea is to create a graph $G = (V, E)$ with $V = \{X_i, i = 1, \ldots, n\}$ and an edge between X_i and X_j if they are conditionally dependent, given the rest of the nodes.

Definition 3.3. Let p be a probability distribution on the variables $X_1, \ldots, X_n$. A **Markov network** is an undirected graph $G = (V, E)$ with $V = \{X_i, i = 1, \ldots, n\}$ and

$$(X_i, X_j) \notin E \iff X_i \perp\!\!\!\perp X_j | (V \setminus \{X_i, X_j\}) \tag{3.14}$$

So an edge means conditional dependence, a missing edge means conditional independence.

We call (3.14) the **Markov property**.

Remark 3.1. In the literature [Lau96, Whi90], this is called the *pairwise Markov property*. They also define the *local Markov property* (all variables are conditionally independent of their non-neighbors, given their neighbors) and the *global Markov property* (sets of variables are conditionally independent, given a set that separates them, meaning that every path between them must lead through the separating set). It can be shown that global induces local and local induces pairwise Markov property. If the distribution is not degenerated in a special sense, all three are equivalent. For our purposes, this distinction is not necessary, so we will use the weakest definition.

The probability function of a Markov network with the local Markov property is also called a *Markov random field*. This is often defined as a probability distribution with

$$p(x_k | \mathbf{x} \setminus x_k) = p(x_k | bd(x_k)) \tag{3.15}$$

where $bd(x_k)$ is the boundary of x_k, the set of its neighbors in the graph G. Obviously, (3.15) denotes exactly conditional independence; this the local Markov property.

3.2.2. Markov Random Fields and Additive Decomposability

Proposition 3.2. *The Boltzmann distribution for an additively decomposable function is a Markov random field. In the Markov network two variables are connected iff they appear together in a subfunction of the ADF.*

Proof. The Boltzmann distribution for an ADF is given by (2.40). We will use this for calculating the conditional probability of (3.15).

$$p(x_k | \mathbf{x} \setminus x_k) = \frac{p(x_k, \mathbf{x} \setminus x_k)}{\sum_{\tilde{x}_k} p(\tilde{x}_k, \mathbf{x} \setminus x_k)} \tag{3.16}$$

$$= \frac{\frac{1}{Z} \prod_i e^{\beta f_i(\mathbf{x}_{s_i})}}{\sum_{\tilde{x}_k} \frac{1}{Z} \prod_i e^{\beta f_i(\tilde{\mathbf{x}}_{s_i})}} \tag{3.17}$$

Now Z as well as all factors in which x_k does not appear cancel out. So we see that this probability depends only on the variables which appear in a factor with x_k. This is exactly the neighborhood structure imposed in the proposition. □

3.2.3. Bayesian Networks

Often Markov networks are not powerful enough. In Ex. 3.3, we must add an edge between E and B, because they are conditionally dependent given A. The information that they are independent cannot be expressed by a Markov network.

Ex. 3.3 induces another structure. Burglary and an earthquake are direct causes for the alarm to go off. So we can build a *causal network*, a directed graph with edges from causes to results; in our case one edge from burglary to alarm and one from earthquake to alarm (see Fig. 3.1).

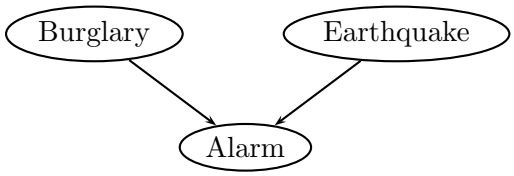

Figure 3.1.: The causal network for the alarm example

Adding to a causal network a probability distribution for every node X_i, conditioned on its direct parents $\Pi_i = \{Y \in V | (Y, X_i) \in E\}$, we get the *Bayesian network.*

To give the formal definition of a Bayesian network, we first need a few definitions for directed graphs.

Definition 3.4. Let $G = (V, E)$ be a directed graph with $V = \{X_i, i = 1, \ldots, n\}$ and $E \subseteq (V \times V)$.

1. The **parents** of a node X_i are $\Pi_i = \{Y \in V | (Y, X_i) \in E\}$.
2. A **path** $\rho(X, Y)$ between two nodes $X, Y \in V$ is a sequence of nodes $(R_i)_{i \in \{0,\ldots,m\}}$ with $R_0 = X$, $R_m = Y$, and $\forall i \in \{1, \ldots, m\} : (R_{i-1}, R_i) \in E$. $m \geq 1$ is called the length of the path. It is the number of edges to be traversed on the way from X to Y.
3. The **ancestors** of X_i are $A_i = \{Y \in V | \exists \rho(Y, X_i)\}$.
4. The **descendents** of X_i are $D_i = \{Y \in V | \exists \rho(X_i, Y)\}$. The **non-descendents** of X_i are $\overline{D}_i = V \setminus (\{X_i\} \cup D_i)$.

Definition 3.5. A **Bayesian network** for a distribution p is a directed acyclic graph $G = (V, E)$ with $V = \{X_i, i = 1, \ldots, n\}$ and

$$\forall i : X_i \perp\!\!\!\perp (\overline{D}_i \setminus \Pi_i) | \Pi_i \tag{3.18}$$

The definition is justified by the following theorem.

Theorem 3.3 (Factorization). *Let $G = (\{X_1, \ldots, X_n\}, E)$ be a Bayesian network for the distribution $p(\mathbf{x})$. Then*

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i | \pi_i) \tag{3.19}$$

Proof. Alternative, constructive proofs are given in the literature [Lau96, Mah01].

First, assume without loss of generality that the numbering of the nodes is such that there is no edge (X_i, X_j) with $i > j$. Since the graph is acyclic, such a numbering without "backward" edges is always possible.

Then, we can write

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i | x_1, \ldots, x_{i-1}) \tag{3.20}$$

This is called the *trivial factorization*, conditioning each variables on all its predecessors. It is trivially true because all fractions cancel out. From our assumed ordering, we know that $\Pi_i \subseteq \{X_1, \ldots, X_{i-1}\}$. Define $\overline{\Pi}_i := \{X_1, \ldots, X_{i-1}\} \setminus \Pi_i$. So, (3.20) can be written as

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i | \pi_i, \overline{\pi}_i) \tag{3.21}$$

But since $\overline{\Pi}_i \subseteq (\overline{D}_i \setminus \Pi_i)$ (they cannot be descendents, since they come before X_i), it follows from (3.18) and (3.4) that

$$p(x_i | \pi_i, \overline{\pi}_i) = p(x_i | \pi_i) \tag{3.22}$$

and therefore

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i | \pi_i) \tag{3.23}$$

□

Proposition 3.4. *Bayesian networks are equivalent to factorizations.*

Proof. Given a Bayesian network, the induced distribution (3.19) can be written in the form (2.29) by setting each $\mathbf{X}_{b_i} = \{X_i\}$ and $\mathbf{X}_{c_i} = \Pi_i$.

Conversely, let p be a factorized distribution

$$p(\mathbf{x}) = \prod_{i=1}^{m} p(\mathbf{x}_{b_i} | \mathbf{x}_{c_i}) \tag{3.24}$$

The difference to a Bayesian network distribution is that b_i may contain several variable indices, whereas Bayesian network distributions are a product of conditional distributions for single variables, given their parents. Therefore, the index set b_i must be split up in

$$b_i = \{b_i(\kappa) | 1 \leq \kappa \leq |b_i|\} \ , \tag{3.25}$$

imposing an arbitrary ordering of the variables.

This gives rise to the following separation:

$$p(\mathbf{x}_{b_i} | \mathbf{x}_{c_i}) = \prod_{\kappa=1}^{|b_i|} p(x_{b_i(\kappa)} | \mathbf{x}_{c_i}, x_{b_i(1)}, \ldots, x_{b_i(\kappa-1)}) \tag{3.26}$$

It is the trivial factorization of $p(\mathbf{x}_{b_i}|\mathbf{x}_{c_i})$.

Graphically, the Bayesian network contains edges from each variable in $\mathbf{X}_{c_i}$ to each variable in $\mathbf{X}_{b_i}$, and all edges among the variables in $\mathbf{X}_{b_i}$, of course without introducing cycles. The complete Bayesian network factorization is

$$p(\mathbf{x}) = \prod_{i=1}^{m} \prod_{\kappa=1}^{|b_i|} p(x_{b_i(\kappa)}|\mathbf{x}_{c_i}, x_{b_i(1)}, \ldots, x_{b_i(\kappa-1)}) \tag{3.27}$$

□

3.2.4. Inference in Graphical Models

Bayesian networks are very good for calculating and sampling from the distribution $p(\mathbf{x})$. But often we want to calculate other probabilities, or update the probabilities when the values of some variables are known. Such issues are called *inference*.

Example 3.4 (Continuing Ex. 3.3). Suppose that the neighbor uses the same alarm system. We add the events A' (neighbor's alarm going off) and B' (burglary in neighbor's house) with the same probabilities as in Ex. 3.3. The Bayesian network for these events is given in Fig. 3.2.

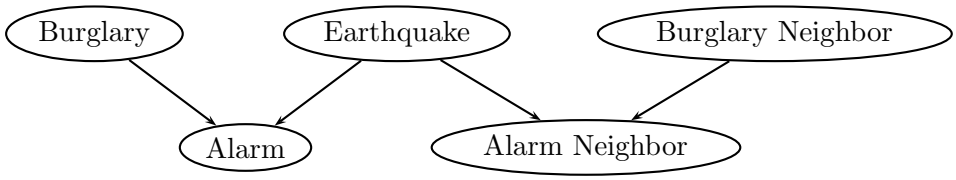

Figure 3.2.: The causal network including the neighbor

- An example for an inference problem is: Supposing that the neighbor's alarm went off, what is the probability that our alarm goes off, too?

  As can be seen from Fig. 3.2, the events are only linked by the event *earthquake*. Since we assume the neighbor's alarm to be equivalent to ours, we use the result (3.11): $p(E|A') = 0.101265$. Using this updated probability of an earthquake, given the knowledge A', we calculate (remember that $p(A|\bar{B}, \bar{E}) = 0$):

$$\begin{aligned} p(A) &= p(A|B,E)p(B)p(E) + p(A|\bar{B},E)p(\bar{B})p(E) + p(A|B,\bar{E})p(B)p(\bar{E}) && (3.28)\\ &= 0.0983790125 && (3.29) \end{aligned}$$

- If we observe that both alarms go off, the probability of an earthquake is

$$p(E|A,A') = \frac{\sum_{b,b'} p(B=b)p(B=b')p(E)p(A|b,E)p(A'|b',E)}{\sum_{b,b',e} p(B=b)p(B=b')p(E=e)p(A|b,e)p(A'|b',e)} = 0.926917 \tag{3.30}$$

So, the probability is rather high that both alarms were triggered by an earthquake, than independently by a burglar. And indeed, the probability of a burglar, given both alarms, is

$$p(B|A, A') = 0.082862 \tag{3.31}$$

This phenomenon is called *explaining away*: Given one reason for the alarm (an earthquake with high probability), the other reason (burglary) becomes improbable.

Doing inference like this is a lot of work. In formulas like (3.30) the sums run over all ancestors of the considered variables, so they can become quite large. But we can build another structure from the Bayesian network, which allows doing these calculations efficiently. This structure is the *junction tree*. There inference can be done by *message passing*, which distributes information across the whole structure.

3.3. Junction Trees

Definition 3.6. A **junction tree** (or join tree) is a tree $J = (V, E)$. The nodes are sets of variables, the **clusters** of the junction tree: $V = \{C_1, \dots, C_\gamma\}$ with $C_i \subseteq \mathbf{X}$. An edge $\{C_i, C_j\} \in E$ is identified with a **separator** (or separating set) $S = C_i \cap C_j$.

To avoid complicated notation, we make no distinction between a node of the junction tree and its variable cluster, as well as between an edge and the separator.

A junction tree fulfills the **junction property**, that for all nodes $C_i, C_j \in V$, all clusters on the unique path from C_i to C_j contain $C_i \cap C_j$.

For each cluster C, there is a local belief p_C, which is a marginal distribution defined on the variables in C. Likewise, for each separator S, there is a distribution p_S of the variables in S.

We demand the beliefs to be consistent: For a cluster C and a neighboring separator S,

$$\forall \mathbf{x}_S \sum_{\mathbf{x}_{C \setminus S}} p_C(\mathbf{x}_C) = p_S(\mathbf{x}_S) \tag{3.32}$$

Like we did in the context of factorization systems, we denote by $\mathbf{x}_C$ an assignment of the variables in C, which is a subvector of $\mathbf{x}$. $\mathbf{x}_S$ and $\mathbf{x}_{C \setminus S}$ are combined to $\mathbf{x}_C$. Further on, we will use this notation without any ado. We will also use the notation $\mathbf{x}_{\overline{C}}$ for assignments of all variables outside C.

Then we can define a distribution for the whole domain:

$$p_J(\mathbf{x}) = \frac{\prod_C p_C(\mathbf{x}_C)}{\prod_S p_S(\mathbf{x}_S)} \tag{3.33}$$

Lemma 3.5. *p_J has the following properties:*

1. *p_J is a probability distribution.*

2. *p_J is consistent with all marginal distributions on the clusters*

$$\forall \mathbf{x}_C \sum_{\mathbf{x}_{\overline{C}}} p_J(\mathbf{x}) = p_C(\mathbf{x}_C) \tag{3.34}$$

Proof. 1. That $p_J(\mathbf{x}) \geq 0$ holds, is obvious. We only have to show that it sums up to one. This is done by induction over γ, the number of clusters in the junction tree.

- For $\gamma = 1$, we have only one cluster C, and $p_J = p_C$ is trivial.
- Suppose that the claim holds for all junction trees of size $\gamma - 1$. Let J now be of size γ.

  We choose a leaf cluster C', which is connected to only one subset S'. Since J is a tree, such a cluster must exist. Then

$$\sum_{\mathbf{x}} p_J(\mathbf{x}) = \sum_{\mathbf{x}} \frac{\prod_{C \neq C'} p_C(\mathbf{x}_C)}{\prod_{S \neq S'} p_S(\mathbf{x}_S)} \frac{p_{C'}(\mathbf{x}_{C'})}{p_{S'}(\mathbf{x}_{S'})} \tag{3.35}$$

$$= \sum_{\mathbf{x}_{\overline{C' \setminus S'}}} \frac{\prod_{C \neq C'} p_C(\mathbf{x}_C)}{\prod_{S \neq S'} p_S(\mathbf{x}_S)} \sum_{\mathbf{x}_{C' \setminus S'}} \frac{p_{C'}(\mathbf{x}_{C'})}{p_{S'}(\mathbf{x}_{S'})} \tag{3.36}$$

$$= \sum_{\mathbf{x}_{\overline{C' \setminus S'}}} \frac{\prod_{C \neq C'} p_C(\mathbf{x}_C)}{\prod_{S \neq S'} p_S(\mathbf{x}_S)} \tag{3.37}$$

  because of (3.32).

  By removing a leaf C' and all variables which appear only in this leaf (the junction property asserts that $C' \setminus S'$ do not appear elsewhere), we get a junction tree with $\gamma - 1$ nodes, for which the induction claim holds.

2. The proof of consistence with a cluster belief $p_{C_i}(\mathbf{x}_{C_i})$ works similarly. Choose a leaf node $C' \neq C_i$ in the junction tree, and its only separator S'. The junction property ensures that $C_i \cap (C' \setminus S') = \emptyset$. Then

$$\sum_{\mathbf{x}_{\overline{C_i}}} p_J(\mathbf{x}) = \sum_{\mathbf{x}_{\overline{C_i}}} \frac{\prod_{C \neq C'} p_C(\mathbf{x}_C)}{\prod_{S \neq S'} p_S(\mathbf{x}_S)} \frac{p_{C'}(\mathbf{x}_{C'})}{p_{S'}(\mathbf{x}_{S'})} \tag{3.38}$$

$$= \sum_{\mathbf{x}_{\overline{C_i} \setminus \overline{C' \setminus S'}}} \frac{\prod_{C \neq C'} p_C(\mathbf{x}_C)}{\prod_{S \neq S'} p_S(\mathbf{x}_S)} \sum_{\mathbf{x}_{C' \setminus S'}} \frac{p_{C'}(\mathbf{x}_{C'})}{p_{S'}(\mathbf{x}_{S'})} \tag{3.39}$$

$$= \sum_{\mathbf{x}_{\overline{C_i} \setminus \overline{C' \setminus S'}}} \frac{\prod_{C \neq C'} p_C(\mathbf{x}_C)}{\prod_{S \neq S'} p_S(\mathbf{x}_S)} \tag{3.40}$$

   We continue to remove clusters from the junction tree like this, until we are left with C_i alone, for which the claim is trivial.

□

3.4. Building a Junction Tree from a Bayesian Network

A Bayesian network is a good device to code our knowledge about a model. However, the junction tree is better suited for inference, because it provides us with an efficient algorithm to compute and propagate beliefs (described in Sect. 3.5). This section presents a way to build a junction tree, given a Bayesian network. The procedure consists of the following steps:

1. Building the moral graph from the Bayesian network
2. Triangulating the moral graph
3. Identifying the cliques of the moral graph
4. Creating junction tree clusters of the cliques and connecting them with separators.

In the following these steps are presented in turn.

3.4.1. Building a Moral Graph

A moral graph is an undirected graphical model. It results from a Bayesian network by dropping the directions of all edges. Then, an additional step called "marrying the parents" is needed, i. e. adding an edge between all pairs of nodes that are parents of some node.

For example, in Fig. 3.2, we see that E and B are both parents of A, because the alarm can be triggered by an earthquake or by a burglary. So, when constructing the moral graph, they must be "married". The same goes for E and B'.

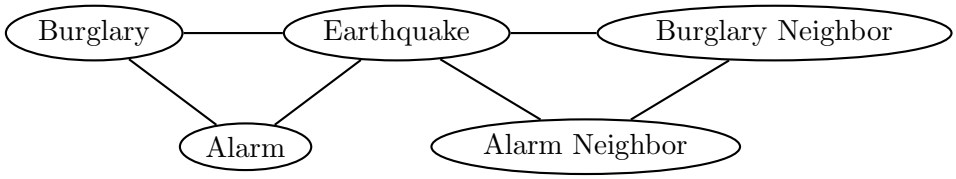

Figure 3.3.: The moral graph of the network in Fig. 3.2

The resulting model is called the *moral graph* of the Bayesian network. [Lau96]

Proposition 3.6. *The moral graph of a Bayesian network is a Markov network.*

Proof. Let $G_B = (V, E_B)$ be the Bayesian network with $V = \{X_1, \ldots, X_n\}$. Let the Markov network $G_M = (V, E_M)$ be constructed according to the algorithm with

$$\{X_i, X_j\} \in E_M \Leftrightarrow (X_i, X_j) \in E_B \vee (X_j, X_i) \in E_B \vee \exists k : (X_i, X_k) \in E_B \wedge (X_j, X_k) \in E_B \tag{3.41}$$

That means, an edge between X_i and X_j has been present in G_B, or they were married, being the parents of X_k.

To prove the Markov property, we need to show (3.14) for any $\{X_i, X_j\} \notin E_M$, i. e. we need to show their conditional independence, given the other variables $\overline{\mathbf{X}} = \mathbf{X} \setminus \{X_i, X_j\}$. We have

$$p(x_i, x_j|\overline{\mathbf{x}}) = \frac{p(\mathbf{x})}{\sum_{x'_i, x'_j} p(x'_i, x'_j, \overline{\mathbf{x}})} \tag{3.42}$$

Here we can now insert (3.19). We first notice that all factors that don't contain x_i or x_j cancel out. This leaves us with

$$p(x_i, x_j|\overline{\mathbf{x}}) = \frac{p(x_i|\Pi_i)p(x_j|\Pi_j)p(ch(i)|x_i, part(i))p(ch(j)|x_j, part(j))}{\sum_{x'_i, x'_j} p(x'_i|\Pi_i)p(x'_j|\Pi_j)p(ch(i)|x'_i, part(i))p(ch(j)|x'_j, part(j))} \tag{3.43}$$

where $ch(i)$ are the children of X_i and $part(i)$ are the partners of X_i, i. e. the nodes which have children in common with X_i.

The denominator is independent of X_i and X_j. It serves as a normalization constant Z. X_i is neither parent nor child of X_j, and there are no common children of X_i and X_j (otherwise they would have been married). So $X_i \notin ch(j)$ and $X_i \notin part(j)$, and vice versa. Therefore the nominator can be split into independent functions $f_i(x_i, \overline{\mathbf{x}}) = p(x_i|\Pi_i)p(ch(i)|x_i, part(i))$ and $f_j(x_j, \overline{\mathbf{x}})$, accordingly. So we get

$$p(x_i, x_j|\overline{\mathbf{x}}) = \frac{f_i(x_i, \overline{\mathbf{x}})f_j(x_j, \overline{\mathbf{x}})}{Z} \tag{3.44}$$

$$= p(x_i|\overline{\mathbf{x}})p(x_j|\overline{\mathbf{x}}) \tag{3.45}$$

□

3.4.2. Triangulation

The next step is to triangulate the moral graph.

Definition 3.7. A graph is **triangulated** (chordal) if it contains no chord-free cycle of length ≥ 4.

For triangulating a graph G, the following algorithm can be applied:

- Copy the graph G to a graph G'
- While there are nodes left in G':
  - Choose a node v from the graph G'
  - Add edges between all his neighbors in G' to G and G'
  - Remove v from G'

Obviously, the result depends strongly on the choice of nodes. Of course, it is desirable to add as few edges as possible. But the problem of optimal triangulation is NP-complete [ACP87]. A simple heuristics is to always choose a node that induces the smallest number of edges to be added.

A refinement of this heuristics, along with hints for efficient implementation, is described in [HD96].

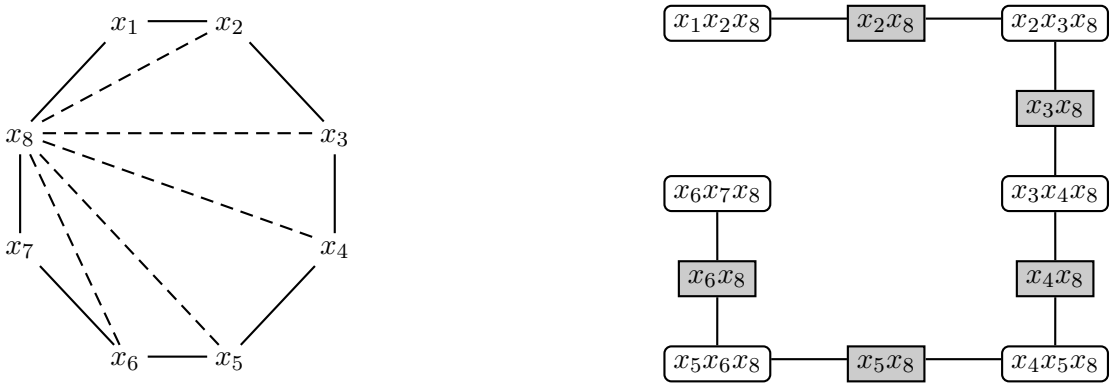

Figure 3.4.: Moral graph with triangulation (dashed edges) and junction tree for a 1-D bivariate circle. The shaded boxes are the separators.

3.4.3. Finding the Cliques

Definition 3.8. A **clique** of a graph is a maximal, fully connected subset of nodes.

Fully connected means that there is an edge between each pair of nodes in the clique. It is maximal in the sense that it is not fully contained in another clique.

In general, finding the cliques of a graph is NP-complete, too. But for triangulated graphs, there exists an efficient algorithm. Even better, it is possible to identify the cliques during the triangulation algorithm [HD96]. In each step, an induced clique is formed by the node v and all its neighbors. The idea is to save each of these cliques, unless it is a subset of a previously saved clique.

3.4.4. Building the Junction Tree Structure

The clusters of the junction tree are the cliques of the triangulated graph.

Then, the cliques must be connected by separators. Not all cliques with nonempty intersection should be connected, because that would not result in a cycle-free junction tree. A possible approach, however, is to first add all possible separators and then remove some, while keeping the junction property fulfilled, until the model is a tree. The algorithm is presented in detail in [HD96].

3.4.5. An Example of Structure Building

We now resume Ex. 2.1 (p. 24) to demonstrate how to build a junction tree from the bivariate circle. The Markov network for this structure is a circle where each variable is connected to its two neighbors, the two variables with which it shares a subfunction in (2.26). It is depicted in Fig. 3.4.

The circle can be triangulated by connecting one node with all other nodes. This triangulation results from the algorithm in Sec. 3.4.2 by choosing the nodes $1, 2, 3, \ldots$ in order, thus connecting x_8 to all other nodes.

The cliques of the triangulated graph are sets of three variables $\{x_i, x_{i+1}, x_8\}$. The resulting junction tree is shown on the right side of Fig. 3.4.

3.5. Belief Propagation in Junction Trees

3.5.1. Message Passing

Once the structure of the junction tree is built, there are only the marginal distributions left to determine. They need to meet two criteria:

- They should be consistent with each other (3.32), and
- the junction tree distribution should be equal to the Bayesian network distribution (3.19).

Consistence is reached by *passing messages* between neighboring nodes. Suppose we want to pass a message from a cluster C_i to a neighboring cluster C_j via the separator S_k (remember that $S_k = C_i \cap C_j$). This is done by the following algorithm:

- The current distribution on the separator S_k is saved in $p^{\text{old}}_{S_k}$.

$$p^{old}_{S_k} = p_{S_k} \tag{3.46}$$

- The new distribution on the separator S_k is calculated as the marginalization of the distribution on C_i for the variables in S_k.

$$p^{\text{new}}_{S_k}(\mathbf{x}_{S_k}) = \sum_{\mathbf{x}_{C_i \setminus S_k}} p_{C_i}(\mathbf{x}_{C_i}) \tag{3.47}$$

- The distribution on the node C_j is updated according to the change in S_k.

$$p^{\text{new}}_{C_j}(\mathbf{x}_{C_j}) = p^{\text{old}}_{C_j}(\mathbf{x}_{C_j}) \frac{p^{\text{new}}_{S_k}(\mathbf{x}_{S_k})}{p^{\text{old}}_{S_k}(\mathbf{x}_{S_k})} \tag{3.48}$$

  Remember that $\mathbf{x}_{S_k}$ is a subvector of $\mathbf{x}_{C_j}$.

Lemma 3.7. *After passing a message from C_i to C_j via S_k, p_{S_k} is consistent with p_{C_i}. But p_{S_k} is consistent with p_{C_j} only if they have already been consistent before.*

Proof. The first claim follows trivially from (3.47). The second claim follows from

$$\sum_{\mathbf{x}_{C_j \setminus S_k}} p^{\text{new}}_{C_j}(\mathbf{x}_{C_j}) = \sum_{\mathbf{x}_{C_j \setminus S_k}} p^{old}_{C_j}(\mathbf{x}_{C_j}) \frac{p^{\text{new}}_{S_k}(\mathbf{x}_{S_k})}{p^{\text{old}}_{S_k}(\mathbf{x}_{S_k})} \tag{3.49}$$

$$= p^{\text{new}}_{S_k}(\mathbf{x}_{S_k}) \frac{\sum_{\mathbf{x}_{C_j \setminus S_k}} p^{old}_{C_j}(\mathbf{x}_{C_j})}{p^{\text{old}}_{S_k}(\mathbf{x}_{S_k})} \tag{3.50}$$

□

Another important property of message passing is the following:

Lemma 3.8. *Passing a message does not change p_J.*

Proof. Let p_J be the distribution before, and p_J^{new} the distribution after passing a message from C_i via S_k to C_j. Then

$$p_J^{\text{new}}(\mathbf{x}) = \frac{\prod_{C \neq C_j} p_C(\mathbf{x}_C)}{\prod_{S \neq S_k} p_S(\mathbf{x}_S)} \frac{p_{C_j}^{\text{new}}(\mathbf{x}_{C_j})}{p_{S_k}^{\text{new}}(\mathbf{x}_{S_k})} \tag{3.51}$$

$$= \frac{\prod_{C \neq C_j} p_C(\mathbf{x}_C)}{\prod_{S \neq S_k} p_S(\mathbf{x}_S)} \frac{p_{C_j}(\mathbf{x}_{C_j})}{p_{S_k}^{\text{new}}(\mathbf{x}_{S_k})} \frac{p_{S_k}^{\text{new}}(\mathbf{x}_{S_k})}{p_{S_k}(\mathbf{x}_{S_k})} \tag{3.52}$$

$$= p_J(\mathbf{x}) \tag{3.53}$$

□

3.5.2. Incorporating the Factorization into the Junction Tree

The way to acquire the junction tree distribution is the following:

- Start with uniform distributions on every cluster and separator.
- For every $i \in \{1, \ldots, n\}$, there must be a cluster $C_{p(i)}$ in the junction tree with $\{X_i\} \cup \Pi_i \subseteq C_{p(i)}$. This is guaranteed by "marrying the parents". Following [HD96], we call this cluster the *parent cluster* of X_i.

  Update this cluster's distribution by multiplying with the given marginal distribution on X_i, given its parents:

$$p_{C_{p(i)}}^{\text{new}}(\mathbf{x}_{C_{p(i)}}) = p_{C_{p(i)}}^{\text{old}}(\mathbf{x}_{C_{p(i)}}) p(x_i|\pi_i) \tag{3.54}$$

Having done this, it follows from (3.33) that p_J is now equal to (3.19):

$$\frac{\prod_C p_C(\mathbf{x}_C)}{\prod_S p_S(\mathbf{x}_S)} = \prod_{i=1}^{n} p(x_i|\pi_i) \tag{3.55}$$

Each factor $p(x_i|\pi_i)$ of (3.19) has been multiplied into some p_C, whereas the denominator of (3.55) is still a product of uniform distributions. This leaves only the consistency to be achieved.

3.5.3. Global Message Distribution

Consistency of all clusters is achieved by two algorithms which pass messages throughout the junction tree: *Collect Evidence* and *Distribute Evidence*. Both work in a recursive way; *Collect Evidence* receives messages from all nodes in direction of some root node, *Distribute Evidence* sends messages out from a root node throughout the junction tree.

Collect Evidence

Collect Evidence on a cluster C consists of the following steps:

- Mark C.
- Call *Collect Evidence* on all unmarked neighbor clusters of X.
- Pass a message from C to the neighbor from which it was invoked.

Distribute Evidence

Distribute Evidence on a cluster C consists of the following steps:

- Mark C.
- Pass a message from C to all unmarked neighbor clusters of C.
- Call *Distribute Evidence* on all unmarked neighbor clusters of C.

Global Message Distribution

Global message distribution is the following algorithm:

- Choose an arbitrary root node C_{root}.
- Call *Collect Evidence* on C_{root}.
- Call *Distribute Evidence* on C_{root}.

It follows from Lemma 3.7 that after the *Collect Evidence* step each separator is consistent with its neighbor cluster on the far side from C_{root}, and that after the *Distribute Evidence* step all separators and clusters are consistent. Meanwhile, Lemma 3.8 asserts that p_J is still the factorization distribution (3.19).

With the consistent junction tree, we can now compute any marginal distribution by doing local calculations (marginalization, conditionalization) on the corresponding node. New evidence can be incorporated into a node, and by message passing its effects are sent into the junction tree. The details of these procedures can be found in [HD96]. They are beside the focus of this thesis.

Within this thesis, the junction tree method will be combined with the Maximum Entropy principle in Sect. 4.2.7. Furthermore, the junction tree will be generalized to a region graph (allowing cycles) in combination with *loopy belief propagation* in Chapter 6.

3.6. Connections Between Graphical Models

This section sheds light on the connections between the different graphical models introduced so far. Very similar theorems can be found in [Lau96], but with a different terminology and notation. Therefore we furnish independent proofs.

3.6.1. Junction Property and Running Intersection Property

Theorem 3.9. *The running intersection property (RIP, eq. (2.41)) and the junction property (JP) are equivalent.*

Proof. This theorem has two directions:

"⇒": A factorization system fulfilling the RIP can trivially be turned into a junction tree. The clusters of the junction tree are the sets $\mathbf{X}_{s_i}$. The junction property is fulfilled.

If the factorization system is separable, we obtain a forest of junction trees which are separated, too.

"⇐": A junction tree can be turned into a factorization system. The index sets are the clusters of the junction tree. The RIP is fulfilled.

Now we will prove both directions.

"⇒": For every set s_i, generate a cluster node C_{s_i}. The RIP guarantees that for every $i \geq 2$ there is a $j < i$ with $c_i \subseteq s_j$. We recall the definitions of histories, residuals and separators from section 2.4.4: c_i is the set of variables in s_i that have appeared in the previous sets. The RIP states that there is a set s_j which contains all of these variables.

We connect in the junction tree C_{s_i} and C_{s_j}. The separator is $S = X_{c_i}$.

Now we have to prove the junction property. This is done by induction over the length of paths in the junction tree. Let C_1 and C_2 be nodes of the junction tree, and the length of the path between them be l.

- For $l = 1$, C_1 is the predecessor of C_2 or vice versa, and the junction property is necessarily fulfilled.
- Now assume that the junction property holds for all paths of length smaller than l. Now let C_1 and C_2 be l edges apart. We must show that $C_1 \cap C_2$ is contained in all nodes on the unique path between C_1 and C_2. If $C_1 \cap C_2 = \emptyset$, this is trivial, so we exclude this case.

  Each node contains a set of the factorization system. Let for every node C be $s_{i(C)}$ the corresponding factorization set, with the index $i(C) \in \{1, \ldots, m\}$.
  Assume without loss of generality that $i(C_1) < i(C_2)$. Since the variables in $C_1 \cap C_2$ are already in the previously added node C_1, they must belong to the separator of C_2:

$$C_1 \cap C_2 \subseteq c_{i(C_2)} \tag{3.56}$$

  With C' being the predecessor of C_2 on the path, and $c_{i(C_2)} = C_2 \cap C'$, we deduce that $C_1 \cap C_2 \subseteq C'$. Therefore also

$$C_1 \cap C_2 \subseteq C_1 \cap C' \tag{3.57}$$

The induction claim states that the junction property holds for the path from C_1 to C' with length $l-1$, so $C_1 \cap C'$ is contained in all nodes on the path. Therefore this must also hold for its subset $C_1 \cap C_2$.

The statement about separable factorizations is obvious. If a separator is empty, we need no edge to a previous node at all.

"⇐": Let there be given a junction tree $J = (V, E)$, fulfilling the junction property. A factorization system is constructed as follows.

First, we choose arbitrarily a root node $C_{\text{root}} \in V$ of the junction tree. Then, for every node $C \neq C_{\text{root}}$ we define its parent node $\rho(C)$ as its predecessor on the path to C_{root}.

Now we can add all the clusters from the junction tree to the factorization system, starting with C_{root}, then traversing through the junction tree in arbitrary order, e. g. in a breadth first search.

The RIP claims that for each node $C \neq C_{\text{root}}$, there is a C' with $i(C') < i(C)$ and $c_{i(C)} \subseteq s_{i(C')}$. But this node is just its parent $C' = \rho(C)$: If a variable x is contained in the separator $c_{i(C')}$, this means that it must have appeared in a previously added factorization set. But the junction property states that it must also be contained in every node on the path to this other set, and the path must necessarily lead through the parent $\rho(C)$, so $x \in s_{i(\rho(C))}$. This proves the RIP.

□

3.6.2. Markov Networks and Bayesian Networks

Theorem 3.10. *If a Markov network $G_M = (V, E_M)$ is turned into a Bayesian network $G_B = (V, E_B)$ by adding a direction to the edges, then the Bayesian network adheres to the Markov properties of G_M if and only if it fulfills the running intersection property. In this case, the Markov network is triangulated.*

Proof.

"⇒": If G_B fulfills the Markov property, this means that two variables X_i, X_j which are not connected in E_M – and therefore neither in E_B – are conditionally independent, given all other variables $\overline{\mathbf{X}} = \mathbf{X} \setminus \{X_i, X_j\}$. Then follows from Lemma 2.8 that

$$p(x_i, x_j, \overline{\mathbf{x}}) = g(x_i, \overline{\mathbf{x}}) h(x_j, \overline{\mathbf{x}}) \tag{3.58}$$

for some functions g, h. In other words, in the Bayesian network factorization (3.19) x_i and x_j do not appear together in any factor.

If two variables X_i, X_j with $\{X_i, X_j\} \notin E_M$ do not appear together in a factor of (3.19), that means conversely that variables which do appear together in a factor are connected in E_M. In other words, every factor $\{X_i\} \cup \Pi_i$ is a clique in E_M.

If all variables in Π_i are connected, there must be a variable $X_j \in \Pi_i$ which receives arrows from all other variables in Π_i (since directed cycles in E_B are not allowed):

$$\exists X_j \in \Pi_i \;\; \forall X \in \Pi_i \setminus \{X_j\} : \;\; (X, X_j) \in E_B \tag{3.59}$$

This means that $\Pi_i \subseteq \{X_j\} \cup \Pi_j$, which proves the running intersection property.

"⇐": We have shown in Theorem 3.9 that a Bayesian network fulfilling the RIP can be turned into an equivalent junction tree. If two variables X_i and X_j are not connected in the Bayesian network, they do not appear together in any cluster of the junction tree. So we can split up the junction tree distribution (3.33):

$$p_J(\mathbf{x}) = \frac{\prod_C p_C(\mathbf{x}_C)}{\prod_S p_S(\mathbf{x}_S)} \tag{3.60}$$

$$= g(x_i, \overline{\mathbf{x}}) h(x_j, \overline{\mathbf{x}}) \tag{3.61}$$

where g is the product of all clusters and separators containing X_i, and h the product of all other clusters and separators. This decomposition means that $X_i \perp\!\!\!\perp X_j | \overline{\mathbf{X}}$, which proves the Markov property.

In this case the Markov network must be triangulated because a chordless cycle in the Bayesian network would violate the RIP. □

3.6.3. Summary of the Connections

We have shown that triangulated Markov networks, Bayesian networks fulfilling the RIP and junction trees are equivalent. The connection of all the graphical models is depicted in the Venn diagram of Fig. 3.5.

For the sake of completeness, polytrees are also included in the diagram. Polytrees are singly connected Bayesian networks which are treated in detail in Sect. 5.2. Polytrees which fulfill the RIP are trees.

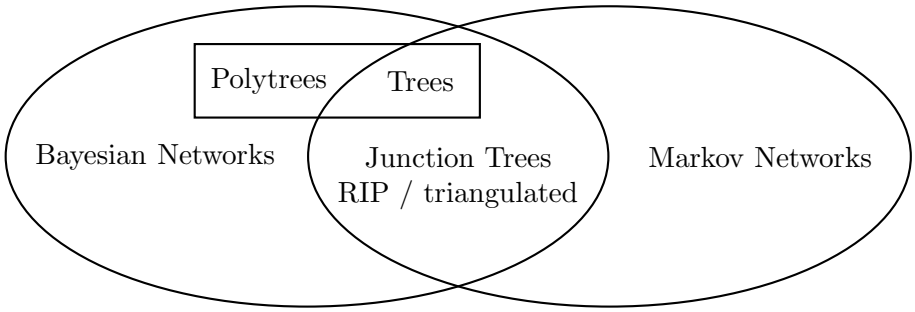

Figure 3.5.: A Venn diagram for the graphical models.

3.7. Summary

In this chapter we introduced conditional independence as a tool to describe probability distributions in an efficient manner. It is used in graphical models, the Markov network and the Bayesian network, to factorize distributions. Bayesian networks are equivalent to the factorizations presented in the previous chapter.

Graphical models are valuable tools for describing probability distributions. For probabilistic inference and for calculating Boltzmann distributions, the junction tree is a suitable structure. Message passing in a junction tree is a numerically efficient procedure to calculate distributions. The junction tree factorization is exact. It fulfills the running intersection property. If no factorization with RIP and constantly bounded clique size exists, approximate factorizations are required.

4. Information Theory and Probability Optimization Principles

This chapter introduces some notions from information theory, which are necessary for the following chapters. Then it discusses the problem of choosing a probability distribution, given a set of constraints. In our case, the constraints are marginal distributions to fulfill.

Usually, there is a large space of possible solutions. We present the principle of maximum entropy and the closely related principle of minimum relative entropy. They are justified in detail. Then, for calculating a maximum entropy distribution, the iterative proportional fitting procedure is presented, along with a method to implement it efficiently using graphical models.

A related topic is the estimation of a distribution from a population. The Bayesian method of parameter estimation is described in Sect. 4.3.

4.1. Information Theory

This section introduces some basic concepts and notions of information theory. An in-depth presentation can be found in [CT89].

The basic problem of information theory is the question of how much information about a variable is contained in a probability distribution. In his pioneering article, Shannon [Sha48] introduces the entropy as a measure of the information. The origin of his work was coding and communication: How to code a piece of information, so that it can be transferred with minimal length through a communication channel? And what is the capacity of a given communication channel? The answers of these problems later turned out to be interesting in many fields, e. g. the connection with statistical physics (thermodynamics) or computer science (Kolmogorov complexity).

4.1.1. Entropy

Definition 4.1. The **entropy** of a probability distribution p for a random variable X is given by

$$H(p) = -\sum_x p(X = x) \log p(X = x) \tag{4.1}$$

The entropy depends only on the distribution of X, not on X itself. That is to say, H depends not on the specific values that the variable X can take, but only on their probabilities. Nevertheless, it is often convenient to write $H(X)$ rather than $H(p)$.

If the logarithm is the natural logarithm, the entropy is given in *nats* (natural units). If it is the binary logarithm, the entropy is given in *bits*. But whether the binary or natural or any other logarithm is used does not matter, since they differ only by a constant factor, e. g.

$$1\,\text{nat} = \log_2 e\,\text{bits} \approx 1.442695\,\text{bits} \tag{4.2}$$

If there is an x for which there is $p(x) = 0$, then $\log p(x)$ is not defined. For this case, we set $0 \log 0 = 0$. So, adding additional states for x which cannot occur does not change the entropy.

The entropy is a measure of uncertainty about a distribution. For example, if there is only one event x with $p(x) = 1$, then the entropy is 0, because there is no uncertainty about the outcome. For a fair coin toss, the entropy is $-0.5 \log 0.5 - 0.5 \log 0.5 = 1\,\text{bit}$. This can be understood in the way that we need one bit of information to describe the result of our coin toss, namely heads or tails. Specifically the average length of an optimal description (or coding) for a random variable X is between $H(X)$ and $H(X)+1$. This can be understood as the expected number of "yes/no"-questions required to learn the outcome.

4.1.2. Joint Entropy and Conditional Entropy

It is straightforward to generalize the entropy for more than one variable:

Definition 4.2. The **joint entropy** of a probability distribution $p(x, y)$ for two variables X and Y is

$$H(X, Y) = -\sum_{x,y} p(X = x, Y = y) \log p(X = x, Y = y) \tag{4.3}$$

This is reasonable, since we could just as well regard a single vector variable instead of two scalars. Of course, more than two variables are possible in an identical manner.

Also, we can define an entropy for a conditional distribution $p(x|y)$.

Definition 4.3. The **conditional entropy** of a probability distribution for a random variable X, given another variable Y, is

$$H(X|Y) = -\sum_{x,y} p(X = x, Y = y) \log p(X = x|Y = y) \tag{4.4}$$

This extension is also straightforward: Like the entropy is the expected value of $-\log p(x)$ in the space of x, the conditional entropy is the expected value of $-\log p(x|y)$ in the space of x, y.

Another way to understand the conditional entropy is provided by the relation

$$H(X|Y) = H(X, Y) - H(Y) \tag{4.5}$$

The conditional entropy measures the reduction of uncertainty by the knowledge of Y.

4.1.3. Relative Entropy (Kullback Leibler Divergence)

Definition 4.4. The **relative entropy** or **Kullback Leibler divergence** between two probability distributions $p(x)$ and $q(x)$ is defined as

$$D(p\|q) = \sum_x p(x) \log \frac{p(x)}{q(x)} \tag{4.6}$$

The relative entropy or Kullback Leibler divergence $D(p\|q)$ is a measure of the distance between two distributions $p(x)$ and $q(x)$. If the true distribution for the variable x is $p(x)$, we can – as mentioned above – devise a coding of expected length $H(p)$. However, if we use instead a code for the distribution q, the expected length of the coding will be

$$E_{p(x)}\left(\log \frac{1}{q(x)}\right) = \sum_x p(x) \log \frac{1}{q(x)} \tag{4.7}$$

$$= \sum_x p(x) \log \frac{p(x)}{q(x)} - \sum_x p(x) \log p(x) \tag{4.8}$$

$$= H(p) + D(p\|q) \tag{4.9}$$

So, the Kullback Leibler divergence measures the inefficiency of assuming a distribution q when the real distribution is p.

We use the convention $0 \log \frac{0}{q} = 0$ and $p \log \frac{p}{0} = \infty$. The relative entropy is always non-negative and zero if and only if $p = q$. But it is not a proper distance function, because it is not symmetric and does not fulfill the triangle inequality. Nevertheless, we will use it to measure the similarity between probability distributions.

The entropy can be seen as the relative entropy to the uniform distribution $q(x) = \text{const}$.

4.1.4. Mutual Information

The mutual information is a measure of how much information about one variable is contained in another variable.

Definition 4.5. The **Mutual Information** $I(X, Y)$ is defined as

$$I(X, Y) = \sum_{x,y} p(x, y) \log \frac{p(x, y)}{p(x)p(y)} \tag{4.10}$$

So, the mutual information of two variables is the Kullback Leibler divergence to the independent product distribution. The following properties are simple to prove.

Lemma 4.1.

$$I(X, Y) = \sum_{x,y} p(x, y) \log \frac{p(x|y)}{p(x)} \tag{4.11}$$

$$= H(X) + H(Y) - H(X, Y) \tag{4.12}$$

$$= H(X) - H(X|Y) \tag{4.13}$$

$$= H(Y) - H(Y|X) \tag{4.14}$$

There is also a conditional variant.

Definition 4.6. The **Conditional Mutual Information** $I(X, Y|Z)$ is defined as

$$I(X, Y|Z) = \sum_{x,y,z} p(x, y, z) \log \frac{p(x, y|z)}{p(x|z)p(y|z)} \tag{4.15}$$

$$= \sum_{x,y,z} p(x, y, z) \log \frac{p(x|y, z)}{p(x|z)} \tag{4.16}$$

4.2. The Maximum Entropy Principle

4.2.1. Probability Concepts

Let there be a probability distribution with a number of constraints. These might be given as pieces of information about the distribution, like probabilities, marginal distributions or moments of the distributions (e. g. expectation values).

A probability distribution has got many degrees of freedom. Usually, it is not completely defined by the constraints. What can we deduce about the distribution from the constraints?

Let us motivate this problem with a few examples.

Example 4.1. Let $p(A, B, C)$ be a distribution of which we know that the univariate marginals are all $p(A = 1) = p(B = 1) = p(B = 1) = 1/2$. Which distribution would we expect? Most probably, we would come up with this:

abc	000	001	010	011	100	101	110	111
$p(a, b, c)$	$1/8$	$1/8$	$1/8$	$1/8$	$1/8$	$1/8$	$1/8$	$1/8$

(4.17)

But there are also other possible solutions, e. g. the distribution from Example 3.1 has the same marginals:

abc	000	001	010	011	100	101	110	111
$p(a, b, c)$	$1/4$	0	0	$1/4$	0	$1/4$	$1/4$	0

(4.18)

A famous example is the following:

Example 4.2 (The Brandeis Dice [Jay78]). When a die is tossed, the number of spots up can have any value i in $1 \leq i \leq 6$. Suppose a die has been tossed N times and we are told only that the average number of spots up was not 3.5 as we might expect from an "honest" die but 4.5. Given this information, *and nothing else*, what probability should we assign to i spots on the next toss?

Jaynes proposed the following answer:

$$(p_1, \ldots, p_6) = (0.05435, 0.07877, 0.11416, 0.16545, 0.23977, 0.34749) \tag{4.19}$$

What is the meaning of these numbers? Is it sensible to give such numbers? And why not other numbers? What should make us expect the distribution (4.17) rather than (4.18), given the marginals in Example 4.1?

To answer these questions, we need to understand what is meant with the word *probability*. In history, there have been two points of view about probability competing with each other:

Objectivist: The probability of an event is its relative frequency, when an experiment is performed many times.

Subjectivist: The probability of an event gives the state of knowledge of an observer.

Due to these two points of view, there are different answers to the question of example 4.2. A pure-blooded objectivist would say: "How can anybody give exact numbers like this? *Ex nihilo nihil*. We cannot obtain information from nowhere. Maybe N is a very small number, and the expectancy 4.5 is not at all accurate. And anyway, this expectancy alone is not enough information about the geometry of the dice to claim that the probability of rolling a 1 is 0.05435 and nothing else."

The subjectivist answers: "No, you got this wrong. I'm talking about probability, not frequency. I don't claim that when you roll the dice another 100.000 times, you will get 5435 times a 1. You should rather see this probability as a better's claim. Imagine a game of chance, where somebody proposes you to pay you 10 euro for each time you roll a 1, provided that you give a stake of x euro. For a normal dice, you would accept this game for a stake of up to 1.66 euro. However, if you are given the new information about the expectancy of 4.5 instead of 3.5, your state of information changes. And if you are not completely out of your mind, you should no longer accept this game for a stake larger than 0.54 euro."

The objectivist retorts: "But how can you define probability as the stake of a gambler? All gamblers are different, some are more daring, some are cautious. Definitions should not depend on human psychology."

"This is true", replies the subjectivist, "and that's why I'm proposing the concept of an 'idealized decision maker' who is perfectly rational and only computes his expected profit given the information that he has. This has nothing to do with psychology."

So, from the subjectivist point of view, it is sensible to give probabilities for the outcome of the experiment, whereas the objectivist resents this. I will go for the subjectivist point of view and compute maximum entropy probabilities, but emphasize that these probabilities cannot be seen as frequencies. It is important to remember that these probabilities are always dependent on the observer's state of knowledge. If the knowledge changes, the probabilities change, too.

See [Alm95] for a discussion of probability concepts, including ways to mix the two views of probability.

4.2.2. Definition of Maximum Entropy and Minimum Relative Entropy

Accepting the premise that we can give probabilities for a dice about which we know almost nothing, the next question to ask is: Which values should we choose, and why? The principle to follow is to maximize the entropy of the distribution [Jay57]:

Definition 4.7. The principle of **Maximum Entropy** states that, given a set of constraints, e. g. marginal distributions, moments of the distribution or single probabilities, among all possible solutions (i. e. distributions compatible with the constraints) the distribution with maximal entropy should be chosen.

We note again that this distribution serves to represent our current subjective knowledge.

A generalization of this is the following:

Definition 4.8. The principle of **Minimum Relative Entropy** states that, given an original distribution and a set of constraints, e. g. marginal distributions, moments of the distribution or single probabilities, among all possible solutions (i. e. distributions compatible with the constraints) the distribution with minimal relative entropy (Kullback Leibler divergence) should be chosen.

Maximum entropy can be seen as the special case of this, when the original distribution is the uniform distribution (see Sect. 4.1.3).

Lemma 4.2. *The maximum entropy and minimum relative entropy solution is unique.*

Proof. This follows from the concavity of the entropy and the convexity of the relative entropy. [CT89] □

There has been a lot of controversial discussion about these principles. Therefore we first present some justifications, namely Jaynes' "concentration phenomenon" and the Shore Johnson axioms.

Then we present a procedure to calculate the maximum entropy distribution. In our context, the constraints are always given in form of marginal distributions. In this case, the *iterative proportional fitting* procedure solves the problem. It is described in Sect. 4.2.5. Afterwards, an efficient implementation using graphical models is presented.

4.2.3. Why Maximum Entropy? The Concentration Phenomenon

Jaynes [Jay78] calculated the probabilities (4.19) for the Brandeis dice as the distribution with maximum entropy, given the constraint. Similarly, in Example 4.1, the distribution (4.17) has an entropy of 3 bits, whereas (4.18) has only 2 bits.

Jaynes justifies his answer with the *concentration phenomenon*: "Almost all outcomes that satisfy a given empirical constraint have frequencies extremely close to the maximum entropy probabilities."

This means that when the dice is rolled a sufficient number of times and the average number of spots is reported as 4.5, of all possible outcomes adhering to the constraint,

almost all have relative frequencies very near to the maximum entropy distribution (4.19).

We demonstrate this using Example 4.1 (p. 55). Suppose that N values are sampled from this model. Since every experiment can yield 8 different values for *abc*, we have 8^N possible outcomes. Given the constraint that variable A has relative frequency 0.5, the number of possible permutations for the values of A is $\binom{N}{0.5N}$. The same holds for the other two variables, so that all in all $\binom{N}{0.5N}^3$ outcomes adhere to the constraints that the three variables have relative frequency 0.5.

The number of outcomes that have relative frequencies (4.17) is

$$W = \frac{N!}{(\frac{N}{8})!^8} \tag{4.20}$$

The factorials can be approximated by Sterling's formula. For our purposes the simplest form suffices:

$$x! = O\left(e^{x \log x}\right) \tag{4.21}$$

The constants do not matter here. We will denote them by α.

With this approximation we can further calculate:

$$W \approx \alpha \frac{e^{N \log N}}{e^{8 \frac{N}{8} \log \frac{N}{8}}} \tag{4.22}$$

$$= \alpha e^{N \log N - N \log \frac{N}{8}} \tag{4.23}$$

$$= \alpha e^{N \log 8} \tag{4.24}$$

The same can be done for (4.18). For these relative frequencies, the number of possible outcomes is

$$W' = \frac{N!}{(\frac{N}{4})!^4} \tag{4.25}$$

The same approximation and an analogous calculation leads to

$$W' \approx \alpha e^{N \log 4} \tag{4.26}$$

Note that $\log 8$ and $\log 4$ are the entropies of the distributions (4.17) and (4.18), respectively. (We use binary logarithms here.)

The quotient of these two values is

$$\frac{W}{W'} \approx \frac{e^{N \log 8}}{e^{N \log 4}} = 2^N \quad , \tag{4.27}$$

so it is exponential in the number of experiments!

This means that the maximum entropy distribution is overwhelmingly the most likely one, provided that the constraints are the same as those assumed in the calculation. If, in an experiment, we find that the frequencies differ significantly from the maximum entropy frequencies, this indicates additional constraints not yet accounted for, so the model has to be adapted. In our example, this means that if we draw values from Example 4.1 and actually measure the relative frequencies (4.18), this means that the model is not completely described by the constraints (that the three univariate probabilities are 0.5).

4.2.4. Shore Johnson axioms

Another strong argument for maximum entropy was put forth in [SJ80]. Therein, Shore and Johnson formulated a number of basic axioms for the desired solution of our problem:

1. (Uniqueness) The result should be unique.

2. (Invariance) Choosing a different coordinate system should not affect the result.

3. (System Independence) Given independent information about independent systems, it should not matter whether we apply the information separately to each system and then combine the resulting densities or whether we apply the information to the joint density.

4. (Subset Independence) It should not matter whether one treats disjoint subsets of system states in terms of separate conditional densities or in terms of the full density.

   That means, supposing we are given information about a subset of possible states of a system, it should not matter whether we apply the information to the density conditioned on this subset or on the full density. Especially, the density conditioned on a different, disjoint subset should not be affected.

5. (No redundancy) In the absence of new information, we should not change the prior density.

They prove that maximum entropy is the only principle that fulfills these axioms. Equivalent axiom systems are also possible [PV97, Ker98].

However, it was argued in [Uff95] that the proof is flawed: The version of the system independence axiom that was used in the proof stated that it should also be valid for *dependent* systems. With this stronger requirement, maximum entropy (or minimum relative entropy) is the only solution; with the original version of the axiom, maximization of one of the *Rényi Entropies*

$$U_r(q,p) = \left(\int \left(\frac{q(x)}{p(x)} \right)^r q(x) dx \right)^{-1/r} \tag{4.28}$$

with the parameter $r > -1$ solves the problem. The relative entropy $D(q\|p)$ is the special case for $r \to 0$ (this is shown in [Uff95] using a Taylor expansion in r). Interesting is the case $r = -0.5$, which is symmetrical in p and q and thus defines a distance function between probability distributions, contrary to the relative entropy.

4.2.5. Iterative Proportional Fitting

Suppose that there is an unknown distribution $q(\mathbf{x})$, and some marginal distributions $p_k(\mathbf{x}_{s_k})$, $k = 1, \ldots, K$, are given, where each $s_k \subseteq \{1, \ldots, n\}$ is the index set of the

subvector of $\mathbf{x}$ on which the distribution p_k is defined. We assume the p_k to be *consistent*, in the sense that there exists a distribution q which satisfies

$$\forall k: \quad q(\mathbf{x}_{s_k}) = p_k(\mathbf{x}_{s_k}) \tag{4.29}$$

The objective is to find such a distribution q; as was stated above, among all possible solutions q, the most plausible one is the one with maximal entropy. Iterative Proportional Fitting (IPF) is a procedure which calculates the maximum entropy distribution.

It was introduced in [DS40] for the special case of a contingency table with some row and column sums. Which frequencies should be chosen in the table? This is a simple, two-dimensional problem.

IPF computes iteratively a distribution $q_\tau(\mathbf{x})$ from the given marginals $p_k(\mathbf{x}_{s_k})$, where $\tau = 0, 1, 2, \ldots$ is the iteration index. Most commonly, $q_{\tau=0}$ is the uniform distribution. The update formula is

$$q_{\tau+1}(\mathbf{x}) = q_\tau(\mathbf{x}) \frac{p_k(\mathbf{x}_{s_k})}{q_\tau(\mathbf{x}_{s_k})} \tag{4.30}$$

with $k = ((\tau - 1) \mod K) + 1$.

The distribution q, which has to be stored and updated in every time step, has exponential size. Also, recall that the denominator of (4.30) is defined as

$$q_\tau(\mathbf{x}_{s_k}) = \sum_{\mathbf{x}_{\overline{s_k}}} q_\tau(\mathbf{x}_{s_k}, \mathbf{x}_{\overline{s_k}}) \tag{4.31}$$

with $\overline{s_k} = \{1, \ldots, n\} \setminus s_k$. This sum is exponential, too. Therefore the naïve implementation takes exponential time and space. If the marginals are inconsistent, it will not converge.

It took a long time to achieve the link between IPF and maximum entropy. The entropy was introduced for computer science in [Sha48] and [Jay57]. Minimum relative entropy was used in [Lew59], and the connection to IPF was presented in [IK68]. The proof that IPF converges against the maximum entropy solution was first tried in [Kul68], but was faulty. The correct proof in a most general measure-theoretic framework was given in [Csi75].

Other disciplines have developed similar algorithms for calculating maximum entropy distributions. If the constraints are given as expected values of feature functions about the distribution, the *Generalized Iterative Scaling* procedure [DR72, Rat97] calculates the maximum entropy distribution. For constraints in form of probabilistic rules, Meyer [Mey98] has presented an algorithm. All these algorithms are equivalent. It is possible to transform the constraints; for example in [MH02b] marginal distributions are transformed into probabilistic rules.

4.2.6. IPF and Maximum Entropy

We now give a proof that IPF converges to the maximum entropy distribution. The proof follows the same ideas as similar proofs in [Rat97, Mey98].

The proof requires the definition of two sets of probability distributions.

Definition 4.9. We define the following sets of probability distributions:

$$\mathcal{P} := \{p(\mathbf{x}) | \forall k, \mathbf{x}_{s_k} : p(\mathbf{x}_{s_k}) = p_k(\mathbf{x}_{s_k})\} \tag{4.32}$$

$$\mathcal{Q} := \{p(\mathbf{x}) | \exists Z, \lambda_k : p(\mathbf{x}) = \frac{1}{Z} \prod_k e^{\lambda_k(\mathbf{x}_{s_k})}\} \tag{4.33}$$

$\mathcal{P}$ is the set of distributions which is consistent with the given marginals. $\mathcal{Q}$ is the set of distributions which comply with the given parametric form of a Boltzmann distribution which is decomposable along the structure of the given marginals. First we show that the maximum entropy distribution as well as the IPF distributions q_τ are of this parametric form.

Lemma 4.3. *The maximum entropy distribution consistent with the p_k is in $\mathcal{Q}$.*

Proof. Maximization under the consistency constraints is done using Lagrange multipliers. For each constraint

$$q(\mathbf{x}_{s_k}) = p_k(\mathbf{x}_{s_k}) \tag{4.34}$$

a new variable $\lambda_k(\mathbf{x}_{s_k})$ is added, and the function to be maximized is the Lagrangian, the entropy plus one term for each constraint:

$$L = \sum_{\mathbf{x}} q(\mathbf{x}) \log q(\mathbf{x}) + \sum_k \sum_{\mathbf{x}_{s_k}} \lambda_k(\mathbf{x}_{s_k}) \left(p_k(\mathbf{x}_{s_k}) - q(\mathbf{x}_{s_k})\right) \tag{4.35}$$

Setting the derivatives with respect to the $\lambda_k(\mathbf{x}_{s_k})$

$$\frac{\partial L}{\partial \lambda_k(\mathbf{x}_{s_k})} = p_k(\mathbf{x}_{s_k}) - q(\mathbf{x}_{s_k}) \tag{4.36}$$

equal to zero ensures that the constraints (4.34) are fulfilled.

The derivatives with respect to the $q(\mathbf{x})$ are

$$\frac{\partial L}{\partial q(\mathbf{x})} = \log q(\mathbf{x}) + 1 - \sum_k \lambda_k(\mathbf{x}_{s_k}) \ . \tag{4.37}$$

Setting these equal to zero and solving for $q(\mathbf{x})$ yields

$$q(\mathbf{x}) = \exp\left(\sum_k \lambda_k(\mathbf{x}_{s_k}) - 1\right) \tag{4.38}$$

which is equivalent to (4.33) with appropriate Z. □

Lemma 4.4. *All distributions q_τ – and therefore also the limit distribution q_∞, provided that the iteration converges – are in $\mathcal{Q}$.*

Proof. We prove the lemma by induction over τ.

- For $\tau = 0$, we have q_0 initialized uniformly. So with all $\lambda_k = 0$, the claim is fulfilled.

- Assuming that $q_\tau \in \mathcal{Q}$, we now consider (4.30). The fraction on the right side of (4.30) depends only on the variables in $\mathbf{x}_{s_k}$. Therefore we calculate

$$q_{\tau+1}(\mathbf{x}) = q_\tau(\mathbf{x}) \frac{p_k(\mathbf{x}_{s_k})}{q_\tau(\mathbf{x}_{s_k})} \tag{4.39}$$

$$= \frac{1}{Z} \left(\prod_{k' \neq k} e^{\lambda_{k'}^\tau(\mathbf{x}_{s_{k'}})} \right) e^{\lambda_k^\tau(\mathbf{x}_{s_k})} \frac{p_k(\mathbf{x}_{s_k})}{q_\tau(\mathbf{x}_{s_k})} \tag{4.40}$$

  Updating $\lambda_k(\mathbf{x}_{s_k})$ as

$$\lambda_k^{\tau+1}(\mathbf{x}_{s_k}) = \lambda_k^\tau(\mathbf{x}_{s_k}) + \log \frac{p_k(\mathbf{x}_{s_k})}{q_\tau(\mathbf{x}_{s_k})} \tag{4.41}$$

  proves the induction step and thus also the lemma.

□

For proving the IPF theorem we need the following lemma, which provides a "pythagorean" property of the distributions q_τ.

Lemma 4.5. *Let q_τ, $q_{\tau+1}$ be distributions of consecutive steps of IPF, and $p \in \mathcal{P}$. Then*

$$D(p\|q_{\tau+1}) + D(q_{\tau+1}\|q_\tau) = D(p\|q_\tau) \tag{4.42}$$

Proof. Several times we apply (4.30) for $q_{\tau+1}(\mathbf{x})$.

$$D(p\|q_{\tau+1}) + D(q_{\tau+1}\|q_\tau) \tag{4.43}$$

$$= -H(p) - \sum_{\mathbf{x}} p(\mathbf{x}) \log q_{\tau+1}(\mathbf{x}) + \sum_{\mathbf{x}} q_{\tau+1}(\mathbf{x}) \log \frac{q_{\tau+1}(\mathbf{x})}{q_\tau(\mathbf{x})} \tag{4.44}$$

$$= -H(p) - \sum_{\mathbf{x}} p(\mathbf{x}) \log q_\tau(\mathbf{x}) + \sum_{\mathbf{x}} \left(-p(\mathbf{x}) \log \frac{p_k(\mathbf{x}_{s_k})}{q_\tau(\mathbf{x}_{s_k})} + q_{\tau+1}(\mathbf{x}) \log \frac{q_{\tau+1}(\mathbf{x})}{q_\tau(\mathbf{x})} \right) \tag{4.45}$$

$$= D(p\|q_\tau) + \sum_{\mathbf{x}} \left(-p(\mathbf{x}) \log \frac{p_k(\mathbf{x}_{s_k})}{q_\tau(\mathbf{x}_{s_k})} + q_{\tau+1}(\mathbf{x}) \log \frac{p_k(\mathbf{x}_{s_k})}{q_\tau(\mathbf{x}_{s_k})} \right) \tag{4.46}$$

$$= D(p\|q_\tau) + \sum_{\mathbf{x}} \left(-p(\mathbf{x}) + q_\tau(\mathbf{x}) \frac{p_k(\mathbf{x}_{s_k})}{q_\tau(\mathbf{x}_{s_k})} \right) \log \frac{p_k(\mathbf{x}_{s_k})}{q_\tau(\mathbf{x}_{s_k})} \tag{4.47}$$

$$= D(p\|q_\tau) + \sum_{\mathbf{x}_{s_k}} \left(-p(\mathbf{x}_{s_k}) + q_\tau(\mathbf{x}_{s_k}) \frac{p_k(\mathbf{x}_{s_k})}{q_\tau(\mathbf{x}_{s_k})} \right) \log \frac{p_k(\mathbf{x}_{s_k})}{q_\tau(\mathbf{x}_{s_k})} \tag{4.48}$$

$$= D(p\|q_\tau) \tag{4.49}$$

because of (4.32). □

Theorem 4.6. *IPF converges to the maximum entropy distribution consistent with the p_k.*

Proof. First we show that IPF converges. Let $q^* \in \mathcal{P} \cap \mathcal{Q}$. A simple induction using Lemma 4.5 gives

$$D(q^* \| q_0) = D(q^* \| q_n) + \sum_{\tau=1}^{n} D(q_\tau \| q_{\tau-1}) \ . \tag{4.50}$$

So the series

$$\sum_{\tau=1}^{\infty} D(q_\tau \| q_{\tau-1}) \tag{4.51}$$

consists of non-negative summands and is bounded above by $D(q^* \| q_0)$. This means that the summands converge to zero. But if the relative entropies between the q_τ and $q_{\tau-1}$ converge to zero, the distances between the subsequent iterations of IPF disappear, too.

(If the marginals p_k are contradictory, IPF does not converge. In this case, $\mathcal{P} = \emptyset$, and q^* does not exist.)

Now we show that the maximum entropy distribution is the unique distribution q^* in $\mathcal{P} \cap \mathcal{Q}$ and therefore coincides with the stationary point of IPF q_∞.

We show that the entropy of $q^* \in \mathcal{P} \cap \mathcal{Q}$ is higher than the entropy of any other distribution $q \in \mathcal{P}$. We begin by calculating the relative entropy between these distributions:

$$\begin{aligned} D(q \| q^*) &= \sum_{\mathbf{x}} q(\mathbf{x}) \log q(\mathbf{x}) - \sum_{\mathbf{x}} q(\mathbf{x}) \log q^*(\mathbf{x}) && (4.52) \\ &= -H(q) - \sum_{\mathbf{x}} q(\mathbf{x}) \log \frac{1}{Z^*} \prod_k e^{\lambda_k^*(\mathbf{x}_{s_k})} && (4.53) \\ &= -H(q) + \log Z^* - \sum_{\mathbf{x}} q(\mathbf{x}) \sum_k \lambda_k^*(\mathbf{x}_{s_k}) && (4.54) \\ &= -H(q) + \log Z^* - \sum_k \sum_{\mathbf{x}_{s_k}} \lambda_k^*(\mathbf{x}_{s_k}) q(\mathbf{x}_{s_k}) && (4.55) \end{aligned}$$

Since both q and q^* are consistent with the p_k, so both fulfill (4.34), we have

$$q(\mathbf{x}_{s_k}) = p_k(\mathbf{x}_{s_k}) = q^*(\mathbf{x}_{s_k}) \tag{4.56}$$

and therefore

$$\begin{aligned} D(q \| q^*) &= -H(q) + \log Z^* - \sum_k \sum_{\mathbf{x}_{s_k}} \lambda_k^*(\mathbf{x}_{s_k}) q^*(\mathbf{x}_{s_k}) && (4.57) \\ &= -H(q) + \sum_{\mathbf{x}} q^*(\mathbf{x}) \log Z^* - \sum_{\mathbf{x}} q^*(\mathbf{x}) \sum_k \lambda_k^*(\mathbf{x}_{s_k}) && (4.58) \\ &= -H(q) - \sum_{\mathbf{x}} q^*(\mathbf{x}) \log \left(\frac{1}{Z^*} \prod_k e^{\lambda_k^*(\mathbf{x}_{s_k})} \right) && (4.59) \\ &= -H(q) - \sum_{\mathbf{x}} q^*(\mathbf{x}) \log q^*(\mathbf{x}) && (4.60) \\ &= -H(q) + H(q^*) && (4.61) \end{aligned}$$

So, since $H(q^*) = H(q) + D(q\|q^*)$, and since we know from Sect. 4.1.3 that the relative entropy is always non-negative, this proves that q^* is the maximum entropy solution among all distributions consistent with the p_k.

The maximum entropy solution is unique because if another distribution q has the same entropy, $H(q) = H(q^*)$, from (4.61) follows that $D(q\|q^*) = 0$, which is only the case if $q = q^*$.

Since a stationary point of IPF is also consistent with the p_k and in $\mathcal{Q}$, it coincides with q^* and is therefore the unique maximum entropy solution. □

4.2.7. IPF on Junction Trees

IPF can be implemented efficiently if the probability distribution is feasibly decomposable using a graphical model. In [JP95], junction trees are used for an efficient implementation of IPF.

The algorithm is identical to IPF, except that q_τ is replaced by the junction tree distribution p_J (3.33). The junction tree can be built from the Markov network given by the s_k using the algorithm described in Sect. 3.4. In every step, for learning the marginal distribution p_k, the node of the junction tree which contains $\mathbf{X}_{s_k}$ is updated using (4.30) (the construction ensures that such a node exists). Then, a global message distribution (see Sect. 3.5.3) is performed from this node throughout the junction tree. The resulting distribution is exactly equal to $q_{\tau+1}$, but the algorithm is polynomial.

An improvement of this technique was presented in [Mey98]. There, instead of a complete global message passing at every step, a more sophisticated message passing scheme is used:

1. Construct a closed tour through the junction tree that visits each node at least once. Link every given marginal p_k with at least one junction tree node with $C_i \supseteq \mathbf{X}_{s_k}$.
2. Now while there is no convergence:
   a) Carry out local IPF on the current node C_i, using all marginals which fulfill $\mathbf{X}_{s_k} \subseteq C_i$ linked to it.
   b) Send a message to the next node on the tour, then proceed to the next node.

In addition to this scheme, he proposes a different local iteration instead of IPF, too. It works with a set of probabilistic logic rules instead of the marginals. In this way he links graphical models and maximum entropy with probabilistic logic. His algorithm is more similar to Generalized Iterative Scaling [DR72] than to IPF. Additionally, this scheme provides a way to recognize inconsistency of the input which leads to divergence. Details of this can be found in [Mey98].

4.2.8. Graphical Models and Maximum Entropy

The methods described in Sect. 4.2.7 are justified by the following theorem.

Theorem 4.7. *Let $J = (V, E)$ be a junction tree. For each node $C \in V$ a probability distribution $p_C(\mathbf{x}_C)$ is given. The distributions on the node are consistent with each other.*

Then the junction tree probability distribution (3.33)

$$p_J(\mathbf{x}) = \frac{\prod_C p_C(\mathbf{x}_C)}{\prod_S p_S(\mathbf{x}_S)} \tag{4.62}$$

is the maximum entropy distribution for $\mathbf{x}$*, given the marginal distributions on the nodes.*

Proof. Let $q(\mathbf{x})$ be a probability distribution which is consistent with all $p_C(\mathbf{x}_C)$, too. This means, $q(\mathbf{x}_C) = p_J(\mathbf{x}_C)$. Then we can calculate the entropy of q:

$$\begin{aligned}
H(q) &= -\sum_{\mathbf{x}} q(\mathbf{x}) \log q(\mathbf{x}) \\
&= -\sum_{\mathbf{x}} q(\mathbf{x}) \log \frac{q(\mathbf{x})}{p_J(\mathbf{x})} - \sum_{\mathbf{x}} q(\mathbf{x}) \log p_J(\mathbf{x}) \\
&= -D(q\|p_J) - \sum_{\mathbf{x}} q(\mathbf{x}) \left(\sum_C \log p_C(\mathbf{x}_C) - \sum_S \log p_S(\mathbf{x}_S) \right) \\
&= -D(q\|p_J) - \sum_C \sum_{\mathbf{x}_C} q(\mathbf{x}_C) \log p_J(\mathbf{x}_C) + \sum_S \sum_{\mathbf{x}_S} q(\mathbf{x}_S) \log p_J(\mathbf{x}_S) \\
&= -D(q\|p_J) - \sum_C \sum_{\mathbf{x}_C} p_J(\mathbf{x}_C) \log p_J(\mathbf{x}_C) + \sum_S \sum_{\mathbf{x}_S} p_J(\mathbf{x}_S) \log p_J(\mathbf{x}_S) \\
&= H(p_J) - D(q\|p_J) \\
&\leq H(p_J)
\end{aligned}$$

with equality only if $q = p_J$ (see section 4.1.3 about the Kullback Leibler divergence: $D(q\|p_J)$ is non-negative and zero iff $q = p_J$). □

Corollary 4.8. *A factorization fulfilling the RIP is the maximum entropy distribution, adhering to the given marginal distributions.*

Proof. Combine Theorems 3.9 and 4.7. □

In [WPS+04], special cases of this result were proven from a genetic algorithms perspective. There, marginal probability distributions are called "schema family". Since they do not use the powerful tools of probability calculus, their proofs require hard work.

4.3. Bayesian Probabilities and Prior Distributions

Up to now the constraints have been taken for granted. It was assumed that the measurements are exact and the values can be used as given data.

However, especially for small amounts of data this assumption is inaccurate. This section describes the Bayesian method which is favorable in this case.

4.3.1. Bayesian Parameter Estimation

We present the problem of Bayesian parameter estimation with a simple example. A good introduction to the subject can be found in [Hec99].

Consider spinning a coin. The result can be `H` (heads) or `T` (tails). We want to estimate the parameter of the coin, $\theta = p(\texttt{H})$.

Suppose we have spun the coin N times, of which N_H had the result `H`. The first estimate for θ is then

$$\theta = \frac{N_H}{N} \tag{4.63}$$

This goes along the objectivist way: The probability is the number of positive outcomes divided by the total number of trials. It is the *maximum likelihood* estimate of θ: Given this parameter, the outcome (N_H of N) is the most probable.

Consider the case $N = 3$, $N_H = 3$. Certainly no subjectivist would claim that "tails" is impossible (we assume that there is a tails side on the coin), on the grounds that in three trials the result was heads. He would point to the law of large numbers and ask for a larger sample size.

A more extreme case is described in the theater play *Rosencrantz & Guildenstern Are Dead* [Sto67]. Here coins are spun $N = 79$ times and come up heads $N_H = 79$ times. The characters are impressed by this phenomenon (Guildenstern: "A weaker man might be moved to re-examine his faith, if in nothing else at least in the law of probability."), but they go on spinning coins. They are more inclined to bet that the next coin will come up heads, too, but they still do not believe that tails is impossible. This shows that Rosencrantz and Guildenstern follow the subjectivist view of probability.

In the subjectivist or Bayesian framework, even before the coin was spun at all, a probability of heads can be given. It uses the *a priori* knowledge ξ to give an *a priori distribution* for the parameter, $p(\theta|\xi)$. (Objectivists would never give a distribution for θ, they would say that θ is a physical constant given by the geometry of the coin.)

Now, a set of data D is generated, which leads to the *a posteriori* distribution $p(\theta|D,\xi)$. This is calculated by Bayes' law (hence the name *Bayesian statistics*):

$$p(\theta|D,\xi) = \frac{p(D|\theta,\xi)p(\theta|\xi)}{p(D|\xi)} \tag{4.64}$$

$$= \frac{p(D|\theta,\xi)p(\theta|\xi)}{\int p(D|\tilde{\theta},\xi)p(\tilde{\theta}|\xi)\,d\tilde{\theta}} \tag{4.65}$$

The denominator is independent of θ and serves only for normalization. $p(D|\theta,\xi)$ is the probability to observe the data D, given the parameter θ and the background knowledge ξ. For coin tossing, it is binomially distributed. If $D \in \{\texttt{H},\texttt{T}\}^N$ is a vector of N results, in which heads occurs N_H times and tails N_T times, the formula reads

$$p(D|\theta,\xi) = \theta^{N_H}(1-\theta)^{N_T} \tag{4.66}$$

Recall that the question is not: "What is the probability of the probability of heads?", but "What is the probability of heads, given the data D?" For answering this question,

the expected value is used:

$$p(\mathtt{H}|D,\xi) = \int p(\mathtt{H},\theta|D,\xi)\,d\theta \tag{4.67}$$

$$= \int p(\mathtt{H}|\theta,D,\xi)p(\theta|D,\xi)\,d\theta \tag{4.68}$$

$$= \int \theta \cdot p(\theta|D,\xi)\,d\theta \tag{4.69}$$

If this integral is too complicated to be solved exactly, it can be approximated with the maximal value of θ:

$$p(\mathtt{H}|D,\xi) \approx \operatorname*{argmax}_{\theta} p(\theta|D,\xi) \tag{4.70}$$

This is justifiable when the distribution is sharply peaked around this maximum.

4.3.2. Bayesian Priors

Now the only open question is which *a priori* distribution to use. This is not trivial, and many distributions are possible.

For the problem of coin tossing, the *Beta distribution* is convenient. Its density is defined as

$$p(\theta|\xi) = \mathrm{Beta}(\theta|\eta_H,\eta_T) = \frac{\Gamma(\eta_H+\eta_T)}{\Gamma(\eta_H)\Gamma(\eta_T)}\theta^{\eta_H-1}(1-\theta)^{\eta_T-1} \tag{4.71}$$

Here the background knowledge consists of the two parameters $\eta_H, \eta_T > 0$.

We recall that

$$\Gamma(z) = \int_0^\infty t^{z-1}e^{-t}dt \tag{4.72}$$

and for integer numbers $\Gamma(n) = (n-1)!$. So if η_H and η_T are integers, the Beta distribution is

$$\mathrm{Beta}(\theta|\eta_H,\eta_T) = \binom{\eta_H+\eta_T-2}{\eta_H-1}\theta^{\eta_H-1}(1-\theta)^{\eta_T-1} \tag{4.73}$$

But the parameters do not have to be integers.

The Beta distribution is convenient because then the *a posteriori* distribution is Beta-distributed, too. Furthermore, it can be used as an *a priori* distribution for more experiments. The *a priori* distribution can be updated after each experiment, or once at the end, or whenever one likes; the result is independent of this. Also, the expected value of the Beta distribution is simply

$$\int \theta \cdot \mathrm{Beta}(\theta|\eta_H,\eta_T)\,d\theta = \frac{\eta_H}{\eta_H+\eta_T} \tag{4.74}$$

The *a posteriori* distribution of θ after observing the data D with the frequencies N_H and N_T of heads and tails is

$$p(\theta|D,\xi) = \mathrm{Beta}(\theta|\eta_H+N_H,\eta_T+N_T) \tag{4.75}$$

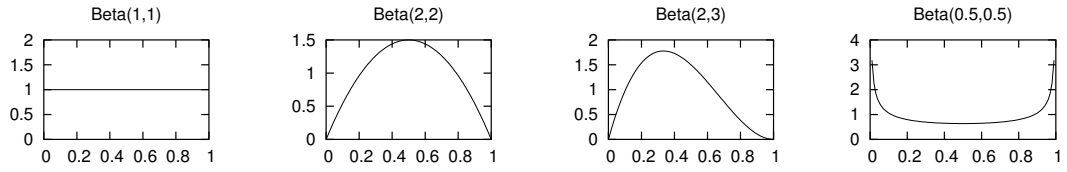

Figure 4.1.: Some example Beta densities

Putting all these formulas together, we get the probability of observing heads in the next experiment as

$$p(\mathtt{H}|D,\xi) = \frac{N_H + \eta_H}{N + \eta_H + \eta_T} \tag{4.76}$$

The classical maximum likelihood estimation is the limit case for $\eta_H, \eta_T \to 0$. Several Beta densities are depicted in Fig. 4.1. If one believes that the coin is biased to favor heads, one can set $\eta_H > \eta_T$, and vice versa. If they are equal, none of the two outcomes is favored, and the distribution is symmetric. But the values of η_H and η_T code the belief of whether the coin is biased or not. For $\eta_H = \eta_T = 1$, the *a priori* probability of the parameter θ is uniform. If they are smaller than 1, the coin is more believed to be biased, and if they are larger than 1, the coin is more believed to be fair.

In the context of *EDA*, Bayesian priors are used to estimate the probabilities in the probabilistic models from the population data. Since there is no reason to favor the value 0 or 1 of the bits, we will from now on assume $\eta_H = \eta_T =: \eta$.

4.3.3. EDA, Bayesian Priors and Mutation

In [Mah01], the following connection between Bayesian priors and mutation (see Sect. 2.2.3) is presented.

Proposition 4.9. *For UMDA with binary variables, estimating probabilities with the Bayesian prior η is equivalent to mutation with mutation rate $\mu = \eta/(N + 2\eta)$.*

Proof. We recall *UMDA*: The population size is N. For a bit, let N_1 be the number of individuals in the population where this bit is 1. Classically, the probability of this bit to be 1 would be N_1/N. With the Bayesian prior, however, this is changed to

$$p(X = 1) = \frac{N_1 + \eta}{N + 2\eta} \ . \tag{4.77}$$

Mutation, on the other hand, uses the classical formula and then flips the bit with probability μ. This gives

$$p(X = 1) = (1 - \mu)\frac{N_1}{N} + \mu\frac{N - N_1}{N} \tag{4.78}$$

$$= \mu\frac{N - 2N_1}{N} + \frac{N_1}{N} \tag{4.79}$$

These two formulas are now set equal and then solved for μ:

$$\mu = \left(\frac{N_1 + \eta}{N + 2\eta} - \frac{N_1}{N}\right) \frac{N}{N - 2N_1} \tag{4.80}$$

$$= \frac{\eta N - 2\eta N_1}{N(N + 2\eta)} \cdot \frac{N}{N - 2N_1} \tag{4.81}$$

$$= \frac{\eta}{N + 2\eta} \tag{4.82}$$

□

It is interesting to note that N_1 drops out in this derivation. We see that Bayesian priors and mutation indeed serve the same purpose.

From Sect. 2.2.3 we recall the rule of thumb that a mutation rate of $\mu = 1/n$ (where n is the dimension of the problem) is a good choice. This is equivalent to the prior $\eta = N/(n-2)$. In general, $\eta \approx N/n$ is a good rule of thumb.

For the more general case of *FDA*, a Bayesian prior changes the probability from classical

$$p(\mathbf{x}_{b_i}|\mathbf{x}_{c_i}) = \frac{N(\mathbf{x}_{s_i})}{N(\mathbf{x}_{c_i})} \tag{4.83}$$

to

$$p(\mathbf{x}_{b_i}|\mathbf{x}_{c_i}) = \frac{N(\mathbf{x}_{s_i}) + \eta}{N(\mathbf{x}_{c_i}) + 2^{|b_i|}\eta} \quad . \tag{4.84}$$

As mentioned in Sect. 2.4.6, if $N(\mathbf{x}_{c_i}) = 0$, this is a uniform distribution.

The prior itself is

$$\eta = \frac{\hat{N}}{2^{|s_i|-1}n} \quad . \tag{4.85}$$

4.3.4. Combination of Bayesian Parameter Estimation and Maximum Entropy

The maximum entropy method and the Bayesian parameter estimation method are similar especially in their views of probabilities. Both of them use the subjectivist standpoint.

The difference lies in the interpretation of the given data. In Bayesian parameter estimation, the given data are *outcomes of experiments*, and the subjective probabilities are updated according to the observed data. In maximum entropy, the given values (like "the expected number of dots is 4.5" or "the probability of A to be 1 is 0.5") are *model assumptions*. They can be formulated using empirical data, but just as well using theoretic or systematic derivations. If empirical data is used, it might be favorable to use the Bayesian method to estimate their values.

4.4. Summary

This chapter introduced basic notions from information theory, especially various flavors of entropy and mutual information. Then, different interpretations of probability were

presented and discussed. The subjectivist approach to probability justifies the maximum entropy principle and the minimum relative entropy principle. These were presented and motivated. They will be used in many different guises throughout the thesis.

A special application of maximum entropy is the task of estimating a probability distribution when some marginal distributions are known. The iterative proportional fitting procedure can be used to calculate the maximum entropy distribution consistent with the marginals. Belief propagation in junction trees can be used to implement this scheme more efficiently.

Another result of the subjectivist approach to probability is the Bayesian method for parameter estimation. It was presented along with its application to *EDA* and the connection to mutation in genetic algorithms.

5. Maximum Entropy and Sampling From Graphical Models

This chapter presents a new way to incorporate the maximum entropy principle into estimation of distribution algorithms.

First, in Sect. 5.1, a new heuristic method for building a factorization for an additively decomposable function is introduced. By merging the subfunctions of the ADF, it captures all the dependencies of the variables. Only if this results in too large dependency sets, some of them must be disregarded.

This algorithm is applied on the 2-D grid, resulting in a new factorization of the grid called *pentavariate*, because it uses sets of five variables.

Then, Sect. 5.2 presents maximum entropy sampling in the context of EDA, for the special case of polytrees (singly connected models). Previous work [OHSM03] is presented, and the role of the running intersection property is investigated for the example of polytrees.

Sect. 5.3 then presents a new development again, combining the work of the previous sections: The concept of maximum entropy sampling is applied to merged factorizations, yielding the new algorithm *MEFDA* (Maximum Entropy *FDA*). Finally, numerical results are presented which demonstrate the value of the new algorithms.

5.1. Manipulation of the Factorization Graph

5.1.1. Merging subfunctions

In Sect. 2.4.9, an additive decomposition of $f(\mathbf{x})$ was turned into a factorization system by choosing a subset of the dependencies. This is only an approximation, which disregards some possibly important dependencies.

Another possibility is, instead of leaving out some subfunctions, merging them. By merging subfunctions, it is possible to create a factorization system that accounts for all connections and even complies with the running intersection property. Actually, this is obvious, since the trivial *complete* merge of all subfunctions has this property. This is of course useless, because such big sets of variables make the algorithms exponentially costly.

We now present a heuristic algorithm that merges subfunctions in order to use all dependencies, but minimizing the number of merges. Like in Sect. 2.4.9, it builds a factorization system $\{\tilde{s}_j\}$. We use the definitions of $\tilde{c}_j$, $\tilde{b}_j$ and $\tilde{d}_j$ analogous to (2.23). We also need the information which variables are dependent of each other, i. e. which appear together in a subset s_i.

The idea of algorithm 5.1 is that each new variable is added in a set with the previous variables on which it depends. However, if another variable depends on a superset of variables, the two are merged and added together.

Algorithm 5.1: Subfunction Merger

```
1  S ⇐ {s_1, ..., s_m}
2  j ⇐ 1
3  while d̃_j ≠ {1, ..., n} do {
4     Choose an s_i ∈ S to be added, the same way as in Alg. 2.6
5     S ⇐ S \ {s_i}
6     Let the new variable indices in s_i be b_i = {k_1, ..., k_l}
7     for λ = 1 to l do {
8        Let δ_λ be all variables in d̃_{j-1} on which k_λ depends
9     }
10    for λ = 1 to l do {
11       if exists λ' ≠ λ with δ_λ ⊆ δ_λ' and not exists λ'' with δ_λ' ⊂ δ_λ''
12          δ_λ' ⇐ δ_λ' ∪ {k_λ}
13          Mark k_λ superfluous
14    }
15    for λ = 1 to l do {
16       if not k_λ superfluous
17          s̃_j ⇐ δ_λ ∪ {k_1, ..., k_λ}
18          while |s̃_j| > maximal cluster size do {
19             Choose randomly an index k from c̃_j
20             Remove k from s̃_j
21          }
22          j ⇐ j + 1
23    }
24 }
```

Every time we add a new subset s_i, we need to consider the new variables b_i to be added to the factorization system. Let the variable indices in b_i be $b_i = \{k_1, \ldots, k_l\}$. For each $\lambda \in \{1, \ldots, l\}$, the indices of all previous variables on which x_{k_λ} depends are stored in δ_λ. However, if there is a λ' with $\delta_\lambda \subseteq \delta_{\lambda'}$, it suffices to add only one set $\{k_\lambda, k_{\lambda'}\} \cup \delta_{\lambda'}$. Therefore k_λ is marked superfluous and, instead, added to $\delta_{\lambda'}$ in order to be added with this subset.

For densely connected problems, it can occur that the sizes of the sets $|\tilde{s}_j|$ become too large. Therefore we introduce a maximal size of the subsets. If this is exceeded, we remove variables from $\tilde{s}_j$ randomly. We remove only dependencies on previous variables, i. e. indices from $\tilde{c}_j$, not indices of new variables from $\tilde{b}_j$.

5.1.2. An Important Example: Pentavariate Factorization of the 2-D Grid

To make the algorithm clearer, here is an example. A very important dependency structure is the 2-D grid. An exact factorization of the grid would result in cliques of linear size [MM72] and therefore in exponentially large marginal distributions. Now we present an approximate, but polynomial factorization.

Sometimes the variables of the grid depend on their four direct neighbors only, sometimes also on the diagonal neighbors. An instance of the latter case is when the subfunctions of the ADF are defined on 2×2 blocks.

An example function is the *Deceptive 4-Grid* with

$$f_{\text{Dec4}}(u := x_{i,j} + x_{i+1,j} + x_{i,j+1} + x_{i+1,j+1}) = \begin{cases} 3 & \Longleftrightarrow u = 0 \\ 2 & \Longleftrightarrow u = 1 \\ 1 & \Longleftrightarrow u = 2 \\ 0 & \Longleftrightarrow u = 3 \\ 4 & \Longleftrightarrow u = 4 \end{cases} \tag{5.1}$$

and the fitness function being the sum of these subfunctions for all 2×2 blocks, with overlapping blocks:

$$F_{\text{Dec4}}(\mathbf{x}) = \sum_{i=1}^{m-1} \sum_{j=1}^{m-1} f_{\text{Dec4}}(x_{i,j} + x_{i+1,j} + x_{i,j+1} + x_{i+1,j+1}) \tag{5.2}$$

We present as an example a 4×4 grid with the variables indexed by 1 through 16. It induces directly the following factorization system:

$$\begin{array}{lll} s_1 = \{1,2,5,6\} & s_2 = \{2,3,6,7\} & s_3 = \{3,4,7,8\} \\ s_4 = \{5,6,9,10\} & s_5 = \{6,7,10,11\} & s_6 = \{7,8,11,12\} \\ s_7 = \{9,10,13,14\} & s_8 = \{10,11,14,15\} & s_9 = \{11,12,15,16\} \end{array} \tag{5.3}$$

This is a valid factorization system, violating the RIP. It was already presented in [Mah01, Sect. 4.2.6]. It is possible to sample points from it, using the probability distribution

$$\begin{aligned} p_{\text{Fact4}}(\mathbf{x}) = {} & p(x_1, x_2, x_5, x_6) p(x_3, x_7 | x_2, x_6) p(x_4, x_8 | x_3, x_7) \\ & p(x_9, x_{10} | x_5, x_6) p(x_{11} | x_6, x_7, x_{10}) p(x_{12} | x_7, x_8, x_{11}) \\ & p(x_{13}, x_{14} | x_9, x_{10}) p(x_{15} | x_{10}, x_{11}, x_{14}) p(x_{16} | x_{11}, x_{12}, x_{15}) \end{aligned} \tag{5.4}$$

The Bayesian network for this distribution is depicted in Fig. 5.1.

It can be seen that there are edges missing, namely $x_7 - x_{10}$, $x_8 - x_{11}$, $x_{11} - x_{14}$ and $x_{12} - x_{15}$. So we recognize that this factorization system is not using all dependencies. Now we will investigate in detail what happens when we apply Alg. 5.1 on it. We assume that all sets are added in ascending order.

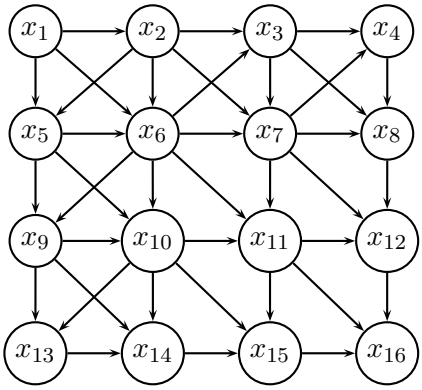

Figure 5.1.: The Bayesian network for the example 4×4 grid

Adding the first subset. We begin by adding $s_1 = \{1, 2, 5, 6\}$. Since $d_0 = \emptyset$, all δ_λ are empty, so three of the four variables are superfluous. We end up adding only one set, just $\tilde{s}_1 = \{1, 2, 5, 6\} = s_1$ itself.

The first row. We continue with $s_2 = \{2, 3, 6, 7\}$. The new variables are $b_2 = \{3, 7\}$, both having $\delta_\lambda = \{2, 6\}$. So once again, one is superfluous, and again we add the unchanged set, $\tilde{s}_2 = \{2, 3, 6, 7\}$. With $s_3 = \{3, 4, 7, 8\}$, the same thing happens.

The second row. We now turn to $s_4 = \{5, 6, 9, 10\}$. The new variables are $k_1 = 9$ and $k_2 = 10$, having $\delta_1 = \{5, 6\}$ and $\delta_2 = \{5, 6, 7\}$. Since $\delta_1 \subseteq \delta_2$, k_1 is marked superfluous. We add the set $\tilde{s}_4 = \{5, 6, 7, 9, 10\}$. Now for the first time, the factorization differs from (5.3). A 5-variate set enters.

The next set has only one new variable index, $k_1 = 11$. It depends on $\delta_1 = \{6, 7, 8, 10\}$, so we add $\tilde{s}_5 = \{6, 7, 8, 10, 11\}$. The final set of this row contains four indices again, $\tilde{s}_6 = \{7, 8, 11, 12\}$.

The third row. The third row is built in the same way as the second row.

The complete factorization system is

$$\begin{array}{lll} \tilde{s}_1 = \{1, 2, 5, 6\} & \tilde{s}_2 = \{2, 3, 6, 7\} & \tilde{s}_3 = \{3, 4, 7, 8\} \\ \tilde{s}_4 = \{5, 6, 7, 9, 10\} & \tilde{s}_5 = \{6, 7, 8, 10, 11\} & \tilde{s}_6 = \{7, 8, 11, 12\} \\ \tilde{s}_7 = \{9, 10, 11, 13, 14\} & \tilde{s}_8 = \{10, 11, 12, 14, 15\} & \tilde{s}_9 = \{11, 12, 15, 16\} \end{array} \tag{5.5}$$

In the Bayesian network (Fig. 5.2) we see that the missing edges have been included.

The factorization (5.5) of the 2-D grid is called the *pentavariate factorization*, because it contains subsets of size 5.

Definition 5.1. Let there be given an $m \times m$ 2-D grid of variables $x_{i,j}$ with $i, j \in$

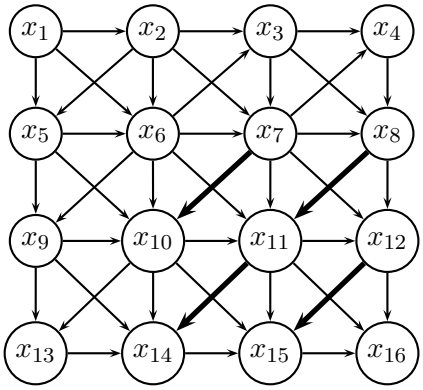

Figure 5.2.: The example 4×4 grid and its pentavariate factorization. The fat arrows are the added dependencies.

$\{1, \ldots, m\}$. The **pentavariate factorization** of such a grid is given by

$$p(\mathbf{x}) = p(x_{1,1}, x_{1,2}, x_{2,1}, x_{2,2}) \prod_{i=3}^{m} p(x_{i,1}, x_{i,2} | x_{i-1,1}, x_{i-1,2})$$
$$\prod_{j=3}^{m} \Bigg(p(x_{1,j}, x_{2,j} | x_{1,j-1}, x_{2,j-1}, x_{3,j-1}) \prod_{i=3}^{m-1} p(x_{i,j} | x_{i-1,j}, x_{i-1,j-1}, x_{i,j-1}, x_{i+1,j-1})$$
$$p(x_{m,j} | x_{m-1,j}, x_{m-1,j-1}, x_{m,j-1}) \Bigg) \quad (5.6)$$

It also violates the RIP. E. g. in the marginal $p(x_{11}|x_6, x_7, x_8, x_{10})$, the indices $\{6, 7, 8, 10\}$ do not previously show up in the same subset together.

Remark 5.1. Alg. 5.1 chooses the ordering of the sets to be added not in ascending order, like in this example, but using Alg. 2.6. An example for a distribution chosen by the algorithm is

$$p(\mathbf{x}) = p(x_9, x_{10}, x_{13}, x_{14}) p(x_{11}, x_{15} | x_{10}, x_{14}) p(x_5, x_6 | x_9, x_{10}, x_{11})$$
$$p(x_7 | x_6, x_{10}, x_{11}) p(x_{12}, x_{16} | x_7, x_{11}, x_{15}) p(x_8 | x_7, x_{11}, x_{12})$$
$$p(x_1, x_2 | x_5, x_6, x_7) p(x_3 | x_2, x_6, x_7, x_8) p(x_4 | x_3, x_7, x_8) \quad (5.7)$$

It uses all dependencies, too, but some of the edges in Fig. 5.2 are reversed.

Numerical results of these factorizations are presented in Sect. 5.4. But before, a way to combine this algorithm with maximum entropy is introduced, first for singly connected Bayesian networks, then in Sect. 5.3 for factorizations.

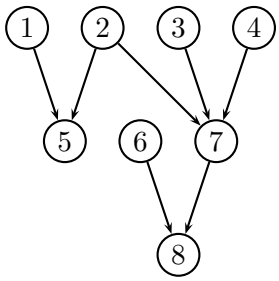

Figure 5.3.: An example polytree.

5.2. Polytrees, PADA, and Maximum Entropy

This section will present the PADA (Polytree Approximation Distribution Algorithm) [SO00, Sot03]. Whereas *FDA* uses a given factorization, PADA learns a graphical model from the selected data and then samples new data from it. The difference is that it learns a *polytree*, a singly connected Bayesian network. Belief propagation on polytrees was discussed thoroughly by Pearl [Pea88].

An improvement of this algorithm using maximum entropy [OHSM03] is described. In Sect. 5.3 this idea will be generalized to multiply connected networks by combining it with the techniques of Sect. 5.1.

5.2.1. Polytrees

Definition 5.2. A **polytree** is a singly connected (tree-like) Bayesian network. Removing the directions of all edges of the polytree, we obtain a tree which we call the **skeleton** of the polytree.

Lemma 5.1. *A polytree for n variables has $n-1$ edges.*

A small example for a polytree is shown in Fig. 5.3. It has 8 nodes and 7 edges. But if we added, e. g., an edge between X_2 and X_6, it would still be a Bayesian network, but no longer a polytree, because of the cycle $X_2 - X_6 - X_8 - X_7 - X_2$ in the skeleton.

Polytrees are a computationally favorable subclass of Bayesian networks. They retain many computational advantages of tree structures, but are more powerful. Whereas in a tree, each variable can be conditioned only on at most one other variable, in a polytree a variable can depend on more than one variable. E. g. in Fig. 5.3, X_5 depends on X_1 and X_2.

Definition 5.3. In a polytree, a connection of the kind $X \rightarrow Z \leftarrow Y$ is called a **head-to-head** connection.

These head-to-head connections constitute the gain in expressive power, in comparison to trees. Note as well that polytrees in general violate the RIP (2.41) (see Fig. 3.5).

5.2.2. The PADA2 Algorithm

The algorithm PADA2 is the special case of PADA which works with second-order marginal distributions. We present briefly the algorithm to construct a polytree from a set of selected data. Details can be found in [SO00, Sot03].

- Construct a maximum weight spanning tree on the set of variables, using as weight of an edge $\{X_i, X_j\}$ the mutual information of the variables $I(X_i, X_j)$ in the distribution given by the data.

  In [Sot03] it was shown that it is computationally favorable to use (4.12) for calculating the mutual informations.

  The spanning tree can be constructed by a greedy algorithm, described in [CL68]. The result is the skeleton of the polytree.

- Next, the directions of the edges must be added. Supposing there is a connection $X - Z - Y$ in the skeleton, and we find that the mutual information $I(X, Y)$ is smaller than a given threshold, we direct both edges toward Z, creating a head-to-head connection $X \to Z \leftarrow Y$.

  After this step, all other connections are directed at random, without creating any more head-to-head connections (if this is possible).

Having constructed the polytree, PADA2 continues like *FDA*, calculating the needed conditional probabilities and sampling a new population from the polytree, using (3.19).

5.2.3. Head-to-Head Connections: RIP Revisited

We demonstrate the algorithm and the properties of the head-to-head connection with a small example. Suppose that there is a population of $N = 100$ individuals for the three variables X, Y, Z. The relative frequencies of all assignments are given in the column p_{Pop} in Table 5.1.

The mutual informations of the variable pairs are

$$I(X, Y) = 0.701471 + 0.964800 - 1.661366 = 0.004905\, bit \tag{5.8}$$

$$I(X, Z) = 0.701471 + 0.881291 - 1.500873 = 0.081889\, bit \tag{5.9}$$

$$I(Y, Z) = 0.964800 + 0.881291 - 1.789480 = 0.056611\, bit \tag{5.10}$$

Since X and Y have a very small mutual information, PADA2 chooses the polytree skeleton $X - Z - Y$.

There are four ways to direct the two edges $X - Z$ and $Z - Y$. Because of the Bayesian Law $p(a)p(b|a) = p(b)p(a|b) = p(a, b)$, three of them lead to the same probability distribution, the serial connection

$$p_{\text{Ser}}(x, y, z) = \frac{p(x, z)p(y, z)}{p(z)} \tag{5.11}$$

x	y	z	p_{Pop}	p_{Ser}	p_{H2H}	p_{Indep}	p_{MaxEnt}	$p_{\text{ME},X \perp\!\!\!\perp Y}$
0	0	0	0.1100	0.1080	0.1158	0.0948	0.1097	0.1193
0	0	1	0.1900	0.1890	0.2001	0.2211	0.1903	0.1966
0	1	0	0.0700	0.0720	0.0678	0.1482	0.0703	0.0607
0	1	1	0.4400	0.4410	0.4263	0.3459	0.4397	0.4334
1	0	0	0.0700	0.0720	0.0576	0.0222	0.0703	0.0607
1	0	1	0.0200	0.0210	0.0165	0.0519	0.0197	0.0134
1	1	0	0.0500	0.0480	0.0580	0.0348	0.0497	0.0593
1	1	1	0.0500	0.0490	0.0580	0.0811	0.0503	0.0566
		$H(.)$	2.4088	2.4091	2.4234	2.5476	2.4088	2.4003
		$D(.\|\|p_{\text{Pop}})$	0	0.0002	0.0049	0.1413	$8 \cdot 10^{-6}$	0.0062

Table 5.1.: Example distributions for PADA2. p_{Pop} is given by the population, p_{Ser} is the distribution for the serial connection $X \to Z \to Y$, p_{H2H} for the head-to-head connection $X \to Z \leftarrow Y$, p_{Indep} is the product of the univariate distributions, p_{MaxEnt} is the maximum entropy distribution given the three bivariate marginals, $p_{\text{ME},X \perp\!\!\!\perp Y}$ is the maximum entropy distribution, given $p(x,z)$, $p(y,z)$ and independence of x and y. $H(.)$ gives the entropy of the distribution, $D(.\|p_{\text{Pop}})$ the Kullback-Leibler divergence to the population distribution; both in *bits*, using the binary logarithm.

The fourth is the head-to-head connection. It gives the distribution

$$p_{\text{H2H}}(x,y,z) = p(x)p(y)p(z|x,y) \tag{5.12}$$

For comparison, in Table 5.1 the independent product distribution

$$p_{\text{Indep}}(x,y,z) = p(x)p(y)p(z) \tag{5.13}$$

is also included.

In this particular example, the serial connection is closer to the population distribution than the head-to-head connection, but of course this is not generally the case. Interesting properties of the distributions can be recognized by comparing the bivariate marginal distributions, given in Table 5.2.

First we notice that $p_{\text{H2H}}(x,y)$ is equal to $p_{\text{Indep}}(x,y)$.

Lemma 5.2. *In p_{H2H}, $X \perp\!\!\!\perp Y$. In p_{Ser}, this is not necessarily the case.*

Proof.

$$p_{\text{H2H}}(x,y) = \sum_z p(x)p(y)p(z|x,y) \tag{5.14}$$

$$= p(x)p(y)\sum_z p(z|x,y) \tag{5.15}$$

$$= p(x)p(y) \tag{5.16}$$

x	y	p_{Pop}	p_{Ser}	p_{H2H}	p_{Indep}	p_{MaxEnt}	$p_{\text{ME},X \perp\!\!\!\perp Y}$
0	0	0.3000	0.2970	0.3159	0.3159	0.3000	0.3159
0	1	0.5100	0.5130	0.4941	0.4941	0.5100	0.4941
1	0	0.0900	0.0930	0.0741	0.0741	0.0900	0.0741
1	1	0.1000	0.0970	0.1159	0.1159	0.1000	0.1159
x	z	p_{Pop}	p_{Ser}	p_{H2H}	p_{Indep}	p_{MaxEnt}	$p_{\text{ME},X \perp\!\!\!\perp Y}$
0	0	0.1800	0.1800	0.1836	0.2430	0.1800	0.1800
0	1	0.6300	0.6300	0.6264	0.5670	0.6300	0.6300
1	0	0.1200	0.1200	0.1156	0.0570	0.1200	0.1200
1	1	0.0700	0.0700	0.0744	0.1330	0.0700	0.0700
y	z	p_{Pop}	p_{Ser}	p_{H2H}	p_{Indep}	p_{MaxEnt}	$p_{\text{ME},X \perp\!\!\!\perp Y}$
0	0	0.1800	0.1800	0.1735	0.1170	0.1800	0.1800
0	1	0.2100	0.2100	0.2165	0.2730	0.2100	0.2100
1	0	0.1200	0.1200	0.1258	0.1830	0.1200	0.1200
1	1	0.4900	0.4900	0.4842	0.4270	0.4900	0.4900

Table 5.2.: The bivariate marginals of the distributions in Table 5.1

□

Lemma 5.3. *$p_{Ser}(x,z) = p_{Pop}(x,z)$ and $p_{Ser}(y,z) = p_{Pop}(y,z)$. For p_{H2H}, this is usually not the case.*

Proof.

$$p_{\text{Ser}}(x,z) = \sum_y \frac{p(x,z)p(y,z)}{p(z)} \tag{5.17}$$

$$= p(x,z) \sum_y \frac{p(y,z)}{p(z)} \tag{5.18}$$

$$= p(x,z) \tag{5.19}$$

□

It may be a disadvantage of the head-to-head connection that its bivariate marginals differ from the ones used to construct it. This is the curse of the violated RIP.

Another possibility to construct a distribution of the three variables is IPF, using the maximum entropy principle. $p_{\text{MaxEnt}}(x,y,z)$ has been calculated by IPF using the three bivariate marginal distributions. $p_{\text{ME},X \perp\!\!\!\perp Y}$ was calculated by IPF using $p(x,z)$, $p(y,z)$ and $p(x)p(y)$. So it is the maximum entropy distribution conforming to the former two bivariate marginals and assuming $X \perp\!\!\!\perp Y$. This is well visible in Table 5.2 by comparison with p_{Pop} and p_{Indep}.

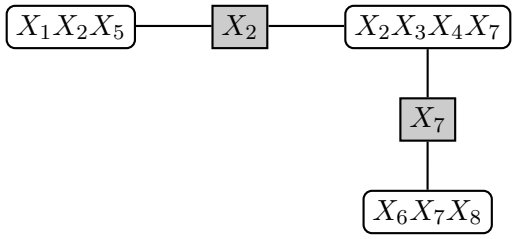

Figure 5.4.: The junction tree for the polytree of Fig. 5.3.

Note that this is different from IPF using $p(x, z)$ and $p(y, z)$ only. This by default does not assume independence of X and Y, only their conditional independence given Z. In fact, IPF using $p(x, z)$ and $p(y, z)$ computes the distribution p_{Ser} (see Corollary 4.8).

It is no surprise that p_{MaxEnt} is closest to the original distribution: This is the only distribution that uses the information $p(x, y)$.

5.2.4. From Polytree to Junction Tree

A problem of PADA2 is that while it needs only bivariate distributions for constructing the polytree, the marginals needed for sampling are larger.

For example, for the polytree in Fig. 5.3 the sampling distribution is

$$p_{\text{Poly}}(\mathbf{x}) = p(x_1)p(x_2)p(x_3)p(x_4)p(x_5|x_1, x_2)p(x_6)p(x_7|x_2, x_3, x_4)p(x_8|x_6, x_7) \tag{5.20}$$

Distributions up to size four are needed, namely $p(x_7|x_2, x_3, x_4)$. The degrees of freedom of such a conditional distribution grows exponentially with the number of parents. Accordingly, also the population size required to estimate the distribution with the necessary accuracy grows. So what has been gained by choosing a quite restricted graphical model which can be learned with limited effort can be lost again in the sampling step.

In [OHSM03], a method to solve this problem is presented. It constructs the needed higher-order marginal distributions from the bivariate marginals (which have already been computed for the learning step) by IPF on a junction tree, as described in Sect. 4.2.7.

Constructing a junction tree from a polytree is very simple. After *marrying the parents* and thus constructing the moral graph of the polytree, we notice that it is already triangulated. The junction tree consists of one cluster C_i for each variable i with $\Pi_i \neq \emptyset$, containing $C_i = \{X_i\} \cup \Pi_i$, the child and all its parents.

All separators of such a junction tree contain only one variable. For example, the polytree in Fig. 5.3 is turned into the junction tree given in Fig. 5.4.

One separator contains X_2, because it is parent of two variables (X_5 and X_7) and therefore connects their clusters. The other separator contains X_7, which is child and parent at the same time.

Lemma 5.4. *The IPF procedure on such a junction tree converges after only one step.*

Proof. The IPF procedure of Sect. 4.2.7 consists of a sequence of IPF and message passing on a closed tour through the junction tree. Now consider a separator S between two nodes C and C'. It contains only one variable, say X_i. The marginals that are used for IPF in C and C' are consistent. So, IPF on C and on C' result in the same univariate marginal distribution of X_i. This means that after one step of IPF in both C and C', message passing has no effect any more. Then IPF does not change the distributions either. □

In more general Bayesian networks, it is possible that information about the dependencies of the variables in S is contained only in one of the neighboring nodes, but not in the other. But if S contains only one variable, and both clusters contain marginals with complete information about this variable, the algorithm converges after one step.

The distribution on the junction tree (3.33) is for our example

$$p_{\mathrm{JT}}(\mathbf{x}) = p(x_1, x_2, x_5)p(x_3, x_4, x_7|x_2)p(x_6, x_8|x_7) \tag{5.21}$$

It turns out to be favorable to sample from this distribution, rather than from (5.20).

In [OHSM03], numerical results are presented comparing this sampling method with the conventional way. It was found that for a separable deceptive function with five blocks of order 4 (having 20 variables), with a population size of $N = 800$, the maximum entropy method has a success rate of 93 %, whereas the conventional polytree sampling succeeds only in 16 % of the cases. For comparison: *LFDA* (with truncation selection threshold $\tau = 0.1$) has a success rate of 77.3 % on the mentioned problem[1].

Improvements are similarly impressive on all three given benchmark functions. I will not repeat the results here. Instead, the next section presents the application of the maximum entropy method on a multiply connected model, as has been announced in the conclusion of [OHSM03].

5.3. MEFDA: IPF for Merged Factorizations

Joining subfunctions has a severe disadvantage: By using larger marginal distributions, more connections between nodes are captured, but the number of degrees of freedom grows exponentially in the size of the marginal.

Suppose e. g. that a distribution $p(X_i|\Pi_i)$ is to be estimated from a selected data set. The number of degrees of freedom of such a distribution is $2^{|\Pi_i|}$: for each one of the $2^{|\Pi_i|}$ possible values of the parents, the probability $p(X_i = 1|\pi_i)$ must be chosen. Given a population to estimate these probabilities, suppose that $N(\pi_i)$ is the number of individuals in which the variables Π_i have the values π_i, and $N(x_i, \pi_i)$ is the number of individuals in which X_i and Π_i have the respective values. Then the classical estimate (without Bayesian priors, see Sect. 4.3.2) is

$$p(x_i|\pi_i) = \frac{N(x_i, \pi_i)}{N(\pi_i)} \tag{5.22}$$

[1] The structure learning algorithm *LFDA* is described in Chapter 7. The parameters of this run were: Deceptive-5, 4 blocks, $N = 800$, $\tau = 0.1$, $\alpha = 0.5$, average over 10000 runs. It needed 2.32 generations on average, with standard deviation 0.693.

For constant N, the numbers $N(x_i, \pi_i)$ and $N(\pi_i)$ will decrease exponentially in $|\Pi_i|$. Thus the estimate is more sensitive to noise in the data. In the most extreme case, there are many assignments π_i for which $N(x_i, \pi_i) = N(\pi_i) = 0$, so there is no information at all in the population. Larger distributions require a larger population size to estimate.

We see that the problem is identical to the one mentioned about the polytrees in Sect. 5.2.4. Therefore, we propose the same solution: Instead of estimating the large distribution from the population, use the maximum entropy distribution, given smaller marginals.

We recall that in the merging algorithm (Alg. 5.1), from a factorization system $\mathfrak{S} = \{s_j\}, j = 1, \ldots, m$, a new factorization system $\tilde{\mathfrak{S}} = \{\tilde{s}_j\}, j = 1, \ldots, \tilde{m}$ is formed, consisting of supersets of the s_j.

Therefore, for a set $\tilde{s}_j \in \tilde{\mathfrak{S}}$ we use the set

$$\mathfrak{S}_{\mathrm{IPF},j} = \{s_i \cap \tilde{s}_j | s_i \in \mathfrak{S} \wedge s_i \cap \tilde{s}_j \neq \emptyset\} \tag{5.23}$$

as the smaller marginals for IPF. The choice can be improved by removing every set from $\mathfrak{S}_{\mathrm{IPF},j}$ of which a superset is already contained. This affects only the running time, not the result of IPF.

To make the resulting algorithm clearer, we put together all the parts described so far and present the complete pseudocode in Alg. 5.2.

Algorithm 5.2: *MEFDA* – Maximum Entropy *FDA*

1. Using Alg. 5.1, construct a merged factorization system $\tilde{\mathfrak{S}}$ from the given additive decomposition $\mathfrak{S}$
2. $t \Leftarrow 1$. Generate an initial population with N individuals from the uniform distribution.
3. **do** {
4. Perform selection
5. Estimate the marginal probabilities $p(\mathbf{x}_{s_i}, t)$ from the selected points.
6. Calculate all $p(\mathbf{x}_{\tilde{s}_j}, t)$ by IPF using $\mathfrak{S}_{\mathrm{IPF},j}$
7. Generate new points according to $p(\mathbf{x}, t+1) = \prod_{j=1}^{m} p(\mathbf{x}_{\tilde{b}_j} | \mathbf{x}_{\tilde{c}_j}, t)$.
8. $t \Leftarrow t + 1$.
9. } **until** stopping criterion reached

5.4. Numerical Results

In this chapter were presented two new variants of the *FDA* algorithm:

- *FDA* with merging of subfunctions using Alg. 5.1 (instead of picking a subset of the factorization sets using Alg. 2.6) and
- *MEFDA* (Alg. 5.2), merging subfunctions like above, but using IPF to construct the larger marginal distributions.

Grid size	Pop. size	*FDA* SR	*FDA* Gen ± SD	*FDA* with merge SR	*FDA* with merge Gen ± SD	*MEFDA* SR	*MEFDA* Gen ± SD
10×10	700	79	8.58 ± 0.810	70	7.81 ± 0.748	98	7.58 ± 0.745
= 100	900	92	8.54 ± 0.895	91	7.44 ± 0.733	100	7.33 ± 0.667
	1000	95	8.36 ± 0.757	99	7.45 ± 0.689	100	7.22 ± 0.786
15×15	1200	34	15.38 ± 0.853	64	13.17 ± 1.121	96	13.15 ± 0.754
= 225	1600	56	14.68 ± 1.081	96	12.57 ± 0.692	98	12.83 ± 0.643
	2000	71	14.70 ± 0.852	98	12.10 ± 0.601	100	12.48 ± 0.643
	2700	89	14.29 ± 0.920	100	11.92 ± 0.580	100	12.07 ± 0.537
20×20	1900	8	21.88 ± 0.641	64	18.25 ± 0.836	97	18.75 ± 0.902
= 400	2000	5	21.20 ± 1.095	77	17.94 ± 0.767	97	18.43 ± 0.900
	2400	17	21.65 ± 0.996	95	17.65 ± 0.796	100	18.28 ± 0.697
	3000	31	21.10 ± 1.106	99	17.16 ± 0.584	100	18.05 ± 0.609
25×25	3300	2	27.50 ± 0.707[2]	94	22.61 ± 0.722	99	24.31 ± 0.804
= 625	3600	3	28.33 ± 0.578	99	22.44 ± 0.610	99	24.11 ± 0.754

Table 5.3.: Results of the joining and Maximum Entropy methods on the deceptive grid (5.2) of different grid sizes and population sizes. *SR* gives the success rate (number of successful runs), *Gen* the average number of generations until success and *SD* its standard deviation. Parameters: 100 runs each. Truncation Selection with $\tau = 0.3$. Maximal number of generations is 20 for the 10×10 and 15×15 grids, 30 for the bigger grids.

Now we compare these algorithms with conventional *FDA* (Alg. 2.5). Their results on the deceptive grid function (5.2) are depicted in Table 5.3.

We are mostly interested in the minimal population size for which the algorithms have a large success rate (above 95 %). It can be noticed that both new algorithms perform better than *FDA*. They manage to reduce the population size impressively. Also, the number of required generations until convergence is reduced. Since the number of function evaluations is proportional to the population size and the number of generations, the methods allow large savings in the number of function evaluations.

FDA with merge and *MEFDA* need a comparable number of generations until convergence. *MEFDA* needs slightly more generations, but manages to decrease the required population size even more than *FDA* with subfunction merge. The price to pay for this is the additional overhead of the IPF procedures, but it is decent for not too large set sizes, and furthermore, IPF does not need any fitness function evaluations. So these methods

[2]Note here that the unbiased estimator for the standard deviation is used:

$$\sigma_{N-1} = \sqrt{\frac{1}{N-1}\sum_x (x-\mu)^2}$$

This is necessary because the formula needs the sample mean μ, which is also estimated from the same data. Therefore, the standard formula which divides by N instead of $N-1$ would underestimate the sample variance. The difference only matters for such small sample sizes as here, with only two successful runs.

are particularly interesting for problems in which evaluation of the fitness function is expensive.

The difference between *FDA* with join and *MEFDA* becomes smaller with increasing grid size. It can be seen in Table 5.3 that for the 25×25 grid the success rates are almost the same. The reason for this is that both algorithms need a larger population for large grids. Large populations lead to better estimations of the marginal distributions. Therefore, the large marginals created by the joining algorithm can be estimated well from the population, and the precision gain of IPF is lost.

5.5. Summary

This chapter presented several new variants and improvements of *FDA* using the maximum entropy principle. First, a new method to join the subfunctions of the objective function is introduced. This allows to build a factorization system without having to disregard known connections between the variables (as long as the factorization does not become too large).

Then, a previous work of the author is resumed, which uses maximum entropy to improve a special EDA. In this case, the graphical models are confined to singly connected Bayesian networks (polytrees). The properties of this algorithm are investigated more extensively, and the role of the running intersection property in this algorithm is illuminated.

Finally, a new variant of *FDA* was presented which generalizes this work to multiply connected Bayesian networks and combines it with the subfunction join algorithm. Numerical results of this algorithm are presented and its performance is compared with conventional *FDA*, without or with subfunction join.

6. The Bethe-Kikuchi Approximation and Loopy Belief Models

Conventional graphical models do not have loops. Bayesian networks are acyclic (considering the direction of edges), and junction trees are trees.

Constructing a feasible junction tree is in general NP-complete. The triangulation step often results in too large clique sizes. One prominent example, which is very important in practice, is the 2-D $m \times m$ grid. Triangulation of a grid results in cliques of size $\Omega(m)$ [BB72, MM72].

But it is also possible to generalize the graphical models to loopy structures. This has already been proposed by Pearl in [Pea88]. But recently, loopy belief propagation has been combined with free energy minimization from statistical physics [AM01, MY02, YFW01, YFW04]. This idea can be used for optimization using probabilistic models [MH05].

Since the loopy models are inspired and motivated by statistical physics, in Sect. 6.1 we give some introduction and foundation in statistical physics. Then, in Sect. 6.2 we introduce the loopy graphical model, the region graph (also called partially ordered set or poset), and in Sect. 6.3 the generalized belief propagation (GBP) algorithm derived by minimizing the Bethe free energy in the region graph, which is shown to be equivalent to minimizing the relative entropy to the Boltzmann distribution.

These presentations lay the grounds for the new optimization algorithm *BKDA* (Bethe Kikuchi Distribution Algorithm), which is described in Sect. 6.4. We show how to generate a factorization from which points can be sampled, and how to build a region graph using the cluster variation method (CVM). The results on important example structures, namely a bi- and trivariate cycle and the 2-D grid, are given as we go along.

Sect. 6.5 presents numerical results on these structures, particularly for the Ising problem. This demonstrates the applicability of the new approach.

GBP is not guaranteed to converge. A concave-convex procedure (CCCP) was derived in [Yui02] which leads to a more stable convergence and can readily replace GBP in *BKDA*. This is presented in Sect. 6.6.

6.1. Foundation in Statistical Physics

We recall from section 2.4.1 that our goal is to sample from a Boltzmann distribution (2.12). In general, calculating the Boltzmann distribution requires exponential time. In this chapter, we consider approximations from statistical physics, and explain how they can be used for optimization in computer science.

We use a very important example, which is also well-known in statistical physics: The 2-D Ising spin glass model [Isi25, Pei36, MPV87].

6.1.1. The Ising model

Suppose that there is given a grid of cells. We call the grid width m, so we have $n = m^2$ cells. A cell can be referenced by two coordinates $x, y \in \{1, \ldots, m\}$. We assign to each cell a unique index $\nu \in \{1, \ldots, n\}$, e. g. by setting

$$\nu(x, y) = m(y - 1) + x \tag{6.1}$$

We define a symmetric neighborhood relationship $\mathcal{N} \subseteq \{1, \ldots, n\}^2$ and say that two cells ν and μ are neighbored if $(\nu, \mu) \in \mathcal{N}$. We use 4-neighborhood, with $(\nu, \mu) \in \mathcal{N}$ if μ is the left-hand, right-hand, upper or lower neighbor of ν. We assume a fixed-boundary condition, so there is no wrap-around at the edges of the grid – e. g. $(\nu(x, 1), \nu(x, m)) \notin \mathcal{N}$.

Example 6.1. We give a simple example for this rather technical definition. Fig. 6.1 shows a simple 3×3 grid. The cells contain their indices ν. Neighbored nodes are connected by a line.

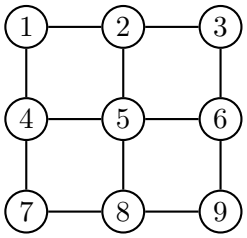

Figure 6.1.: An example 3×3 grid

Each cell ν contains a **spin** s_ν which can be in the state $+1$ or -1. A **state** of the grid is a vector of spins, one for each cell: $\mathbf{s} = (s_\nu)_{\nu=1,\ldots,n}$.

For each pair of spins on neighboring cells $(\nu, \mu) \in \mathcal{N}$ there is a **coupling constant** $J_{\nu,\mu} \in \mathbb{R}$. We demand $J_{\nu,\mu} = J_{\mu,\nu}$, and we assume $J_{\nu,\mu} = 0$ for $(\nu, \mu) \notin \mathcal{N}$. This results in the symmetric **energy matrix** $\mathbf{J} = (J_{\mu,\nu})_{\mu,\nu}$.

Using these couplings we define the **Hamiltonian energy** of a state $\mathbf{s}$ as

$$E(\mathbf{s}) = -\sum_{1 \le \nu < \mu \le n} J_{\mu,\nu} s_\mu s_\nu \tag{6.2}$$

The problem can also be generalized by adding a vector $\mathbf{H} = (H_\nu)_{\nu \in \{1,\ldots,n\}}$, which is called an **external magnetic field**. This changes the energy to

$$E(\mathbf{s}) = -\sum_{1 \le \nu < \mu \le n} J_{\mu,\nu} s_\mu s_\nu - \sum_{\nu=1}^{n} H_\nu s_\nu \tag{6.3}$$

The external magnetic field pushes each spin s_ν towards $+1$ or -1, depending on the sign of H_ν.

Our goal is to minimize this energy. A state with minimal energy is called a **ground state** of the grid. Ground state search is an important and difficult problem of statistical physics.

To understand why this problem is difficult, consider the following: For two cells ν, μ minimization is done by choosing

$$s_\nu = \begin{cases} s_\mu & \text{if } J_{\nu,\mu} > 0 \text{ (so both spins +1 or both -1)} \\ -s_\mu & \text{if } J_{\nu,\mu} < 0 \end{cases} \tag{6.4}$$

But on a grid, it is usually not possible to satisfy all these conditions.

Example 6.2. For example, let us consider a very simple 2×2 grid with the energy matrix given in Fig. 6.2.

$$\mathbf{J} = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \end{pmatrix}$$

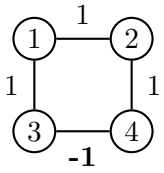

Figure 6.2.: An example 2×2 grid with its energy matrix

No matter how we choose the spins, at least one pair will have $J_{\mu,\nu} s_\mu s_\nu = -1$. We cannot satisfy all pairs and reach an energy of -4. The minimal energy for this energy matrix is -2.

Such a situation is called **frustration**.

For a 1-D Ising spin glass model, which is just a chain of cells, it is easy to find a ground state. Just choose the first cell arbitrarily, and then proceed through the chain, choosing the spins according to (6.4). This will give a ground state with no frustrations at all.

If the 1-D Ising model is constructed with wrap-around, so $(1, n) \in \mathcal{N}$, we proceed in the same way, only beginning not at cell 1, but at the cell ν with $|J_{\nu-1,\nu}|$ minimal. Thus, it may turn out that after choosing all spins this connection will be frustrated, but at least it will be the connection which gives a minimal penalty to the energy. An example is Fig. 6.2, which can also be considered as a 1-D model with $n = 4$ and wrap-around.

For a 2-D Ising grid, ground state search is much harder. Without an external magnetic field, Onsager [Ons44] found an ingenious solution. But the 2-D model with a magnetic field, as well as the 3-D model, are NP-complete [Bar82].

6.1.2. Symmetry of Ising

Without an external magnetic field, the problem is symmetric. If we flip all spins, the energy does not change; $E(-\mathbf{s}) = E(\mathbf{s})$. This makes the problem particularly difficult

for evolutionary algorithms [HGN01, Hoy03].

This can be seen from the work of Mahnig [Mah01]: He has shown that genetic algorithms perform a gradient ascent in the space of the expected fitness. But in this case, the derivation of the expected fitness with respect to a single spin is zero, so the gradient ascent will fail.

Example 6.3. Consider a 1-D ferromagnetic spin glass with wrap-around, so all $J_{\nu,\mu} = 1$ for $(\nu, \mu) \in \mathcal{N}$. Now imagine that we have a solution with $s_\nu = +1$ for $\nu < n/2$ and $s_\nu = -1$ otherwise. We have two frustrated couplings, $(n/2 - 1, n/2)$ and $(n, 1)$.

It is very hard to flip $n/2$ spins in one step, which is necessary to remove these frustrations and reach a ground state. But it is possible to move the boundaries between the $+1$ and -1 blocks. E. g. by flipping bit 1, one of the frustrations moves to $(1, 2)$.

Moving these boundaries will not change the energy, so it cannot be said whether it is better to move a boundary to the left or to the right. Instead, they will be shifted around in a *random walk* manner, blindly, until by chance one of the blocks vanishes. This will take a very long time to achieve.

We now describe a technique which exploits the symmetry in population-based algorithms in order to reduce the size of the search space. For this the following definition is needed:

Definition 6.1. For two spin vectors $\mathbf{s}$, $\mathbf{t}$, define the **Hamming distance** as the number of differing spins:

$$h(\mathbf{s}, \mathbf{t}) = |\{\nu \in \{1, \ldots, n\} | s_\nu \neq t_\nu\}| \tag{6.5}$$

Now for every generated vector $\mathbf{s}$, if the Hamming distance to the vector with maximal fitness among the population (minimal energy) $\mathbf{s}_{\max}$ is larger than $n/2$, we flip all bits of $\mathbf{s}$. This does not change the fitness of $\mathbf{s}$. Doing this for the whole population halves the search space. This procedure is called *flip bits to best*.

The method also improves the fitness of the offspring. This can be seen in an extreme example with the *UMDA* algorithm. Suppose that we start with a random population, and then we do truncation selection. Notice: In a problem where we can always flip all bits and obtain identical fitness, the univariate marginal probability for a single spin is $p(s_\nu = 1) = 1/2$. So in the next generation, we will have a completely random population again! "Flip bits to best" is a way to break this symmetry and help the algorithm to converge.

6.1.3. The Boltzmann Distribution

We have already introduced the Boltzmann distribution as a tool for optimization (see Sect. 2.4.1). For the Ising spin glass we note that minimizing $E(\mathbf{s})$ is identical to maximizing

$$p_\beta(\mathbf{s}) = \frac{1}{Z} e^{-\beta E(\mathbf{s})} \tag{6.6}$$

with $\beta > 0$.

To be more conform with the computer scientist notions, we can also replace the spins $s_\nu \in \{-1, +1\}$ by bits $x_\nu \in \{0, 1\}$ using the following transformation:

$$s_\nu = 2x_\nu - 1 \tag{6.7}$$

$$\tilde{J}_{\nu,\mu} = 4J_{\nu,\mu} \tag{6.8}$$

$$\tilde{H}_\nu = 2H_\nu - 4\sum_{\mu=1}^{n} J_{\nu,\mu} \tag{6.9}$$

$$\tilde{E}(\mathbf{x}) = -\sum_{1\leq\nu<\mu\leq n} \tilde{J}_{\mu,\nu}x_\mu x_\nu - \sum_{\nu=1}^{n} \tilde{H}_\nu x_\nu \tag{6.10}$$

It can be easily verified that $\tilde{E}(\mathbf{x}) = E(\mathbf{s}) + const$, so that $p_\beta(\mathbf{s}) = p_\beta(\mathbf{x})$ (the constant difference cancels out due to the partition function Z).

Because of this equivalence, we will from now on stick to the computer science notation and use the variables $\mathbf{x} \in \{0, 1\}^n$.

6.1.4. The Gibbs Free Energy

Once again, we are faced with the problem of calculating the Boltzmann distribution or sampling from it. This is not possible in an efficient way. So the idea is to replace the Boltzmann distribution p_β by an approximation distribution q. For this, we introduce some terms from statistical physics.

Definition 6.2. Let $q(\mathbf{x})$ be a distribution on the space of spin vectors. The **average energy** of q is

$$U(q) = \sum_{\mathbf{x}} E(\mathbf{x})q(\mathbf{x}) \tag{6.11}$$

The **Gibbs free energy** is

$$G(q) = U(q) - H(q) \tag{6.12}$$

where $H(q)$ is the entropy of q (4.1).

These terms are useful for estimating the accuracy of the approximation. Once again, we use the relative entropy (Kullback-Leibler divergence), and we calculate

$$D(q\|p_\beta) = \sum_{\mathbf{x}} q(\mathbf{x}) \log \frac{q(\mathbf{x})}{p_\beta(\mathbf{x})} \tag{6.13}$$

$$= \sum_{\mathbf{x}} q(\mathbf{x}) \log q(\mathbf{x}) - \sum_{\mathbf{x}} q(\mathbf{x}) \log p_\beta(\mathbf{x}) \tag{6.14}$$

$$= -H(q) - \sum_{\mathbf{x}} q(\mathbf{x}) \log \left(\frac{1}{Z} e^{-\beta E(\mathbf{x})} \right) \tag{6.15}$$

$$= -H(q) + \sum_{\mathbf{x}} q(\mathbf{x}) \log Z + \sum_{\mathbf{x}} q(\mathbf{x})\beta E(\mathbf{x}) \tag{6.16}$$

$$= -H(q) + \log Z + \beta U(q) \tag{6.17}$$

Now, the unknown distribution p_β is only contained in the term $\log Z$, which is just a constant.

For $\beta = 1$, we have

$$D(q\|p_{\beta=1}) = G(q) + \log Z \tag{6.18}$$

For $q = p$, $D(q\|p) = 0$, and $G(q)$ achieves its minimum $-\log Z$. So, minimizing $G(q)$ is a variant of minimizing the relative entropy to the Boltzmann distribution p_β.

To the best of our knowledge, so far in the field of belief propagation and free energy the role of β has not yet been recognized. Either β is set to 1 and further on disregarded [YFW04, AM01] or not mentioned at all [MY02, HMP03]. This is fair enough from their point of view, since their focus is not optimization (maximization of the optimum's probability). For us, the choice of β is an important tool to influence the algorithm's behavior.

Therefore, it is necessary to generalize the notions for arbitrary β. We call these generalizations *tempered*, because β plays the role of an inverse temperature. The conventional versions known from literature are always the special case for $\beta = 1$.

Definition 6.3. The **tempered average energy** $U_\beta(q)$ of a distribution q is

$$U_\beta(q) := \beta U(q) = \beta \sum_{\mathbf{x}} E(\mathbf{x}) q(\mathbf{x}) \tag{6.19}$$

The **tempered Gibbs free energy** is

$$G_\beta(q) := U_\beta(q) - H(q) \tag{6.20}$$

Lemma 6.1. *The tempered Gibbs free energy of a distribution q is equivalent to the relative entropy between q and the Boltzmann distribution:*

$$D(q\|p_\beta) = G_\beta(q) + \log Z \tag{6.21}$$

Proof. Follows directly from (6.17). □

We see that minimizing the tempered Gibbs free energy is equivalent to minimizing the relative entropy to the Boltzmann distribution.

In Sect. 4.1.3, we said that the relative entropy $D(p\|q)$ measures the inefficiency of assuming a distribution q when the real distribution is p. If we compare this to (6.21), we see that the relative entropy is reversed: Here we minimize $D(q\|p_\beta)$, where p_β is the true Boltzmann distribution and q our approximation. The relative entropy is not symmetric, so these are different objectives. But anyway, the minimum for both is achieved when the two distributions coincide.

6.1.5. Mean-Field Approximation

To turn this approximation into a feasible algorithm, the search is restricted on a special class of distributions q. The simplest example is, like in EDA, the case of the independent product distribution:

$$q(\mathbf{x}) = \prod_i q(x_i) \tag{6.22}$$

The product distribution is determined by a vector of univariate probabilities, which we call q_i:

$$q_i := q(X_i = 1) \tag{6.23}$$

For this class the Gibbs free energy can be computed directly. It consists of the average energy and the entropy. The entropy is

$$H(q) = \sum_i H(q_i) \tag{6.24}$$

$$= -\sum_i \left(q_i \log q_i + (1 - q_i) \log(1 - q_i)\right) \tag{6.25}$$

For the average energy, we assume $E(\mathbf{x})$ to be additively decomposable, so similar to (2.22) we have a decomposition $\mathfrak{S}$ and

$$E(\mathbf{x}) = -\sum_{i=1}^{m} f_i(\mathbf{x}_{s_i}) \tag{6.26}$$

(The minus sign is included because we want to maximize f whereas E is traditionally minimized.)

This is the case for the Ising model, as can be seen in (6.2) or (6.3). In this case we have

$$U(q) = -\sum_{i=1}^{m} \sum_{\mathbf{x}_{s_i}} q(\mathbf{x}_{s_i}) f_i(\mathbf{x}_{s_i}) \tag{6.27}$$

$$= -\sum_{i=1}^{m} \sum_{\mathbf{x}_{s_i}} f_i(\mathbf{x}_{s_i}) \prod_{k \in s_i} q(x_k) \tag{6.28}$$

For the special case of the Ising model, this becomes

$$U(q) = -\sum_{1 \le \nu < \mu \le n} J_{\mu,\nu} q(s_\mu) q(s_\nu) - \sum_{\nu=1}^{n} H_\nu q(s_\nu) \tag{6.29}$$

To minimize $G(q) = U(q) - H(q)$, we can calculate the derivative with respect to q_i and get

$$\frac{\partial G(q)}{\partial q_i} = \frac{\partial U(q)}{\partial q_i} + \ln \frac{q_i}{1 - q_i} \tag{6.30}$$

Setting this equal to zero and solving for q_i yields

$$q_i = \frac{1}{1 + \exp\left(-\frac{\partial U(q)}{\partial q_i}\right)} \quad . \tag{6.31}$$

A solution can be found by iterating these equations.

Our goal is to minimize $G(q)$ for more complicated models than the simple product distribution. We now use it for a cyclic graphical model which can be understood as a generalization of the junction tree, the *region graph*.

6.2. The Region Graph

The region graph [YFW01, YFW02, YFW04] is a loopy graphical model. It is strongly related to partially ordered sets (posets) or Hasse diagrams [Sta86]. Similar or identical structures have been presented in [AM01, MY02, TW03]. This section follows roughly the notation of [YFW04].

6.2.1. Regions

Our starting point is an additively decomposable fitness function f. We recall the definitions for ADFs from Sect. 2.4.4. In [YFW04] another graphical model is used, the *factor graph*. The factor graph is identical to an additive decomposition. Therefore we omit this structure and use the ADF notation instead.

Definition 6.4. Let $\mathfrak{S} = \{s_1, \ldots, s_m\}$ be an additive decomposition for a fitness function f, such that

$$f(\mathbf{x}) = \sum_{s_i \in \mathfrak{S}} f_i(\mathbf{x}_{s_i}) \tag{6.32}$$

A **region** $R = (\mathbf{X}_R, F_R)$ is a set of variables $\mathbf{X}_R \subseteq \{X_1, \ldots, X_n\}$ and a set of subfunctions $F_R \subseteq \{f_1, \ldots, f_m\}$, such that

$$\forall f_i \in F_R : \mathbf{X}_{s_i} \subseteq \mathbf{X}_R \tag{6.33}$$

It is asserted by (6.33) that all variables needed for the contained subfunctions are in $\mathbf{X}_R$. F_R can also be empty.

Example 6.4. For an example for a region, consider the 3×3 grid in Fig. 6.3. We might want to introduce a region R for the top left 2×2 square, having $\mathbf{X}_R = \{X_1, X_2, X_4, X_5\}$. Assuming a top-left to bottom-right numbering for the bivariate subfunctions of f, we set $F_R = \{f_1, f_3, f_4, f_6\}$, in order to include $f_1(x_1, x_2)$, $f_3(x_1, x_4)$, $f_4(x_2, x_5)$, and $f_6(x_4, x_5)$.

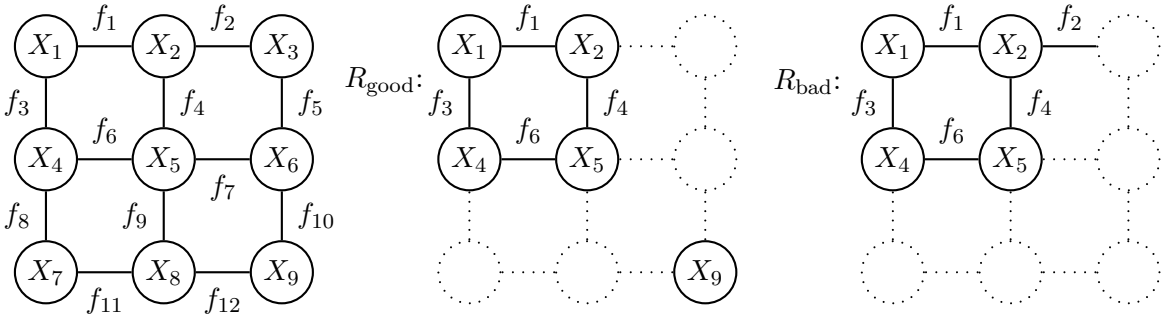

Figure 6.3.: An example 3×3 grid and two example regions. R_{good} is a valid region; R_{bad} is invalid because the subfunction f_2 depends on X_3 which is not in $\mathbf{X}_{R_{\text{bad}}}$.

The definition also allows to include variables which do not appear in any subfunction in F_R, e. g. setting $\mathbf{X}_R = \{X_1, X_2, X_4, X_5, X_9\}$ (the region R_{good} in Fig. 6.3). It only forbids a region like R_{bad} in Fig. 6.3 with $R_{\text{bad}} = (X_{R_{\text{bad}}} = \{X_1, X_2, X_4, X_5\}, F_{R_{\text{bad}}} = \{f_1, f_2, f_3, f_4, f_6\})$, because $f_2(x_2, x_3)$ depends on X_3, but $X_{R_{\text{bad}}}$ does not contain X_3.

We keep in mind our goal to approximate the Boltzmann distribution (6.6) with the energy $E(\mathbf{x}) = -f(\mathbf{x})$. The minus sign is necessary because f is maximized and E is minimized.

For a region, we can define a local energy analogous to (6.2).

Definition 6.5. For a region R, define the **region energy**

$$E_R(\mathbf{x}_R) := -\sum_{f_i \in F_R} f_i(\mathbf{x}_{s_i}) \tag{6.34}$$

This is well-defined because of the condition (6.33).

Similar to the mean-field approach, we define a local approximation of the objective Boltzmann distribution on a region, q_R. In [YFW04] this local distribution is called the *belief* on R. We introduce the afore-mentioned notions of statistical physics for regions by applying them on the distribution q_R.

Definition 6.6. The **region tempered average energy** $U_{\beta,R}(q_R)$, the **region entropy** $H_R(q_R)$ and the **region tempered free energy** $F_{\beta,R}(q_R)$ are defined as

$$U_{\beta,R}(q_R) := \beta \sum_{\mathbf{x}_R} q_R(\mathbf{x}_R) E_R(\mathbf{x}_R) \tag{6.35}$$

$$H_R(q_R) := -\sum_{\mathbf{x}_R} q_R(\mathbf{x}_R) \log q_R(\mathbf{x}_R) \tag{6.36}$$

$$F_{\beta,R}(q_R) := U_{\beta,R}(q_R) - H_R(q_R) \tag{6.37}$$

The conventional (non-tempered) versions are the special case for $\beta = 1$.

6.2.2. Region Graph

Definition 6.7. A **region graph** is a graph $G = (\mathcal{R}, E_{\mathcal{R}})$, where $\mathcal{R}$ is a set of regions and $E_{\mathcal{R}}$ is a set of directed edges. An edge $(R_p, R_c) \in E_{\mathcal{R}}$ is only allowed if $\mathbf{X}_{R_c} \subset \mathbf{X}_{R_p}$. If $(R_p, R_c) \in E_{\mathcal{R}}$, we call R_p a parent of R_c and R_c child of R_p.

Since $E_{\mathcal{R}}$ imposes a partial ordering on the set of regions, in [MY02] the same structure was called a partially ordered set or *poset*.

Lemma 6.2. *A region graph is directed acyclic.*

Proof. This follows immediately from the requirement that edges are only allowed from supersets to subsets. □

A junction tree can be turned into a region graph by creating a region for every cluster and every separator and adding edges from each node to each neighboring separator.

We recall from (3.33) that the global distribution of a junction tree is the product of all distributions on the clusters, divided by the distributions of all the separators. We generalize this concept too, by introducing counting numbers of the regions.

Definition 6.8. The **counting number** c_R of a region R is defined recursively as

$$c_R = 1 - \sum_{R' \in A(R)} c_{R'} \tag{6.38}$$

where $A(R)$ is the set of all ancestors of R, as in Def. 3.4.

This is well-defined, because the region graph is cycle-free. The maximal regions (without ancestors) have counting number 1. From there, the counting numbers can be calculated from the top to the bottom of the graph. Fig. 6.4 shows an example region graph and the counting numbers of its regions.

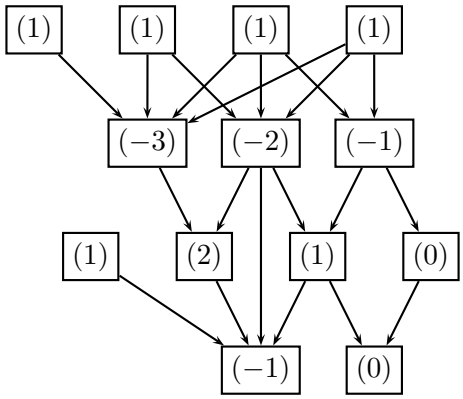

Figure 6.4.: Example region graph. Each box represents one region, and the arrows give the edges in $E_{\mathcal{R}}$. The numbers in parentheses are the counting numbers of the regions.

For a junction tree, all clusters have counting number 1, and all separators have counting number -1, because their ancestors are the two clusters that they connect.

For a junction tree, the local marginal distributions were combined to a global distribution (3.33). Formally, this can be generalized in the following way:

Definition 6.9. The **Kikuchi approximation** of a probability distribution for a region graph is

$$k(\mathbf{x}) = \prod_{R \in \mathcal{R}} q_R(\mathbf{x}_R)^{c_R} \tag{6.39}$$

In general, it is not normalized and therefore no probability distribution. The **normalized Kikuchi approximation**

$$p_k(\mathbf{x}) = \frac{k(\mathbf{x})}{\sum_{\mathbf{y}} k(\mathbf{y})} \tag{6.40}$$

is a probability distribution.

The computation of $p_k(\mathbf{x})$ is in general exponential.

6.2.3. Region Graph and Junction Tree

If the region graph is derived from a junction tree, with the q_R being the local distributions on the clusters and separators, $k(\mathbf{x})$ is a valid distribution, since its definition coincides with the junction tree distribution (3.33).

The junction property also has an equivalent on the region graph.

Definition 6.10. We call a region graph **valid** if it fulfills the **region graph condition**, which states that

1. For all variables $X_i \in \{X_1, \ldots, X_n\}$ the set $\mathcal{R}_{X_i} := \{R \in \mathcal{R} | X_i \in \mathbf{X}_R\}$ of all regions R that contain X_i form a connected subgraph with

$$\sum_{R \in \mathcal{R}_{X_i}} c_R = 1 \ , \tag{6.41}$$

   and

2. For all subfunctions $f_i \in \{f_1, \ldots, f_m\}$ the set $\mathcal{R}_{f_i} := \{R \in \mathcal{R} | f_i \in F_R\}$ of all regions R that contain f_i form a connected subgraph with

$$\sum_{R \in \mathcal{R}_{f_i}} c_R = 1 \ . \tag{6.42}$$

The connectivity of the subgraph, like the junction property, prevents that in different parts of the graph contradictory beliefs evolve. The condition on the counting numbers makes sure that every variable and every subfunction is counted exactly once.

In a junction tree it is often the case that several separators contain the same variables. In [JJ94] it was proposed to replace these by a single separator that is connected to all clusters in which it is contained. Then care must be taken that the local distribution p_S of such a separator S is counted the appropriate number of times. This can also be done in the region graph. The definition of counting numbers and the Kikuchi approximation ensure that the distribution is divided by p_S the appropriate number of times.

Furthermore, [JJ94] proposes separators not only between clusters, but also between other separators. They call the resulting tree an *Almond tree*. This is another step of generalizing the junction tree towards the region graph, and it is straightforward to do this with our region graph definition, too.

In fact, it has been proven in [PA05] that cycle-free region graphs give exact results. In the following an adaptation of the proof for our notation is presented. For this we prove some lemmata.

Lemma 6.3. *In a cycle-free region graph, the counting numbers are*

$$c_R = 1 - |\Pi_R| \tag{6.43}$$

where $|\Pi_R|$ is the number of parents of the region R.

Proof. The proof exploits the fact that the sets of ancestors for all parents of a node R are disjunct.

$$\begin{aligned} c_R &= 1 - \sum_{R' \in A(R)} c_{R'} && (6.44)\\ &= 1 - \sum_{R' \in \Pi_R} \left(c_{R'} + \sum_{R'' \in A(R')} c_{R''} \right) && (6.45)\\ &= 1 - \sum_{R' \in \Pi_R} \left(1 - \sum_{R'' \in A(R')} c_{R''} + \sum_{R'' \in A(R')} c_{R''} \right) && (6.46)\\ &= 1 - \sum_{R' \in \Pi_R} 1 && (6.47)\\ &= 1 - |\Pi_R| && (6.48) \end{aligned}$$

□

Lemma 6.4. *Cycle-free region graphs originating from junction trees are valid.*

Proof. It follows immediately from the junction property that the subgraph $\mathcal{R}_{X_i}$ of the region graph that contains a variable X_i is connected. For every region R that the subgraph contains, it contains also all its parents Π_R. Since it is cycle-free, it is also a tree.

We only have to prove (6.41) (the sum of the counting numbers is 1). We use Lemma 6.3:

$$\begin{aligned} \sum_{R \in \mathcal{R}_{X_i}} c_R &= \sum_{R \in \mathcal{R}_{X_i}} 1 - |\Pi_R| && (6.49)\\ &= |\mathcal{R}_{X_i}| - \sum_{R \in \mathcal{R}_{X_i}} |\Pi_R| && (6.50) \end{aligned}$$

This is equal to the number of nodes in the tree $\mathcal{R}_{X_i}$ minus the number of edges in the tree, and it is an obvious property of trees that this difference is 1.

The second part with the subfunctions can be proven analogously. □

Theorem 6.5. *For a valid region graph without cycles, the Kikuchi approximation is exact.*

Proof. We prove that the Kikuchi approximation (6.39) gives exact marginals on the regions:

$$k(\mathbf{x}_R) = q_R(\mathbf{x}_R) \tag{6.51}$$

The proof works similar to the proof of Lemma 3.5. We choose leaf regions – regions that are connected to only one other region – and eliminate those from the graph, until only the region R remains.

Choose a leaf region $S \neq R$. Since the graph is cycle-free, such a leaf must exist. For the single connection of S, there exist two possibilities:

- S is the child of another region. But since it has only one parent, from Lemma 6.3 follows that it has counting number $c_S = 0$. So, its local belief $q_S(\mathbf{x}_S)^{c_S}$ has no effect in (6.39), and we can remove this region.
- S is parent of another region T. Remember that $\mathbf{X}_T \subset \mathbf{X}_S$. From local consistence of the beliefs we have

  $$\sum_{\mathbf{x}_S \setminus \mathbf{x}_T} q(\mathbf{x}_S) = q(\mathbf{x}_T) \tag{6.52}$$

  From this, it follows that

  $$q(\mathbf{x}_T)^{c_T} \sum_{\mathbf{x}_S \setminus \mathbf{x}_T} q(\mathbf{x}_S) = q(\mathbf{x}_T)^{c_T+1} \tag{6.53}$$

  Again, we see that it has no effect on (6.39) to eliminate the region S and increase the counting number of T by one, since it has then one parent less.

□

For cycle-free region graphs derived from a junction tree, the Kikuchi approximation is equal to the junction tree distribution. But in general, k is not even a distribution. And the normalized Kikuchi approximation p_k cannot be computed in polynomial time. What we wish for is an approximative distribution which can be computed polynomially and from which points can be sampled.

In Sect. 3.6.1, the connection of junction trees and factorizations was explained. Our goal is to create a connection between region graphs and factorizations, too. This will be achieved in Sect. 6.4. Now, we first present the generalized belief propagation algorithm which provides us with the marginal distributions q_R.

6.3. Generalized Belief Propagation

6.3.1. Free Energy of a Region Graph

We recall the objective of Sect. 6.1.4: The Boltzmann distribution p_β is approximated by a distribution q with minimal free energy. But instead of a distribution $q(\mathbf{x})$, we have only a collection of region beliefs $q_\mathcal{R} = \{q_R(\mathbf{x}_R) | R \in \mathcal{R}\}$. Therefore the free energy of a region graph is composed of the region free energies (see Def. 6.3).

Definition 6.11. The **region-based approximate entropy** for a region graph $\mathcal{R}$ with the set of local beliefs $q_\mathcal{R} = \{q_R | R \in \mathcal{R}\}$ is

$$H_\mathcal{R}(q_\mathcal{R}) := \sum_{R \in \mathcal{R}} c_R H_R(q_R) \tag{6.54}$$

The **region-based tempered average energy** is

$$U_{\beta,\mathcal{R}}(q_\mathcal{R}) := \sum_{R \in \mathcal{R}} c_R U_{\beta,R}(q_R) \tag{6.55}$$

Finally, the **region-based tempered Gibbs free energy** is

$$F_{\beta,\mathcal{R}}(q_\mathcal{R}) = U_{\beta,\mathcal{R}}(q_\mathcal{R}) - H_\mathcal{R}(q_\mathcal{R}) \tag{6.56}$$

How are these definitions justified? In general, $U_{\beta,\mathcal{R}}(q_\mathcal{R})$ and $H_\mathcal{R}(q_\mathcal{R})$ are not the average energy or the entropy of any distribution q. For $U_{\beta,\mathcal{R}}(q_\mathcal{R})$ the following lemma shows that the desired solution has the correct average energy.

Lemma 6.6. *If all local beliefs of a valid region graph are the correct marginals of the Boltzmann distribution, the region-based tempered average energy $U_{\beta,\mathcal{R}}(q_\mathcal{R})$ is the exact average energy of the Boltzmann distribution.*

Proof. The region-based tempered average energy is calculated using (6.34), (6.35), and (6.55):

$$U_{\beta,\mathcal{R}}(q_\mathcal{R}) = -\beta \sum_{R \in \mathcal{R}} c_R \sum_{\mathbf{x}_R} q_R(\mathbf{x}_R) \sum_{f_i \in F_R} f_i(\mathbf{x}_{s_i}) \tag{6.57}$$

$$= -\beta \sum_{R \in \mathcal{R}} c_R \sum_{\mathbf{x}_R} \sum_{f_i \in F_R} q_R(\mathbf{x}_R) f_i(\mathbf{x}_{s_i}) \tag{6.58}$$

$$= -\beta \sum_{R \in \mathcal{R}} c_R \sum_{f_i \in F_R} \sum_{\mathbf{x}_{s_i}} f_i(\mathbf{x}_{s_i}) \sum_{\mathbf{x}_R \setminus \mathbf{x}_{s_i}} q_R(\mathbf{x}_R) \tag{6.59}$$

Since we assumed the local beliefs to be the Boltzmann marginals, we insert

$$U_{\beta,\mathcal{R}}(q_\mathcal{R}) = -\beta \sum_{R \in \mathcal{R}} c_R \sum_{f_i \in F_R} \sum_{\mathbf{x}_{s_i}} f_i(\mathbf{x}_{s_i}) p_\beta(\mathbf{x}_{s_i}) \tag{6.60}$$

Rearranging the summands leads to

$$U_{\beta,\mathcal{R}}(q_\mathcal{R}) = -\beta \sum_{i=1}^{m} \sum_{\mathbf{x}_{s_i}} f_i(\mathbf{x}_{s_i}) p_\beta(\mathbf{x}_{s_i}) \sum_{R \in \mathcal{R}_{f_i}} c_R \tag{6.61}$$

$$= -\beta \sum_{i=1}^{m} \sum_{\mathbf{x}_{s_i}} f_i(\mathbf{x}_{s_i}) p_\beta(\mathbf{x}_{s_i}) \tag{6.62}$$

because of (6.42).

This is equal to (6.27), the tempered average energy of the Boltzmann distribution. □

Unfortunately for the region-based approximate entropy $H_{\mathcal{R}}(q_{\mathcal{R}})$, the analogous statement is generally not true. In [YFW04] there is given an example where the minimum of $F_{\beta,\mathcal{R}}(q_{\mathcal{R}})$ is reached by a set of beliefs which is *contradictory*, i. e. there is no global distribution of which they are the marginals. Furthermore, they give a case in which $H_{\mathcal{R}}(q_{\mathcal{R}})$ is negative, which is of course impossible for an entropy.

Certainly the region-based free energy is an approximation. Its accuracy depends strongly on the structure of the region graph. If the region graph is cycle-free, it is the exact free energy of the Kikuchi distribution (which is a distribution in this case, see Theorem 6.5).

Lemma 6.7. *For a cycle-free region graph, the region-based tempered Gibbs free energy is the tempered free energy of the Kikuchi distribution $k(\mathbf{x})$.*

Proof. We recall (6.51) from Theorem 6.5.

$$F_{\beta,\mathcal{R}}(q_{\mathcal{R}}) = \beta \sum_{R\in\mathcal{R}} c_R \sum_{\mathbf{x}_R} q_R(\mathbf{x}_R) E_R(\mathbf{x}_R) + \sum_{R\in\mathcal{R}} c_R \sum_{\mathbf{x}_R} q_R(\mathbf{x}_R) \log q_R(\mathbf{x}_R) \tag{6.63}$$

$$= \sum_{R\in\mathcal{R}} c_R \sum_{\mathbf{x}_R} q_R(\mathbf{x}_R) \left(\beta E_R(\mathbf{x}_R) + \log q_R(\mathbf{x}_R)\right) \tag{6.64}$$

$$= \sum_{R\in\mathcal{R}} c_R \sum_{\mathbf{x}_R} \left(\beta E_R(\mathbf{x}_R) + \log q_R(\mathbf{x}_R)\right) \sum_{\mathbf{x}\setminus\mathbf{x}_R} k(\mathbf{x}) \tag{6.65}$$

$$= \sum_{\mathbf{x}} k(\mathbf{x}) \sum_{R\in\mathcal{R}} c_R \left(\beta \sum_{f_i\in F_R} -f_i(\mathbf{x}_{s_i}) + \log q_R(\mathbf{x}_R)\right) \tag{6.66}$$

$$= \sum_{\mathbf{x}} k(\mathbf{x}) \left(-\beta \sum_{i=1}^{m} f_i(\mathbf{x}_{s_i}) \sum_{R\in\mathcal{R}_{f_i}} c_R + \sum_{R\in\mathcal{R}} \log q_R(\mathbf{x}_R)^{c_R}\right) \tag{6.67}$$

$$= \sum_{\mathbf{x}} k(\mathbf{x}) \left(-\beta \sum_{i=1}^{m} f_i(\mathbf{x}_{s_i}) + \log \prod_{R\in\mathcal{R}} q_R(\mathbf{x}_R)^{c_R}\right) \tag{6.68}$$

$$= \sum_{\mathbf{x}} k(\mathbf{x}) \left(\beta E(\mathbf{x}) + \log k(\mathbf{x})\right) \tag{6.69}$$

$$= \sum_{\mathbf{x}} k(\mathbf{x}) \beta E(\mathbf{x}) + \sum_{\mathbf{x}} k(\mathbf{x}) \log k(\mathbf{x}) \tag{6.70}$$

$$= U_\beta(k) - H(k) \tag{6.71}$$

□

From this lemma follows that for a cycle-free region graph this method is equivalent to the junction tree method.

Remark 6.1. Another example is the *Bethe approximation*. This is the special case in which the region graph consists of two layers:

- The large regions $\mathcal{R}_L$, containing one region for each subfunction f_i: $R_{f_i} = (\mathbf{X}_{s_i}, \{f_i\})$, and

- the small regions $\mathcal{R}_S$, containing one region for each variable X_i:$R_{X_i} = (\{X_i\}, \emptyset)$.

The edges of this region graph are

$$E_{\mathcal{R}} = \{(R_{f_i}, R_{X_j}) | X_j \in \mathbf{X}_{s_i}\} \ . \tag{6.72}$$

The Bethe approximation is the first straightforward idea. It is a valid region graph. But it is of limited expressive power. In Sect. 6.4.2 we present the *cluster variation method* which constructs more powerful region graphs.

Now we calculate the minima of the free energy. This leads to fixed point equations giving rise to the generalized belief propagation algorithm.

6.3.2. Preliminary Requirements for Minimizing the Free Energy

There are several algorithms which calculate local beliefs minimizing the region-based free energy. In principle, these algorithms are equivalent since they all minimize the same objective, but their dynamics can be very different; e. g. some may converge where others do not.

In the following we derive the *parent-to-child* algorithm [YFW04]. It is favorable in that it uses only one kind of messages, from larger regions to their children (subsets). There are also algorithms that use messages in both ways [Hes03, YFW04].

Before deriving the algorithm, we make three remarks.

No Zero Counting Numbers

In the following we will need to divide by c_R, therefore we exclude the case that there is a region R with $c_R = 0$. It is no problem for the algorithm – the update equation (6.102) does not even contain the counting numbers –, but this restriction spares us some special case considerations. In fact, it is not a restriction at all, because such a regions has no effect in the equations. In [PA05] it is shown that a region with $c_R = 0$ can be removed when all its parents are connected with all its children.

Ancestor and Descendent Sets

We recall Definitions 3.4 for a directed graph: $A(R)$ are the ancestors of a region, $D(R)$ the descendents of a region. We define further on:

Definition 6.12. $A'(R) := \{R\} \cup A(R)$ and $D'(R) := \{R\} \cup D(R)$.

Local Consistence Conditions

The algorithm has some similarities with the junction tree algorithms in that is sends messages around in order to achieve local consistence:

$$\sum_{\mathbf{x}_P \setminus \mathbf{x}_R} q_P(\mathbf{x}_P) = q_R(\mathbf{x}_R) \tag{6.73}$$

In addition to (6.73), another condition for local consistence will be needed, stated by the following lemma:

Lemma 6.8. *In a region graph with no counting number equal to zero, the local region beliefs are consistent if and only if for all* $(P, R) \in E_{\mathcal{R}}$

$$c_R q_R(\mathbf{x}_R) = - \sum_{T \in A(R) \setminus A'(P)} c_T \sum_{\mathbf{x}_T \setminus \mathbf{x}_R} q_T(\mathbf{x}_T) \tag{6.74}$$

Proof. We start with the calculation

$$\begin{aligned} c_R &= 1 - \sum_{T \in A(R)} c_T && (6.75)\\ &= 1 - c_P - \sum_{T \in A(P)} c_T - \sum_{T \in A(R) \setminus A'(P)} c_T && (6.76)\\ &= - \sum_{T \in A(R) \setminus A'(P)} c_T && (6.77) \end{aligned}$$

"⇒": If all regions are consistent, we know that for all ancestors T of R

$$q_R(\mathbf{x}_R) = \sum_{\mathbf{x}_T \setminus \mathbf{x}_R} q_T(\mathbf{x}_T) \tag{6.78}$$

Therefore, multiplying this with (6.77) gives the desired equation.

"⇐": Suppose (6.74) holds. We show consistence by induction over all nodes, from top to bottom of the region graph. Since region graphs are directed acyclic, the regions can be ordered in a way such that there are no backward edges.

- Induction start: For the first region, there are no edges so consistence is trivial.
- Induction step: Assume that all ancestor regions of R are consistent with each other. That means that all marginals of these sets are the same:

$$\forall T \in A(R) : \sum_{\mathbf{x}_T \setminus \mathbf{x}_R} q_T(\mathbf{x}_T) = q_{\text{same}}(\mathbf{x}_R) \tag{6.79}$$

This is inserted into (6.74):

$$c_R q_R(\mathbf{x}_R) = -q_{\text{same}}(\mathbf{x}_R) \sum_{T \in A(R) \setminus A'(P)} c_T \tag{6.80}$$

Inserting (6.77) and dividing by c_R (remember that $c_R \neq 0$ was assumed) shows that $q_R(\mathbf{x}_R) = q_{\text{same}}(\mathbf{x}_R)$.

□

6.3.3. Free Energy Minimization

Now we will minimize the tempered free energy (see Definition 6.11). We recall that this is an approximation of minimizing the relative entropy to the Boltzmann distribution.

$$F_{\beta,\mathcal{R}}(q_{\mathcal{R}}) = \sum_{R\in\mathcal{R}} c_R F_{\beta,R}(q_R) \tag{6.81}$$

$$= \beta \sum_{R\in\mathcal{R}} c_R \sum_{\mathbf{x}_R} q_R(\mathbf{x}_R) E_R(\mathbf{x}_R) + \sum_{R\in\mathcal{R}} c_R \sum_{\mathbf{x}_R} q_R(\mathbf{x}_R) \log q_R(\mathbf{x}_R) \tag{6.82}$$

$$= \sum_{R\in\mathcal{R}} c_R \sum_{\mathbf{x}_R} q_R(\mathbf{x}_R) \left(\beta E_R(\mathbf{x}_R) + \log q_R(\mathbf{x}_R)\right) \tag{6.83}$$

The minimization is subject to the constraints that

- the beliefs are locally consistent between neighboring regions, and
- the beliefs are normalized.

The constraints are enforced by the Lagrange multiplier method. For this, new variables are added:

- γ_R, enforcing the normalization of region R

$$\sum_{\mathbf{x}_R} q_R(\mathbf{x}_R) = 1 \ , \tag{6.84}$$

- and $\lambda_{PR}(\mathbf{x}_R)$, enforcing consistence. However, as condition for consistence, (6.74) is used rather than (6.73):

$$c_R q_R(\mathbf{x}_R) + \sum_{T\in A(R)\setminus A'(P)} c_T \sum_{\mathbf{x}_T\setminus\mathbf{x}_R} q_T(\mathbf{x}_T) = 0 \tag{6.85}$$

Using these, the function to be minimized is changed to

$$L = F_{\beta,\mathcal{R}}(q_{\mathcal{R}}) + \sum_{R\in\mathcal{R}} \gamma_R \left(1 - \sum_{\mathbf{x}_R} q_R(\mathbf{x}_R)\right) + \sum_{(P,R)\in E_{\mathcal{R}}} \sum_{\mathbf{x}_R} \lambda_{PR}(\mathbf{x}_R) \left(c_R q_R(\mathbf{x}_R) + \sum_{T\in A(R)\setminus A'(P)} c_T \sum_{\mathbf{x}_T\setminus\mathbf{x}_R} q_T(\mathbf{x}_T)\right) \tag{6.86}$$

Deriving L with respect to the γ or λ variables and setting the result equal to zero gives exactly the equations for the constraints, (6.84) and (6.85). Deriving L with respect

to a $q_R(\mathbf{x}_R)$ results in

$$\frac{\partial L}{\partial q_R(\mathbf{x}_R)} = c_R\beta E_R(\mathbf{x}_R) + c_R(1+\log q_R(\mathbf{x}_R)) - \gamma_R + \sum_{P\in\Pi_R} \lambda_{PR}(\mathbf{x}_R)c_R + \sum_{\substack{(P,C)\in E_{\mathcal{R}} \\ R\in A(C)\setminus A'(P)}} \lambda_{PC}(\mathbf{x}_C)c_R \tag{6.87}$$

$$= c_R\beta E_R(\mathbf{x}_R) + c_R(1+\log q_R(\mathbf{x}_R)) - \gamma_R + \sum_{(P,C)\in M(R)} \lambda_{PC}(\mathbf{x}_C)c_R \tag{6.88}$$

Here $M(R)$ is the set of all relevant messages for the region R.

$$M(R) := \{(P,C)\in E_{\mathcal{R}} : C\in D'(R) \wedge P\notin D'(R)\} \tag{6.89}$$

Setting this partial derivative equal to zero and resolving for $q_R(\mathbf{x}_R)$ gives:

$$q_R(\mathbf{x}_R) = e^{-\beta E_R(\mathbf{x}_R)} e^{\frac{\gamma_R}{c_R}-1} \prod_{(P,C)\in M(R)} e^{-\lambda_{PC}(\mathbf{x}_C)} \tag{6.90}$$

For normalization, we set

$$e^{\frac{\gamma_R}{c_R}-1} = \left(\sum_{\mathbf{x}_R} e^{-\beta E_R(\mathbf{x}_R)} \prod_{(P,C)\in M(R)} e^{-\lambda_{PC}(\mathbf{x}_C)}\right)^{-1} \tag{6.91}$$

The variables $\lambda_{PC}(\mathbf{x}_C)$ ensure local consistence between the nodes. For this, an iterative message passing algorithm is derived. We define the messages between the nodes as

$$m_{P\to C}(\mathbf{x}_C) := e^{-\lambda_{PC}(\mathbf{x}_C)} \tag{6.92}$$

With these definitions, we obtain the following solution for the local belief.

Definition 6.13. The local beliefs in the parent to child-algorithm are

$$q_R(\mathbf{x}_R) \propto \prod_{f_i\in F_R} e^{\beta f_i(\mathbf{x}_{s_i})} \prod_{(P,C)\in M(R)} m_{P\to C}(\mathbf{x}_C) \tag{6.93}$$

So, the included messages are the ones that enter into R or one of its descendents from a non-descendent of R. Remember that the variables contained in a descendent C of R are a subset of those in R, so $\mathbf{x}_C$ is well-defined. The motivation of this definition is that all information about the variables in R or a subset thereof, coming from an exterior region P, should be included.

Note also the $\propto$ sign in (6.93). To compute the belief exactly, it must be normalized to sum up to one. This means as well that multiplying a message by a constant factor (e. g. normalizing a message) has no effect.

Inserting the definitions of the beliefs (6.93) into (6.73) gives

$$\sum_{\mathbf{x}_P\setminus\mathbf{x}_R} \prod_{f_i\in F_P} e^{\beta f_i(\mathbf{x}_{s_i})} \prod_{(I,J)\in M(P)} m_{I\to J}(\mathbf{x}_J) = \prod_{f_i\in F_R} e^{\beta f_i(\mathbf{x}_{s_i})} \prod_{(I,J)\in M(R)} m_{I\to J}(\mathbf{x}_J) \tag{6.94}$$

We know that $(P,R) \in M(R)$ and $(P,R) \notin M(P)$. So we can solve this equation for $m_{P\to R}(\mathbf{x}_R)$:

$$m_{P\to R}(\mathbf{x}_R) = \frac{\sum_{\mathbf{x}_P\setminus\mathbf{x}_R} \prod_{f_i\in F_P} e^{\beta f_i(\mathbf{x}_{s_i})} \prod_{(I,J)\in M(P)} m_{I\to J}(\mathbf{x}_J)}{\prod_{f_i\in F_R} e^{\beta f_i(\mathbf{x}_{s_i})} \prod_{(I,J)\in M(R)\setminus\{(P,R)\}} m_{I\to J}(\mathbf{x}_J)} \tag{6.95}$$

F_R is a subset of F_P, so these factors cancel out. Also, the messages in $M(P)\cap M(R)$ cancel out. After these simplifications, we get the fixed point equations for the messages:

$$m_{P\to R}(\mathbf{x}_R) = \frac{\sum_{\mathbf{x}_P\setminus\mathbf{x}_R} \prod_{f_i\in F_P\setminus F_R} e^{\beta f_i(\mathbf{x}_{s_i})} \prod_{(I,J)\in N(P,R)} m_{I\to J}(\mathbf{x}_J)}{\prod_{(I,J)\in D(P,R)} m_{I\to J}(\mathbf{x}_J)} \tag{6.96}$$

where the message sets of the nominator and denominator are

$$N(P,R) := M(P) \setminus (M(P)\cap M(R)) \tag{6.97}$$
$$D(P,R) := M(R) \setminus (M(P)\cap M(R)) \setminus \{(P,R)\} \tag{6.98}$$

More specifically, the edge sets are

$$N(P,R) = \{(I,J)\in E_{\mathcal{R}} : J\in D'(P)\setminus D'(R) \wedge I\notin D'(P)\} \tag{6.99}$$
$$D(P,R) = \{(I,J)\in E_{\mathcal{R}} : (I,J)\neq(P,R) \wedge J\in D'(R) \wedge I\in D'(P)\setminus D'(R)\} \tag{6.100}$$

The definition of the relevant edges becomes clearer with an example. In Fig. 6.5, a poset is depicted. For the message between P and R, the relevant sets of regions and edges are marked.

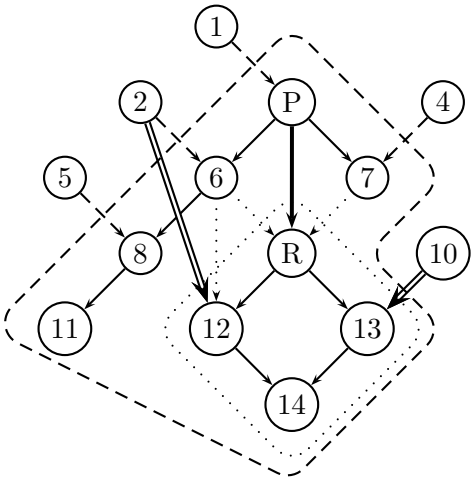

Figure 6.5.: Example region graph. Dashed region is $D'(P)$, dotted region is $D'(R)$. Dashed edges are in $N(P,R)$, dotted edges are in $D(P,R)$. Double-lined edges are in $M(P)\cap M(R)$ and cancel out.

The descendents of P, $D'(P)$, are the regions within the dashed line. The dotted line surrounds the descendents of R, $D'(R)$. The relevant messages $M(P)$ are all messages leading into the dashed region, which are the dashed and the double-lined edges. The edges in $M(R)$ are the dotted and the double-lined edges.

The double-lined edges are in both relevant sets. So they appear on both sides of (6.73) and cancel out. This leaves us with the dashed edges in $N(P,R)$ and the dotted edges in $D(P,R)$.

All in all, we have proven the following theorem.

Theorem 6.9. *The fixed points of* (6.96) *for a valid region graph with no counting number equal zero are extreme points of the region based tempered free energy.*

A proof that stable fixed points are in fact minima of the free energy is given in [Hes03], for a more restricted graphical model. In [Hes04] first steps are made to derive conditions for uniqueness of the fixed point.

6.3.4. Generalized Belief Propagation

Generalized Belief Propagation is an iteration of the fixed point equation (6.96). There are several possibilities to iterate this equation. For example, if all messages are updated *in parallel*, the iteration is

$$m_{P\to R}^{\tau,\mathrm{upd}}(\mathbf{x}_R) = \frac{\sum_{\mathbf{x}_P\setminus\mathbf{x}_R}\prod_{f_i\in F_P\setminus F_R} e^{\beta f_i(\mathbf{x}_{s_i})}\prod_{(I,J)\in N(P,R)} m_{I\to J}^{\tau-1}(\mathbf{x}_J)}{\prod_{(I,J)\in D(P,R)} m_{I\to J}^{\tau-1}(\mathbf{x}_J)} \tag{6.101}$$

where τ is the iteration index.

However, [YFW04] emphasize strongly that this iteration scheme "empirically often results in poor convergence properties". They propose a *sequential update*. The message are updated starting at the bottom of the region graph, working the way up to the top. So for an edge $(P,R)\in E_{\mathcal{R}}$ the messages of the edges $D(P,R)$ have already been updated, whereas the edges $N(P,R)$ still contain the old beliefs. This gives the following iteration:

$$m_{P\to R}^{\tau,\mathrm{upd}}(\mathbf{x}_R) = \frac{\sum_{\mathbf{x}_P\setminus\mathbf{x}_R}\prod_{f_i\in F_P\setminus F_R} e^{\beta f_i(\mathbf{x}_{s_i})}\prod_{(I,J)\in N(P,R)} m_{I\to J}^{\tau-1}(\mathbf{x}_J)}{\prod_{(I,J)\in D(P,R)} m_{I\to J}^{\tau,\mathrm{upd}}(\mathbf{x}_J)} \tag{6.102}$$

It is not always a good idea to change all the messages to $m^{\tau,\mathrm{upd}}$ now. Instead, it is numerically more favorable to choose the new messages somewhere between the old and new messages. This technique is called *damping* or *inertia*. Damping includes a *learning rate* $\alpha\in]0,1]$, where $\alpha=0$ would mean $m^\tau = m^{\tau-1}$, so no progress at all, and $\alpha=1$ means $m^\tau = m^{\tau,\mathrm{upd}}$, i. e. no damping. There are two variants of damping:

- Linear damping [YFW04, Yui02] calculates the messages as a linear combination between the old and update messages:

$$m_{P\to R}^{\tau}(\mathbf{x}_R) = (1-\alpha)m_{P\to R}^{\tau-1}(\mathbf{x}_R) + \alpha m_{P\to R}^{\tau,\mathrm{upd}}(\mathbf{x}_R) \tag{6.103}$$

- Geometrical damping [Hes03, Teh03] uses instead of the additive combination a multiplicative combination:

$$m^{\tau}_{P\to R}(\mathbf{x}_R) = m^{\tau-1}_{P\to R}(\mathbf{x}_R)^{1-\alpha} m^{\tau,\mathrm{upd}}_{P\to R}(\mathbf{x}_R)^{\alpha} \tag{6.104}$$

Since the messages are combined with each other in a multiplicative manner in (6.93), geometrical damping appears more plausible. On the other hand, linear damping is computationally cheaper because it multiplies by α instead of raising to the power of α (although this difference is almost negligible). Empirically we found that it does not matter much which technique is employed. Usually we used linear damping.

As a good value for the learning rate, [YFW04] propose 0.5. If the algorithm has trouble to converge, a smaller value can help. Note however, that this affects only the dynamics of the algorithm, not the fixed points.

For some instances, it may happen that GBP does not converge, independent of the learning rate. Alternative algorithms have been presented that are slower than GBP, but guaranteed to converge [Yui02, Hes03]. The CCCP Algorithm [Yui02] is presented in Sect. 6.6.

In the experiments, as the criterion for convergence we demand that for all messages

$$\max\left\{\frac{m^{\tau}_{P\to R}(\mathbf{x}_R)}{m^{\tau-1}_{P\to R}(\mathbf{x}_R)}, \frac{m^{\tau-1}_{P\to R}(\mathbf{x}_R)}{m^{\tau}_{P\to R}(\mathbf{x}_R)}\right\} - 1 < 10^{-6} \tag{6.105}$$

Since we are interested in optimization and not in the very small probabilities of points with small fitness, we do not need a very strict convergence criterion. Often the most important probabilities are already stable after about 50 iterations.

6.3.5. An Example in Detail

To make the derivation more clear, we present a simple example. The example is a bivariate circle with four nodes, depicted in Fig. 6.6.

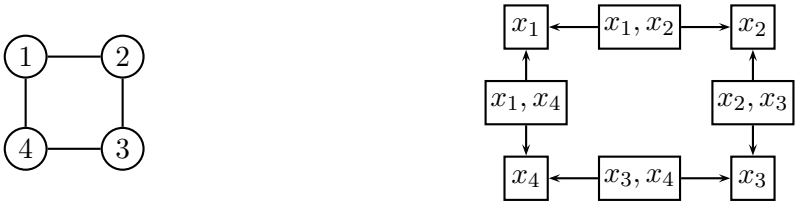

Figure 6.6.: An example 4-variate circle and its region graph

For this example, we assume the objective function to be

$$f(\mathbf{x}) = f_{12}(x_1, x_2) + f_{23}(x_2, x_3) + f_{34}(x_3, x_4) + f_{14}(x_1, x_4) \tag{6.106}$$

The region graph contains eight regions, which we will call R_{12}, R_{23}, R_{34}, R_{14}, R_1, R_2, R_3, and R_4. The bivariate regions contain one subfunction each and have counting

number +1. The univariate regions do not contain subfunctions and have counting number −1. The belief distributions are called $q_{12}(x_1, x_2)$ and so forth.

We now give the Lagrangian in detail. It consists of four parts: The average energy, the entropy, and the Lagrange multipliers for normalization and consistency:

$$L = U_{\beta,\mathcal{R}}(q_\mathcal{R}) - H_\mathcal{R}(q_\mathcal{R}) + \Gamma + \Lambda \tag{6.107}$$

First, the tempered average energy is calculated using (6.34), (6.35) and (6.55) as

$$\begin{aligned} U_{\beta,\mathcal{R}}(q_\mathcal{R}) = &-\beta \sum_{x_1,x_2} q_{12}(x_1,x_2) f_{12}(x_1,x_2) - \beta \sum_{x_2,x_3} q_{23}(x_2,x_3) f_{23}(x_2,x_3) \\ &- \beta \sum_{x_3,x_4} q_{34}(x_3,x_4) f_{34}(x_3,x_4) - \beta \sum_{x_1,x_4} q_{14}(x_1,x_4) f_{14}(x_1,x_4) \end{aligned} \tag{6.108}$$

The entropy is with (6.37) and (6.54)

$$\begin{aligned} H_\mathcal{R}(q_\mathcal{R}) = &- \sum_{x_1,x_2} q_{12}(x_1,x_2) \log q_{12}(x_1,x_2) - \sum_{x_2,x_3} q_{23}(x_2,x_3) \log q_{23}(x_2,x_3) \\ &- \sum_{x_3,x_4} q_{34}(x_3,x_4) \log q_{34}(x_3,x_4) - \sum_{x_1,x_4} q_{14}(x_1,x_4) \log q_{14}(x_1,x_4) \\ &+ \sum_{x_1} q_1(x_1) \log q_1(x_1) + \sum_{x_2} q_2(x_2) \log q_2(x_2) \\ &+ \sum_{x_3} q_3(x_3) \log q_3(x_3) + \sum_{x_4} q_4(x_4) \log q_4(x_4) \end{aligned} \tag{6.109}$$

Now follow the Lagrange multipliers. For normalization they read

$$\begin{aligned} \Gamma = &\gamma_{12} \left(1 - \sum_{x_1,x_2} q_{12}(x_1,x_2) \right) + \gamma_{23} \left(1 - \sum_{x_2,x_3} q_{23}(x_2,x_3) \right) \\ &+ \gamma_{34} \left(1 - \sum_{x_3,x_4} q_{34}(x_3,x_4) \right) + \gamma_{14} \left(1 - \sum_{x_1,x_4} q_{14}(x_1,x_4) \right) \\ &+ \gamma_1 \left(1 - \sum_{x_1} q_1(x_1) \right) + \gamma_2 \left(1 - \sum_{x_2} q_2(x_2) \right) \\ &+ \gamma_3 \left(1 - \sum_{x_3} q_3(x_3) \right) + \gamma_4 \left(1 - \sum_{x_4} q_4(x_4) \right) \end{aligned} \tag{6.110}$$

For consistency, consider the edge $(P, R) = (R_{12}, R_1)$. The set of regions considered in its Lagrange multiplier (6.85) is $A(R) \setminus A'(P) = \{R_{12}, R_{14}\} \setminus \{R_{12}\} = \{R_{14}\}$. Therefore its Lagrange multiplier reads

$$\sum_{x_1} \lambda_{R_{12},R_1}(x_1) \left(-q_1(x_1) + \sum_{x_4} q_{14}(x_1,x_4) \right) \tag{6.111}$$

The definition of the algorithm using Lemma 6.8 causes the Lagrange multiplier of (R_{12}, R_1) to ensure consistency in (R_{14}, R_1), and vice versa!

The whole consistency part of the Lagrangian reads

$$\begin{aligned}\Lambda = &\sum_{x_1} \lambda_{R_{12},R_1}(x_1) \left(\sum_{x_4} q_{14}(x_1, x_4) - q_1(x_1) \right) + \sum_{x_1} \lambda_{R_{14},R_1}(x_1) \left(\sum_{x_2} q_{12}(x_1, x_2) - q_1(x_1) \right) \\ &+ \sum_{x_2} \lambda_{R_{12},R_2}(x_2) \left(\sum_{x_3} q_{23}(x_2, x_3) - q_2(x_2) \right) + \sum_{x_2} \lambda_{R_{23},R_2}(x_2) \left(\sum_{x_1} q_{12}(x_1, x_2) - q_2(x_2) \right) \\ &+ \sum_{x_3} \lambda_{R_{23},R_3}(x_3) \left(\sum_{x_4} q_{34}(x_3, x_4) - q_3(x_3) \right) + \sum_{x_3} \lambda_{R_{34},R_3}(x_3) \left(\sum_{x_2} q_{23}(x_2, x_3) - q_3(x_3) \right) \\ &+ \sum_{x_4} \lambda_{R_{14},R_4}(x_4) \left(\sum_{x_3} q_{34}(x_3, x_4) - q_4(x_4) \right) + \sum_{x_4} \lambda_{R_{34},R_4}(x_4) \left(\sum_{x_1} q_{14}(x_1, x_4) - q_4(x_4) \right)\end{aligned} \tag{6.112}$$

Next, we derive L with respect to the beliefs. We only give the derivations with respect to $q_{12}(x_1, x_2)$ and $q_1(x_1)$; the other derivations work analogously.

$$\frac{\partial L}{\partial q_{12}(x_1, x_2)} = -\beta f_{12}(x_1, x_2) + \log q_{12}(x_1, x_2) + 1 - \gamma_{12} + \lambda_{R_{14},R_1}(x_1) + \lambda_{R_{23},R_2}(x_2) \tag{6.113}$$

$$\frac{\partial L}{\partial q_1(x_1)} = -\log q_1(x_1) - 1 - \gamma_1 - \lambda_{R_{12},R_1}(x_1) - \lambda_{R_{14},R_1}(x_1) \tag{6.114}$$

These equations are set equal to zero and solved for the beliefs:

$$q_{12}(x_1, x_2) = \exp(\beta f_{12}(x_1, x_2) + \gamma_{12} - 1 - \lambda_{R_{14},R_1}(x_1) - \lambda_{R_{23},R_2}(x_2)) \tag{6.115}$$

$$q_1(x_1) = \exp(-\gamma_1 - 1 - \lambda_{R_{12},R_1}(x_1) - \lambda_{R_{14},R_1}(x_1)) \tag{6.116}$$

The γ variables are used for normalization.

We define the messages:

$$m_{R_{12}\to R_1}(x_1) := \exp(-\lambda_{R_{12},R_1}(x_1)) \tag{6.117}$$

The other messages are defined analogously.

The messages are calculated in order to ensure consistency between the nodes, e. g.:

$$q_1(x_1) = \sum_{x_2} q_{12}(x_1, x_2) \tag{6.118}$$

Inserting (6.115) and (6.116) gives

$$\begin{aligned}&\exp(-\gamma_1 - 1) m_{R_{12}\to R_1}(x_1) m_{R_{14}\to R_1}(x_1) = \\ &\qquad \sum_{x_2} \exp(\beta f_{12}(x_1, x_2) + \gamma_{12} - 1) m_{R_{14}\to R_1}(x_1) m_{R_{23}\to R_2}(x_2)\end{aligned} \tag{6.119}$$

We can solve this equation for $m_{R_{12}\to R_1}(x_1)$, after canceling $m_{R_{14}\to R_1}(x_1)$ out:

$$m_{R_{12}\to R_1}(x_1) = e^{\gamma_{12}+\gamma_1} \sum_{x_2} e^{\beta f_{12}(x_1,x_2)} m_{R_{23}\to R_2}(x_2) \tag{6.120}$$

This is the special case of (6.96) for our example. The edge sets $N(P,R)$ and $D(P,R)$ are:

$$N(R_{12}, R_1) = \{(R_{23}, R_2)\} \tag{6.121}$$

The edge (R_{23}, R_2) is the only one which enters into a descendent of R_{12} other than R_1.

$$D(R_{12}, R_1) = \emptyset \tag{6.122}$$

There is only one other edge entering into R_1, namely (R_{14}, R_1), which cancels out in the equation.

The derivation works analogously for all edges and leads to the set of fixed point equations:

$$\begin{aligned} m_{R_{i,i+1}\to R_i}(x_i) &\propto \sum_{x_{i+1}} \exp(\beta f_{i,i+1}(x_i, x_{i+1})) m_{R_{i+1,i+2}\to R_{i+1}}(x_{i+1}) \\ m_{R_{i,i+1}\to R_{i+1}}(x_{i+1}) &\propto \sum_{x_i} \exp(\beta f_{i,i+1}(x_i, x_{i+1})) m_{R_{i-1,i}\to R_i}(x_i) \end{aligned} \tag{6.123}$$

The constants γ can be omitted. Since the beliefs will be normalized, scaling of the messages by a constant factor has no effect.

The fixed points of (6.123) are extreme points of the free energy under the given constraints of normalization and consistency.

Generalized belief propagation in this example uses the following scheme:

$$\begin{aligned} m^{\tau,\mathrm{upd}}_{R_{i,i+1}\to R_i}(x_i) &\propto \sum_{x_{i+1}} \exp(\beta f_{i,i+1}(x_i, x_{i+1})) m^{\tau-1}_{R_{i+1,i+2}\to R_{i+1}}(x_{i+1}) \\ m^{\tau,\mathrm{upd}}_{R_{i,i+1}\to R_{i+1}}(x_{i+1}) &\propto \sum_{x_i} \exp(\beta f_{i,i+1}(x_i, x_{i+1})) m^{\tau-1}_{R_{i-1,i}\to R_i}(x_i) \end{aligned} \tag{6.124}$$

It is often numerically favorable to normalize the messages.

Using geometrical damping, the update equations of the messages are

$$\begin{aligned} m^{\tau}_{R_{i,i+1}\to R_i}(x_i) &= m^{\tau-1}_{R_{i,i+1}\to R_i}(x_i)^{1-\alpha} m^{\tau,\mathrm{upd}}_{R_{i,i+1}\to R_i}(x_i)^{\alpha} \\ m^{\tau}_{R_{i,i+1}\to R_{i+1}}(x_{i+1}) &= m^{\tau-1}_{R_{i,i+1}\to R_{i+1}}(x_{i+1})^{1-\alpha} m^{\tau,\mathrm{upd}}_{R_{i,i+1}\to R_{i+1}}(x_{i+1})^{\alpha} \end{aligned} \tag{6.125}$$

6.4. Optimization Using Loopy Models

Our goal is to use the methods described so far for optimization. GBP provides us with a set of marginal distributions for the regions, all consistent with each other. But it does not give us a global distribution from which we could sample. Therefore, the new idea

is to combine this technique with the factorizations of *FDA* (see Sect. 2.4.4) in order to sample points.

This section presents the new method which combines factorization systems with GBP to devise the *BKDA* (Bethe Kikuchi Distribution Algorithm). It consists of the following basic steps:

1. Construct a factorization system from the ADF.
2. Construct a region graph from the ADF and the factorization system.
3. Perform GBP on the region graph.
4. Calculate the marginals of the factorization from the result of GBP.
5. Sample points from this distribution.

We now present these basic steps of the algorithm in detail.

In the following section, the algorithm is applied on some benchmark functions which are defined on circular dependency structures or on a 2-D grid. Therefore, we will present the results of the steps on these important examples as we go along.

6.4.1. Construct a Factorization System

The factorization system that we are going to use must fulfill the requirements of Def. 2.4, particularly (2.25). We have already presented two algorithms for achieving this:

- Alg. 2.6, which chooses a subset of the subfunctions for the factorization, or
- Alg. 5.1, which merges subfunctions in order to capture more dependencies.

The result of these algorithms on the 2-D grid was already given in Sect. 5.1.2. They are the factorizations (5.4) and (5.6), respectively.

The circular models that we consider are the bivariate circle which was already presented in Ex. 2.1 (p. 24) and in Sect. 3.4.5, and a trivariate circle. The trivariate circle is similar to the bivariate circle, only that it contains sets of three variables, which overlap by one variable. For this structure, the ADF is

$$f(\mathbf{x}) = \sum_{i=1}^{n/2} f_i(x_{2i-1}, x_{2i}, x_{2i+1}) \tag{6.126}$$

with the wraparound defined by $x_{n+1} = x_1$. This gives the additive decomposition

$$\mathfrak{S} = \{\{X_1, X_2, X_3\}, \{X_3, X_4, X_5\}, \ldots, \{X_{n-3}, X_{n-2}, X_{n-1}\}, \{X_{n-1}, X_n, X_1\}\} \ . \tag{6.127}$$

Alg. 2.6 constructs a factorization systems from these structures by leaving out one dependency. It yields for the bivariate circle the factorization

$$p(\mathbf{x}) = p(x_1, x_2) \prod_{i=3}^{n} p(x_i | x_{i-1}) \tag{6.128}$$

and for the trivariate circle

$$p(\mathbf{x}) = p(x_1, x_2, x_3) \left(\prod_{i=2}^{n/2-1} p(x_{2i}, x_{2i+1} | x_{2i-1}) \right) p(x_n | p_1, p_{n-1}) \ . \tag{6.129}$$

Both of these factorizations leave out one connection in the end: For the bivariate circle, the connection between x_1 and x_n is missing, and for the trivariate circle the connection between x_1 and x_{n-1} (because in (6.129), x_{n-1} depends only on x_{n-3} and x_{n-2}).

The subfunction merger Alg. 5.1 includes the missing connection and yields

$$p(\mathbf{x}) = p(x_1, x_2) p(x_3 | x_2) \dots p(x_n | x_1, x_{n-1}) \ . \tag{6.130}$$

for the bivariate circle and

$$p(\mathbf{x}) = p(x_1, x_2, x_3) \prod_{i=2}^{n/2-2} p(x_{2i}, x_{2i+1} | x_{2i-1}) p(x_{n-2}, x_{n-1} | x_1, x_{n-3}) p(x_n | p_1, p_{n-1}) \tag{6.131}$$

for the trivariate circle.

6.4.2. Construct a Region Graph: The Cluster Variation Method

Given a factorization system $\mathfrak{S}$, the task of building a region graph is not easy. Many designs are possible, with different free energies, and different quality of approximation.

The *CVM* (cluster variation method) is a method to build a region graph (or poset), given a set of maximal regions. It was introduced by Kikuchi [Kik51] and developed in the field of statistical physics. It was adopted for GBP in [YFW02, MY02, TW03]. Given a set of maximal regions (like the sets in $\mathfrak{S}$), it adds the maximal intersections of these regions to the region graph, then the intersections of these intersections, and so forth.

If we have built the factorization system with Alg. 2.6, then we choose as the set of maximal regions for CVM not the chosen sets $\tilde{\mathfrak{S}}$, but $\mathfrak{S}$, in order to capture more dependencies. On the other hand, if Alg. 5.1 was applied, it is more advisable to use $\tilde{\mathfrak{S}}$ for CVM, because it contains larger sets, and we are going to need the marginal distributions for these larger sets.

CVM and the 2-D Grid

For the 2-D grid, with the maximal regions being the 2×2 blocks, the result of the CVM is depicted in Fig. 6.7.

The maximal intersections of 2×2 blocks are blocks of two neighboring variables. These can be intersected again, resulting in regions with only one variable. The 4-variate regions have counting number 1, the 2-variate regions -1, and the univariate regions 1. This region graph is presented in [YFW02, MY02, TW03].

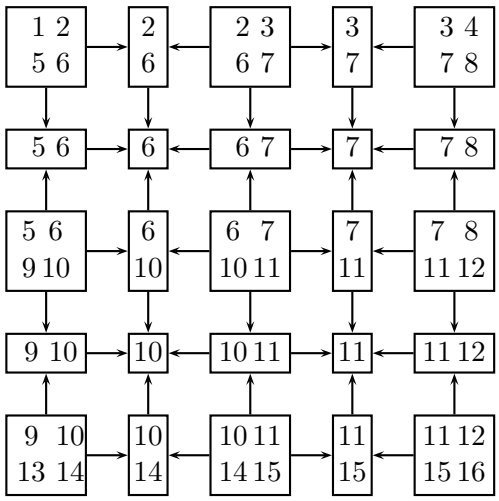

Figure 6.7.: The region graph for a rectangular grid, using 2×2 blocks

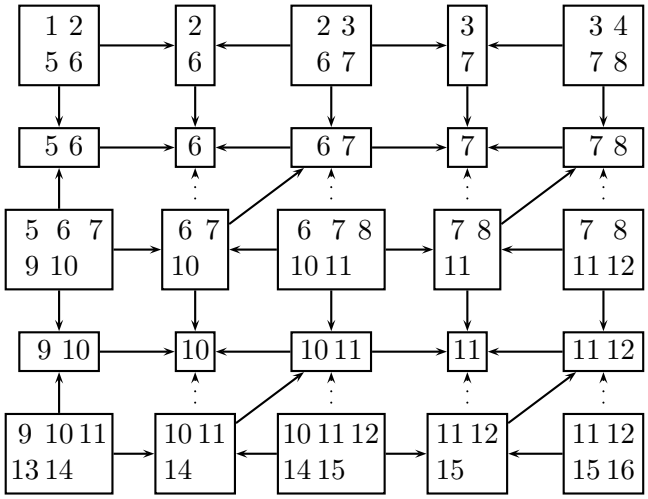

Figure 6.8.: The region graph for Kikuchi, pentavariate factorization

The pentavariate factorization system constructed by Alg. 5.1 results in the region graph depicted in Fig. 6.8.

This region graph is asymmetric, as the pentavariate factorization. The dotted edges in Fig. 6.8 are superfluous: Due to the diagonal edges, they connect grandparent and grandchild. However, they can also be included without changing the free energy. It only changes the dynamics of the algorithm. Empirically it was found that including these edges slightly speeds up stability and convergence, in that fewer iterations are needed. On the other hand, additional messages to be computed result in a larger effort per iteration.

CVM and the Circular Models

For the bivariate circle, we first apply CVM on the sets in (2.26). We find that the intersections between the sets contain one variable each. For the merged factorization (6.130), there is a slight difference owing to the trivariate factor $\{X_1, X_{n-1}, X_n\}$. Here again, the CVM adds univariate regions, except for the variable X_n, which appears only in one region. The region graphs for $n = 8$ are depicted in Fig. 6.9.

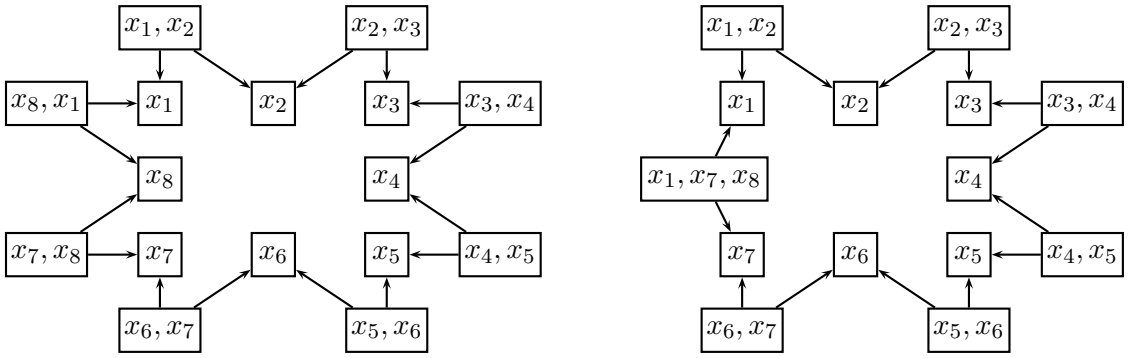

Figure 6.9.: The region graphs for the bivariate circle. On the left without subfunction join, on the right with subfunction join (x_8 was joined with its parents x_1 and x_7).

For the trivariate circle, without subfunction join the result of the CVM is very similar to the one of the bivariate circle: The intersections of the trivariate sets consist of single variables, which are added as the small regions of the region graph. The factorization with join gives a similar region graph, only there is one intersection with two variables, $\{X_1, X_{n-1}\}$. It is the only direct child of the region $\{X_1, X_{n-1}, X_n\}$.

Fig. 6.10 shows all the different graphical models for the trivariate circle. For comparison, the triangulated moral graph and the junction tree for this structure (see Sect. 3.4) are also included, along with the region graphs derived for the factorization systems without and with subfunction join.

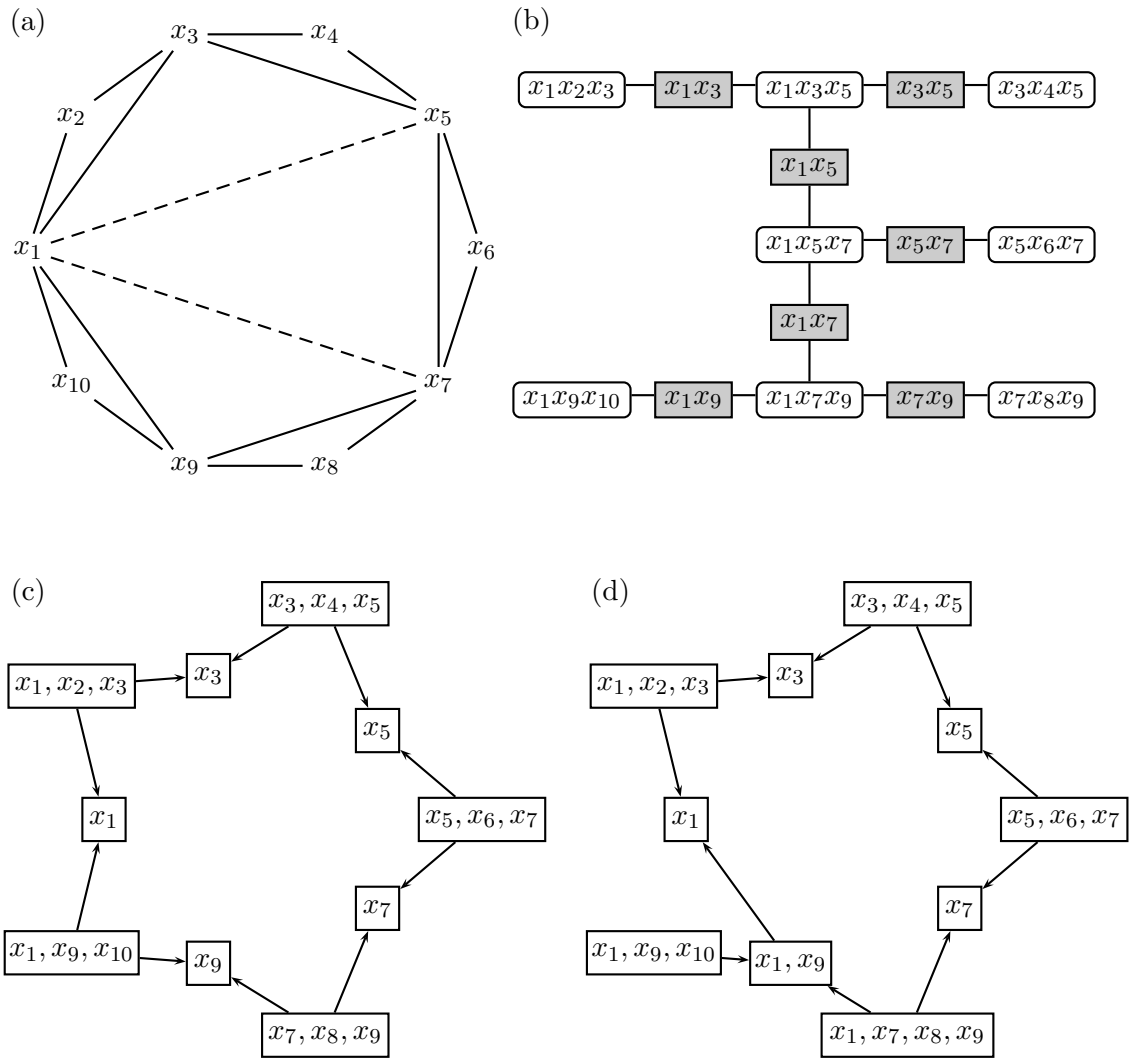

Figure 6.10.: The graphical models for a trivariate circle. (a) The moral graph; the dashed edges are included for triangulation. (b) The junction tree. (c) The region graph without subfunction join. (d) The region graph with subfunction join: Since x_1 and x_9 are connected, x_1 was included in x_7, x_8, x_9. The intersection of this enlarged set with x_1, x_9, x_{10} gives the new region x_1, x_9. In (c) and (d), all the maximal regions have counting number 1, all the small regions -1.

A heuristic rule that was pointed out in [YFW04] is that region graphs with

$$\sum_R c_R = 1 \tag{6.132}$$

give good approximations. We note that this is not fulfilled here: In the circular region graphs the sum of all counting numbers is 0. Certainly, there is an easy way to remedy this: triangulate the graph and go for the acyclic model which gives exact results (see Sect. 3.4.5); and its counting number sum is 1, too. This is surely favorable in real-world problems. Here we use the loopy models in order to gain insight into the behavior and properties of the method.

6.4.3. GBP on the Region Graph

Having constructed the region graph, the subfunctions of the ADF are included in all the regions which contain the required variables. At this point, we must choose a value for the inverse temperature β.

The choice of β is vital. If β is too low, the probability of the maximum is low, too. However, for too high values of β, convergence of the algorithm becomes more and more difficult. It needs more iteration steps to converge. The beliefs (6.93) and messages (6.102) depend exponentially on β, so for very large β they become numerically unstable.

It is possible to use a kind of "cooling schedule". This means to start with a low β and then use the result of GBP as the starting point for the next GBP run with a higher value of β.

GBP is performed using (6.102) and (6.104), as described above. For our optimization purpose, convergence of the GBP is not vital, and the resulting marginal distributions do not have to be very exact, since we use them only for sampling points, not for further calculations.

6.4.4. Calculate Marginals from the GBP Result and Sample Points

The construction of the region graph by CVM guarantees that for each $s_i \in \tilde{\mathfrak{S}}$ (be it constructed without or with subfunction merging), there is a region R in the region graph, such that $\mathbf{X}_{s_i} \subseteq \mathbf{X}_R$. So, we can compute the required conditional probability by

$$p(\mathbf{x}_{b_i}|\mathbf{x}_{c_i}) = \frac{\sum_{\mathbf{x}_R \setminus \mathbf{x}_{s_i}} q_R(\mathbf{x}_R)}{\sum_{\mathbf{x}_R \setminus \mathbf{x}_{c_i}} q_R(\mathbf{x}_R)} \ . \tag{6.133}$$

The marginals are consistent with each other (this is ensured by GBP). They can be combined to a distribution over the whole space

$$p(\mathbf{x}) = \prod_i p(\mathbf{x}_{b_i}|\mathbf{x}_{c_i}) \ . \tag{6.134}$$

This distribution approximates the Kikuchi approximation. Usually it does not fulfill the RIP. If it does, the region graph is cycle-free and equivalent to a junction tree; in this case the result is exact (see Theorems 3.9 and 6.5).

Using the factorization (6.134), points can be sampled similar to *FDA* (see Sect. 2.4.4).

6.4.5. BKDA: Bethe Kikuchi Distribution Algorithm

We summarize this scheme in the form of an algorithm, the Bethe Kikuchi Distribution Algorithm (*BKDA*), presented as Alg. 6.1.

Algorithm 6.1: *BKDA* – Bethe Kikuchi Distribution Algorithm

```
1  Given an additive decomposition 𝔖, construct a factorization system
   𝔖̃ using Alg. 2.6 or Alg. 5.1
2  From the larger one of 𝔖 and 𝔖̃, generate a region graph (ℛ, E_ℛ)
   using CVM.
3  Choose an initial β and a step size Δβ
4  do {
5     Calculate the local region energies E_R(x_R) (6.34)
6     do {
7        Iterate message passing (6.102) and (6.104)
8     } until messages converged or maximal number of steps reached
9     Compute local beliefs q_R(x_R) (6.93)
10    for i = 1 to N do {
11       Sample 𝔛_i using the factorization system 𝔖̃
12    }
13    β ⇐ β + Δβ
14 } until stopping condition fulfilled
```

It is not a population-based optimization algorithm, because the population at a time step is not used for generating the next graphical model. It is only used for the stopping condition (see Sect. 2.2.5) and to find the points of highest fitness.

The objective is to find the most probable point of the Boltzmann distribution. This objective is already mentioned in the introduction of [YFW04], but explicitly not pursued there.

Note that taking the result for a smaller β as a starting point for a higher β does not change the result of the algorithm, since the fixed points of GBP do not depend on the initial values. But it leads to faster convergence, since the difference between the results for β and for $\beta + \Delta\beta$ is small.

Since the result of GBP is only an approximation, and since the factorization distribution is not exact either (see p. 25), it should not be expected that the maximum of the Boltzmann distribution is also the most probable point to be sampled by *BKDA*. Nevertheless, it can be expected to have a not too low probability p_{opt}. When sampling N points, the probability to generate the optimum at least once is

$$p_N(\mathbf{x}_{\text{opt}}) = 1 - (1 - p_{\text{opt}})^N \tag{6.135}$$

So, if the goal is to hit the optimum with a probability higher than 95 %, the sample

size should be

$$N_{95\%} \geq \frac{\log 0.05}{\log(1 - p_{\text{opt}})} \tag{6.136}$$

To give an idea about the magnitude, some values are given in Table 6.1. Roughly, for $p_{\text{opt}} = 10^{-i}$, the sample size should be $3 \cdot 10^i$.

p_{opt}	$N_{95\%}$
10^{-1}	29
10^{-2}	299
10^{-3}	2995
10^{-4}	29956
10^{-5}	299572

Table 6.1.: Minimal sample size for hitting the optimum with probability 0.95

6.4.6. Related Work: Gibbs Sampling

In [San05], Gibbs sampling is used for generating points from a Kikuchi approximation. It uses a random walker $\mathbf{x}$, which is distributed according to the normalized Kikuchi distribution (6.40) on time average.

This is achieved by flipping bits in the bit vector $\mathbf{x}$ according to its Kikuchi value. One iteration of Gibbs sampling consists of the following steps:

- Choose randomly an $i \in \{1, \ldots, n\}$
- Let $\mathbf{x}' := \mathbf{x}$ with bit i flipped
- With probability

$$p_{\text{flip}} = \frac{k(\mathbf{x}')}{k(\mathbf{x}) + k(\mathbf{x}')} \tag{6.137}$$

  flip the bit in $\mathbf{x}$.

This stochastic process is a Markov chain [Rob96, Bré99], since the probability distribution at step t only depends on the value of $\mathbf{x}$ at step $t - 1$, not on the previous steps. It was shown [San04, San05] that the values of $\mathbf{x}$ are distributed according to the normalized Kikuchi distribution (6.40).

The difficulty of Gibbs sampling is that consecutive samples are dependent on each other, so if the samples at all time steps are used, this gives a very biased sample. To draw independent samples with this scheme, the algorithm must be run for a number of time steps, the *mixing time* of the Markov chain, for each value we want to sample. In many cases the mixing time is exponentially large. There are several methods to reduce the mixing time.

BKDA does not depend on such issues, because it does not need an iterative algorithm for sampling. Instead, we can sample directly from the factorization.

6.5. Numerical Results

6.5.1. Objectives of the Experiments

The goal of the experiments is to answer the following questions:

- Is the probability distribution obtained by *BKDA* a good approximation of the Boltzmann distribution?
- Is the probability to sample the optimum high? Thus, is the method applicable for optimization?
- What is the dependency of the inverse temperature β?
- Under what circumstances does the algorithm converge? How many generations does it need?
- What is the influence of different region graphs?

6.5.2. Definition of Circular Problems

To estimate the value of the concept, we first try it on simple circular problems. The results are then compared with the correct solution obtained by the junction tree method (see Sect. 3.4.5).

The following circular functions were investigated:

Deceptive 3-Circle: This function consists of blocks of three variables, on which the following fitness is defined:

$$f_{\text{Dec3}}(x,y,z) = 4xyz - x - y - z + 2 = \begin{cases} 2 & \Longleftrightarrow x+y+z=0 \\ 1 & \Longleftrightarrow x+y+z=1 \\ 0 & \Longleftrightarrow x+y+z=2 \\ 3 & \Longleftrightarrow x+y+z=3 \end{cases} \tag{6.138}$$

These blocks are arranged on a circle, overlapping by one variable:

$$f_{\text{Dec3C}}(\mathbf{x}) = \sum_{i=1} n/2 - 1 f_{\text{Dec3}}(x_{2i-1}, x_{2i}, x_{2i+1}) + f_{\text{Dec3}}(x_{n-1}, x_n, x_1) \tag{6.139}$$

where n, the number of variables, is even.

Iso-Circle: This is a bivariate function where all blocks except one have fitness

$$f_{\text{ordinary}}(x_i, x_{i+1}) = \begin{cases} n-1 & \Longleftrightarrow x_i = x_{i+1} = 1 \\ n & \Longleftrightarrow x_i = x_{i+1} = 0 \\ 0 & \text{otherwise} \end{cases} \tag{6.140}$$

Only the last block is defined differently, namely

$$f_{\text{special}}(x_n, x_1) = \begin{cases} n & \Longleftrightarrow x_i = x_{i+1} = 1 \\ 0 & \text{otherwise} \end{cases} \tag{6.141}$$

It is this last block which pulls the optimum of

$$f_{\text{Iso2C}}(\mathbf{x}) = \sum_{i=1}^{n-1} f_{\text{ordinary}}(x_i, x_{i+1}) + f_{\text{special}}(x_n, x_1) \tag{6.142}$$

to $\mathbf{x}_{\max} = (1, 1, \ldots, 1)$. There is a local optimum at $\mathbf{x}_{\text{loc}} = (0, 0, \ldots, 0)$ with $f_{\text{Iso2C}}(\mathbf{x}_{\text{loc}}) = f_{\text{Iso2C}}(\mathbf{x}_{\max}) - 1$ which appears optimal at all blocks expect for the special one. This makes the Iso-Circle problem very difficult for evolutionary optimization.

Zebra Circle: This bivariate problem is defined as the circular sum of the following blocks:

$$f_{\text{zebra}}(x, y) = \begin{cases} 0.49 & \Longleftrightarrow (x, y) = (0, 0) \\ 1 & \Longleftrightarrow (x, y) = (1, 0) \\ 0 & \text{otherwise} \end{cases} \tag{6.143}$$

The optimum of the problem (for even n) is at $(0, 1, 0, 1, \ldots, 0, 1)$. This problem is not difficult for evolutionary optimization, but we will see that is scales badly for methods like junction tree sampling or *BKDA*, because the probability of the maximum is rather small.

6.5.3. Results on Circular Problems

In Table 6.2, for the three circular problems the probabilities of the optimum are depicted for different values of β. The results of GBP without or with subfunction join are compared with the exact values calculated by the junction tree method.

We notice that for the deceptive and zebra circle the approximation is very good. The factorization with join gives a slight improvement. Comparison with the exact solution obtained with the junction tree method shows only little deviance in the probability of the optimum.

For the zebra problem, the optimum becomes very improbable. Nevertheless, it is the most probable point in the search space; but in this case sampling is a bad optimization method. It is more favorable to either use a method to calculate the most probable point or to prefer an EDA which does not depend on the probability, e. g. *FDA* with truncation selection.

We chose the zebra circle to investigate the dependency of β. The results are depicted in Fig. 6.11. For $\beta \to \infty$, the probability of an optimum goes to 0.5, because there are two optima, $010101\ldots$ and $101010\ldots$

The results seem quite satisfying. In both variants, the optimum will be found with a high probability for large β, although the curves depart from the correct solution

Problem	Size	Without join	With join	Junction tree
Deceptive Circle	10	0.580615	0.593571	0.596293
Deceptive Circle	20	0.349916	0.357724	0.363163
Deceptive Circle	30	0.210882	0.215588	0.21889
Deceptive Circle	40	0.127091	0.129927	0.131918
Deceptive Circle	50	0.076593	0.078302	0.0795022
Iso-Circle	10	0.9996	0.9996	0.731008
Iso-Circle	20	1	1	0.731059
Iso-Circle	30	1	1	0.731059
Iso-Circle	40	1	1	0.731059
Iso-Circle	50	1	1	0.731059
Zebra Circle	10	0.00133647	0.00250266	0.00255636
Zebra Circle	20	9.28684e-06	6.47017e-06	6.53497e-06
Zebra Circle	30	2.37405e-08	1.63548e-08	1.67057e-08
Zebra Circle	40	6.06892e-11	4.18087e-11	4.27058e-11
Zebra Circle	50	5.70751e-14	1.06878e-13	1.09171e-13

Table 6.2.: Results of *BKDA* on circular problems. Probability of sampling the known optimum using the factorization. $\beta = 1$.

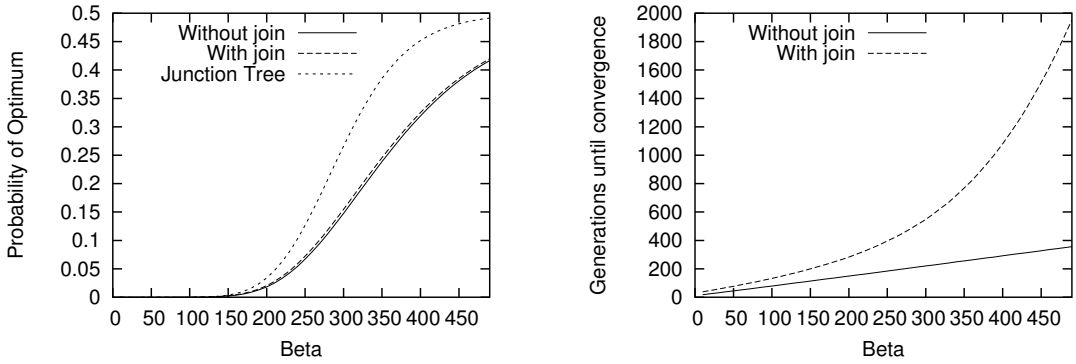

Figure 6.11.: Probability of the optimum and number of steps until convergence (with $\alpha = 0.5$) for the zebra circle of size 50, for varying β.

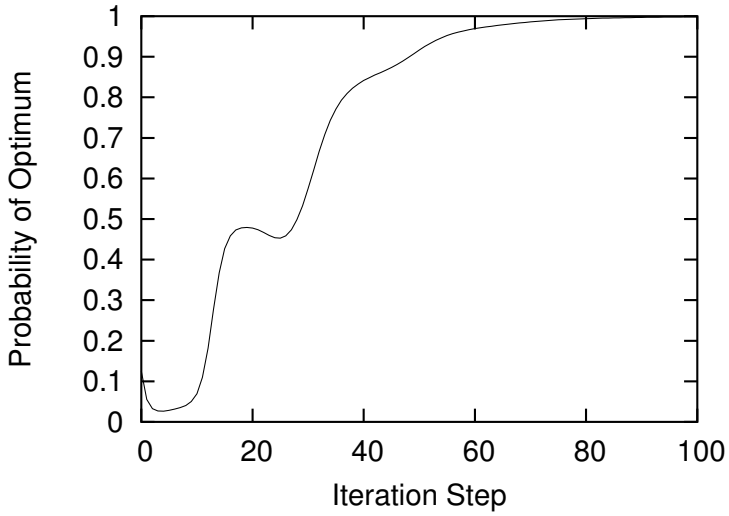

Figure 6.12.: Evolution of the probability of the optimum given by the factorization, for the Iso-Circle problem with $n = 10, \beta = 1, \alpha = 0.5$.

found with the junction tree. As expected, the version with join gives a slightly better probability than the version without. This little gain was to be expected, since there was only one edge added.

The number of required generations until convergence grows linear in β for the normal region graph and somewhat quadratically for the joined one.

The result for the Iso-Circle Problem is very different. The deciding difference is that here there is one local function different from the others, which has an enormous influence on the total probability. The information from this region must be passed on to the other regions. As was already pointed out in [Pea88], it is possible that by passing messages in circles, evidence is counted more than once, thus overestimated. An indication of this phenomenon can be seen in Fig. 6.12: The probability of the optimum starts rather low, then increases rapidly. We also note that convergence is much slower than for the two other problems: For $n = 10$, $\beta = 1$, and $\alpha = 0.5$, the Iso-Circle run depicted in Fig. 6.12 takes 331 iterations (304 with join) to converge, whereas the deceptive circle with the same parameters takes 37 and the zebra circle 16 iterations.

We further investigate this problem on a larger circle. For $n = 100$, Fig. 6.13(a) depicts the probability of the optimum within time; we can see that it rises to 1 with some oscillation between step 2000 and 3000. On the right of Fig. 6.13, bit 40 has been picked for investigating the evolution of the messages.

We note that in the following investigations all messages are normalized to sum up to one. The update equation of GBP in this special case is similar to (6.123):

$$m^{\tau,\mathrm{upd}}_{i-1,i\to i}(x_i) = \sum_{x_{i-1}} e^{\beta f_i(x_{i-1},x_i)} m^{\tau-1}_{i-2,i-1\to i-1}(x_{i-1}) \tag{6.144}$$

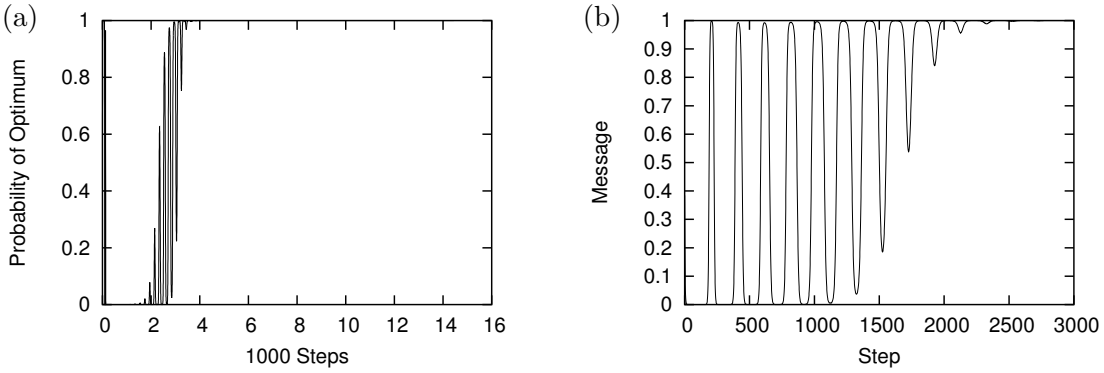

Figure 6.13.: (a): Evolution of the probability of the optimum given by the factorization, for the Iso-Circle problem with $n = 100, \beta = 1, \alpha = 0.5$. (b): Evolution of the message $m_{39,40\to 40}(X_{40} = 1)$ during the same run.

with f_i according to (6.141) or (6.140), respectively.

The messages from left to right $m_{i-1,i\to i}$ are fed through the circle just as the messages from right to left $m_{i,i+1\to i}$, without interacting with each other. Since the circle is symmetric and so both messages are identical, we concentrate on the left-to-right messages.

Fig. 6.13(b) shows $m_{39,40\to 40}(X_{40} = 1)$, the message that region $\{40\}$ received from region $\{39, 40\}$. We recall that the region which pulls the beliefs to 1 is $\{0, 99\}$. After about 200 steps, the information from there has reached bit 40. The message rises to 1, but then drops to 0 again. This oscillation repeats with a frequency of 200 steps, until its amplitude decreases and the message stays at 1.

The messages that all bits receive from the left are shown in Fig. 6.14, as snapshots at some steps during the run. The messages at the left are all 1, the messages on the right 0, so we only see the borderline between these blocks. It proceeds to the right during the run, with the exception that the line for step 1500 is left of the line for step 1000. This coincides with Fig. 6.13(b), which also shows for the message to bit 40 a maximum at step 1000 and a minimum near step 1500.

The borderline reaches the middle, bit 50, between steps 2000 and 3000. This is exactly the time when the probability of $\mathbf{x} = (1, 1, 1, \ldots, 1)$ jumps to one (showing the same oscillation with a frequency of 200 steps). Remember that exactly the same messages have been sent from region $\{0, 99\}$ to the right and to the left. So in step 2000-3000, the two waves of 1-messages meet at bit 50. We recall that in this special case, (6.93) reads

$$q_{i,i+1}(x_i, x_{i+1}) \propto e^{\beta f_i(x_i,x_{i+1})} m_{i-1,i\to i}(x_i) m_{i+1,i+2\to i+1}(x_{i+1}) \tag{6.145}$$

$$q_i(x_i) \propto m_{i-1,i\to i}(x_i) m_{i,i+1\to i}(x_i) \tag{6.146}$$

So if one of the messages is very near to $m(X_i = 1) = 1$, this affects the beliefs in the same way.

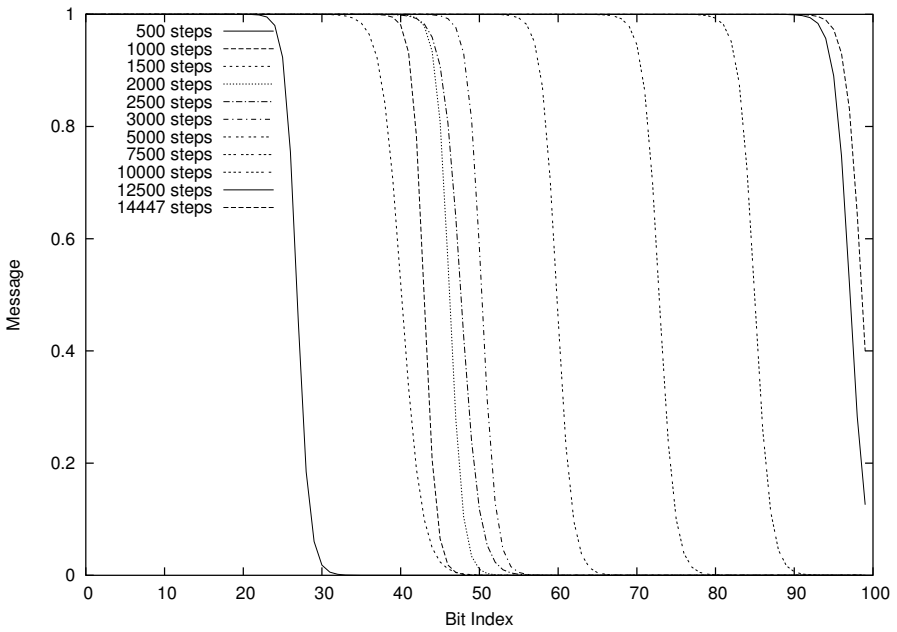

Figure 6.14.: Snapshots of the messages $m_{i-1,i\to i}(X_i = 1)$ during the same run as in Fig. 6.13 (Iso-Circle, $n = 100, \beta = 1, \alpha = 0.5$). On the left, the messages quickly converge to 1, and the very clear-cut border between messages near 1 and near 0 slowly moves to the right. There is a slight oscillation (note that the borderline retreats between step 1000 and 1500). Convergence is not reached before the borderline has reached the far side.

Figure 6.15.: The messages for the first 1500 steps of the Iso-Circle problem with $n = 100, \beta = 1, \alpha = 0.5$. 0 is black, 1 is white. The oscillation can be recognized clearly. Abscissa is steps, ordinate bit index

The oscillation of the messages can be contemplated again in Fig. 6.15. This image shows the value of all messages for the first 1500 iteration steps in greyscales, 0 being black and 1 white. The white wave of messages can be seen oscillating with decreasing amplitude. Recall that it reaches the far side of the bit vector in step 14447, so we see about 10 % of the whole run.

6.5.4. Grid Problems

Grid-like problems are a more formidable task. They contain much more cycles. We have applied the method on the Deceptive-4-Grid problem (5.2), a grid-like Iso problem, and on random instances of the Ising ground state problem.

We have used the factorization using 4-blocks and the pentavariate from Sect. 5.1.2. For comparison, sometimes the results of the exact junction tree method are included, too. This is only possible for small problems, since for an $m \times m$ grid the clique size of the junction tree is $2m$ (two consecutive rows).

Size	4-Grid		Penta-Grid		Junction tree
16	0.918358	31	0.918454	32	0.91842
25	0.918748	47	0.918809	44	0.918813
36	0.917477	37	0.917537	38	0.91754
49	0.916179	33	0.91624	36	0.916243
64	0.914882	32	0.914943	36	0.914946
81	0.913587	32	0.913648	36	0.913651
100	0.912294	32	0.912354	36	0.912357
121	0.911002	32	0.911062	36	0.911065

Table 6.3.: Results of *BKDA* on Deceptive-4-Grid. Depicted is the probability of sampling the known optimum using the factorization and the number of steps until convergence, with the 4-grid and pentavariate grid model, and the exact junction tree probability. $\beta = 1$, $\alpha = 0.5$.

The results for the deceptive grid are presented in Table 6.3. It can be noted that the probability of the optimum is almost independent of the grid size. The problem is very well-behaved. Both factorizations can reproduce the probability of the optimum almost exactly; the pentavariate factorization is slightly more accurate, but requires slightly more iterations. The number of iterations is independent of the grid size.

Size	4-Grid		Penta-Grid		Junction tree
16	0.575591	41	0.575591	50	0.575591
25	0.576116	76	0.576116	107	0.576116
36	0.576118	128	0.576117	185	0.576117
49	0.576117	201	0.576117	330	0.576117
64	0.576117	305	0.576117	506	0.576117
81	0.576117	428	—	—	0.576117
100	0.576117	599	—	—	0.576117
121	—	—	—	—	0.576117

Table 6.4.: Results of *BKDA* on Iso-4-Grid. Depicted is the probability of sampling the known optimum using the factorization and the number of steps until convergence, with the 4-grid and pentavariate grid model, and the exact junction tree probability. "—" denotes divergence. $\beta = 1$, $\alpha = 0.5$.

The Iso-Grid problem is defined on 2×2 blocks, too. The value of the block subfunctions depends only on u, the sum of the four bits in the block. For the $m \times m$ grid, let $k = (m-1)^2$ be the total number of blocks. Then the subfunction is for the upper left corner

$$f_{\text{special}}(u := x_{i,j} + x_{i+1,j} + x_{i,j+1} + x_{i+1,j+1}) = \begin{cases} k & \iff u = 4 \\ 0 & \text{otherwise} \end{cases} \tag{6.147}$$

and for all other blocks

$$f_{\text{ordinary}}(u := x_{i,j} + x_{i+1,j} + x_{i,j+1} + x_{i+1,j+1}) = \begin{cases} k-1 & \iff u = 4 \\ k & \iff u = 0 \\ 0 & \text{otherwise} \end{cases} \tag{6.148}$$

The effect is the same as in the Iso-Circle problem: Almost everywhere on the grid the local optimum $000\ldots0$ is optimal, but there is one special block which pulls the overall optimum to $111\ldots1$. The results on this function are given in Table 6.4.

For this problem, the probability of the optimum is identical for all all but the smallest grids (where the boundary has some influence). Also, both grids result in factorizations which reproduce exactly the correct probability of the optimum. In this case, the convergence slows down with the grid size, and for large grids, the method does not converge any more. This is again due to the subfunctions being dependent on the grid size. Therefore, the Boltzmann distribution for a constant β becomes more cliffy, and the iteration is more inclined to fail. A possibility to solve this problem is to scale the function or reduce β equivalently.

6.5.5. Results on the Ising Model

For application of the method on the Ising spin glass problem (see Sect. 6.1.1), we have generated ten random instances each of size 7×7, 10×10, and 15×15. They are named

	$\beta = 1$			$\beta = 10$		
Seed	4-Grid	Penta-Grid	J. Tree	4-Grid	Penta-Grid	J. Tree
1	4.07061e-06	8.52758e-06	1.17834e-05	0.18868	0.188618	0.188593
2	2.69087e-06	7.40403e-06	1.19999e-05	0.251567	0.19691	0.21759
3	8.02989e-06	1.47073e-05	2.151e-05	0.249284	0.249289	0.24929
4	9.09636e-07	2.11718e-06	3.25675e-06	0.364167	0.395517	0.41582
5	3.6114e-06	1.25921e-05	1.97924e-05	0.41223	0.415024	0.41436
6	2.47283e-07	4.28839e-07	6.74e-07	0.00225527	0.00230321	0.208269
7	1.63541e-06	5.15511e-06	7.14546e-06	0.483804	0.488443	0.488281
8	3.68671e-06	5.90075e-06	9.35291e-06	0.0845613	0.0849755	0.254494
9	3.46514e-07	1.17791e-06	1.67017e-06	0.127583	0.12507	0.125453
10	4.54769e-06	8.20245e-06	1.09055e-05	0.262239	0.262247	0.360243

Table 6.5.: Results of *BKDA* on Ising instances of size 7×7. Depicted is the probability of sampling the known optimum using the factorization, with the 4-grid and pentavariate grid model, and the exact junction tree probability. $\beta = 1$, $\alpha = 0.5$.

"rh$m$_s", where m is the grid size and $s \in \{1, \ldots, 10\}$ the random seed.[1] Their solutions, verified by the Cologne spin glass server [LPHJ03][2], are given in Appendix A.

We now present the probabilities of an optimum, as calculated using our factorization on the region graph. Keep in mind that the Ising problem is symmetrical. So in the best case, the probability of an optimum tends to 0.5; the inverse configuration then gets the other half of the probability.

Table 6.5 gives the results of our method on the instances of size 7×7. Here we observe first that varying β is crucial for the success of the algorithm. For all instances, the probability of sampling the optimum is much higher for $\beta = 10$. Also, the pentavariate grid gives an improvement over the 4-grid, sometimes slight, sometimes quite evident.

An exception is instance 2, where for $\beta = 10$ the pentavariate grid performs worse than the conventional 4-grid. But note that the pentavariate grid is sensitive to rotation and mirroring. Other rotations of the grid are much better.

What strikes the eye are instances 6 and 8. Here the distance to the true probability calculated using the junction tree is quite large. Instance 8 is not very severe: With $\beta = 100$, the optimum has a probability of 0.492639. But in instance 6, with increasing β the probability of the optimum stays low. It is not the case that the probability concentrates on another point: The most probable point has sampling probability 0.0894244 and fitness 30.0934 (whereas the optimum's fitness is 31.4087). In this instance, increasing β does not help. For $\beta = 100$, the optimum has a probability of $1.46195 \cdot 10^{-16}$, whereas the most probable point has a probability of 0.0312451 and fitness 29.9903. There are many cases like this, in which the GBP result does not focus on one point (like the theoretical Boltzmann distribution), but the distribution – and thus also the sampled

[1] The random Ising instances are available at http://www.hoens.net/robin/ising for download.
[2] http://www.informatik.uni-koeln.de/ls_juenger/research/sgs/sgs.html

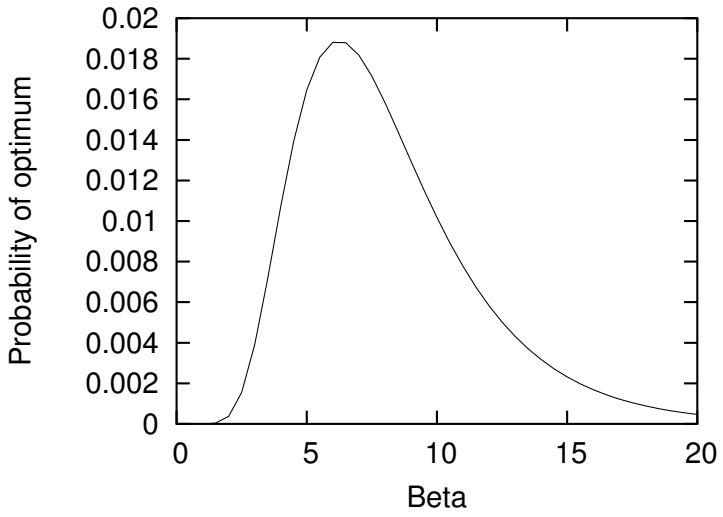

Figure 6.16.: Probability of the optimum for the Ising instance rh7_6 (instance 6 of size 7×7) for varying β (pentavariate grid).

population – stays dispersed.

In Fig. 6.16, this instance is investigated in detail. It can be seen that the probability reaches a maximum at $\beta = 6$, then declines again.

This behavior can be observed for the larger instances, too. The results for the instances of size 10×10 and 15×15 are depicted in the Tables 6.6 and 6.7. For 15×15 the junction tree method was too expensive and no longer applicable.

For the large grids the probability of the optimum calculated by our method is very different from the true probability computed using the junction tree. But for optimization, it is not necessary to reproduce this probability; the idea is to sample from the factorization distribution. And for this matter, all the grids of size 10×10 produce a probability which is high enough to find the optimum by sampling mostly 100 times, sometimes 1000 times. For the 15×15 10000 samples generally suffice.

Table 6.8 shows the result of some *BKDA* runs. GBP was run only for 100 iterations, because then the probabilities are already near the final values. The sample results are consistent with the probabilities in Table 6.6 and 6.7. Only the instances rh10_9, rh15_2, rh15_8, and rh15_10 did not sample the global optimum. But of these, only for rh15_8 the optimum would not be found by using the tenfold population (which is not very expensive). However, the found maximum is not very much worse.

The dependence on β for the Ising instances is depicted in Fig. 6.17. Here again, the instances behave very differently. For some easy instances, the probability rises constantly, for others it reaches a maximum and then drops again. For example, the difficult instance rh15_8 can be solved easier with $\beta = 5$.

	$\beta = 1$			$\beta = 10$		
Seed	4-Grid	Penta-Grid	J. Tree	4-Grid	Penta-Grid	J. Tree
1	1.56227e-13	5.53463e-13	1.25128e-12	0.0158715	0.0143937	0.0598039
2	1.33401e-12	1.41467e-11	4.8858e-11	0.00577445	0.0129105	0.0988492
3	5.55311e-12	2.56564e-11	5.95572e-11	0.133502	0.134356	0.144596
4	8.73883e-12	3.66898e-11	7.73724e-11	0.0114817	0.0221649	0.107321
5	7.42522e-12	2.4684e-11	5.21087e-11	0.0281239	0.0358553	0.0760384
6	6.29962e-12	3.10583e-11	9.27718e-11	0.0791284*	0.0794609*	0.21017
7	2.75456e-11	1.16145e-10	2.12433e-10	0.196065	0.220916	0.343579
8	3.13778e-12	1.33932e-11	3.28464e-11	0.000738162	0.000740625	0.187813
9	5.05162e-13	4.72522e-12	1.00587e-11	3.60448e-05	0.000253353	0.106966
10	3.26034e-11	2.7924e-10	4.52043e-10	0.000203547	0.0877621	0.219027

Table 6.6.: Results of *BKDA* on Ising instances of size 10×10. Depicted is the probability of sampling the known optimum using the factorization, with the 4-grid and pentavariate grid model, and the exact junction tree probability. $\beta = 1$, $\alpha = 0.5$ except *: $\alpha = 0.1$, convergence is unstable here.

	$\beta = 1$		$\beta = 10$	
Seed	4-Grid	Penta-Grid	4-Grid	Penta-Grid
1	2.86264e-26	6.91698e-24	3.70955e-05	0.00268086
2	4.49315e-26	2.44012e-23	2.8e-08	5.4562e-05
3	4.88295e-26	6.84016e-24	3.86662e-06	0.000147852
4	1.13528e-26	1.81016e-24	1.07569e-06	2.47108e-05
5	7.86338e-27	4.43875e-24	1.66141e-07	3.52817e-05
6	2.36921e-26	2.07336e-23	2.1235e-08	0.000383189
7	1.13156e-25	4.81003e-23	0.120241	0.123896
8	2.52638e-25	1.31434e-23	4.32802e-10	5.93849e-10
9	3.17878e-26	4.36585e-24	1.6629e-05	0.000109743
10	2.77733e-27	1.35195e-25	6.52958e-06	3.25408e-06

Table 6.7.: Results of *BKDA* on Ising instances of size 15×15. Depicted is the probability of sampling the known optimum using the factorization, with the 4-grid and pentavariate grid model. $\beta = 1$, $\alpha = 0.5$.

	$10 \times 10,\ N = 1000$			$15 \times 15,\ N = 10000$		
Seed	avg	max	opt	avg	max	opt
1	64.45	66.00	66.00	162.51	165.43	165.43
2	70.91	73.36	73.36	*165.60*	*170.91*	*171.00*
3	71.90	72.70	72.70	164.70	167.66	167.66
4	70.77	72.52	72.52	161.96	164.47	164.47
5	71.59	73.54	73.54	164.94	168.54	168.54
6	71.41	72.80	72.80	166.70	168.82	168.82
7	73.50	73.98	73.98	168.76	169.82	169.82
8	69.80	70.73	70.73	*165.57*	*168.46*	*169.02*
9	*67.39*	*70.60*	*71.00*	162.04	166.84	166.84
10	75.44	76.21	76.21	*159.12*	*161.85*	*162.12*

Table 6.8.: Result of sampling from *BKDA*, pentavariate grid, $\beta = 10$, after 100 steps of GBP with $\alpha = 0.5$. Sample size $N = 1000$ for the 10×10 instances, $N = 10000$ for 15×15. Given is the average of the energy in the sample, the maximum energy and the global optimum. The runs that missed the optimum are emphasized.

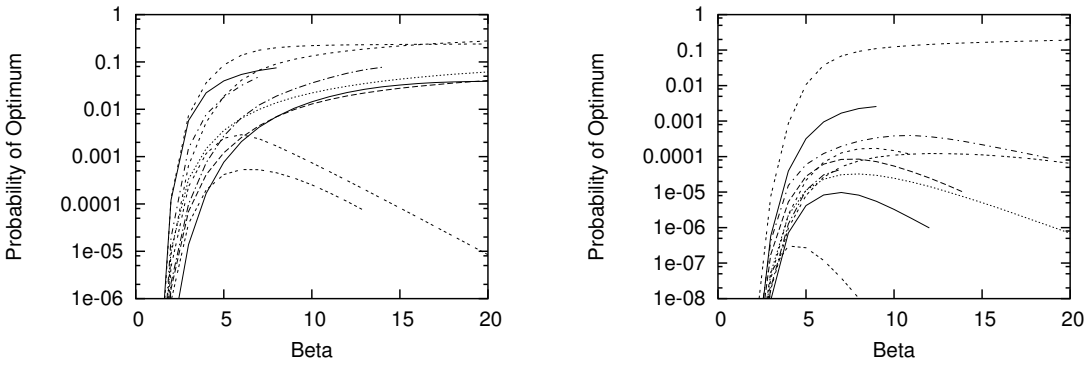

Figure 6.17.: Probability of the optimum for the Ising instances of size 10×10 and 15×15 for varying β. Pentavariate grid, $\alpha = 0.5$.

	FDA 4-grid			*MEFDA* 4-grid			*FDA* 5-grid			*MEFDA* 5-grid		
Seed	SR	Gen	Sdv	SR	Gen	Sdv	SR	Gen	Sdv	SR	Gen	Sdv
1	92	7.45	1.13	98	6.90	0.84	81	7.11	0.96	100	6.95	0.85
2	88	7.15	0.88	96	7.11	0.83	81	7.23	0.98	98	6.98	1.06
3	99	6.67	0.83	100	6.28	0.65	97	6.63	0.68	100	6.37	0.75
4	87	8.09	1.13	94	7.54	1.00	79	7.59	1.01	99	7.65	0.97
5	79	7.65	1.23	85	7.53	1.35	83	7.16	1.03	80	7.54	1.21
6	75	7.99	1.17	85	8.29	1.21	66	7.98	1.18	89	8.28	1.24
7	78	7.44	0.83	82	7.23	0.99	93	7.11	0.88	77	7.12	1.11
8	79	7.24	1.11	81	7.17	0.82	69	7.33	1.11	90	7.26	1.07
9	79	8.42	1.25	89	8.42	1.17	87	7.44	1.14	90	8.31	1.10
10	96	7.34	1.05	95	7.14	0.94	92	7.36	1.15	97	7.21	0.87

Table 6.9.: Results of *FDA* and *MEFDA* on the 7×7 Ising instances. All runs used the tetravariate or pentavariate grid factorization. *MEFDA* calculates the large marginals from the bivariates using IPF. SR gives the successful runs out of 100, Gen the average generations until success and Sdv the standard deviation of the generations until success. 100 runs, population size $N = 400$, truncation threshold $\tau = 0.3$.

6.5.6. Results of EDA on the Ising Instances

Table 6.9 shows the results of *FDA* and *MEFDA* on the Ising instances of size 7×7. To have a fair comparison with *BKDA*, these algorithms also use the tetra- and pentavariate grids.

Without this information, the conventional *FDA* (using the subfunction choice algorithm Alg. 2.6) uses a spanning tree of bivariate marginals and performs very badly; *FDA* with subfunction merge (Alg. 5.1) and *MEFDA* (Alg. 5.2) use mostly trivariate marginals and perform significantly worse than the tetra- and pentavariate versions.

The table shows that *FDA* and *MEFDA* using the same factorization structure as *BKDA* are able to solve the given instances, too. On most instances *MEFDA* pentavariate is the most successful variant, but the difference is not very large. They need between 3000 and 4000 fitness function evaluations.[3]

BKDA needs much less fitness evaluations. Even the difficult instance rh7_6 can be solved by sampling about 100 individuals, using an appropriate β (see Fig. 6.16). On the other hand, *BKDA* must perform the GBP routine, which needs no function evaluations (as soon as the region energies are computed), but requires some effort, too.

The Tables 6.10 and 6.11 give the results of *FDA* and *MEFDA* on the 10×10 and 15×15 instances. Of course, these instances need a much higher population size and more generations to converge. This results in a higher number of required function

[3]The number of function evaluations is $N + g(N-1)$, where g is the number of generations; the first N for the initial population, and in the following generations $N-1$ each, since in elitist runs the best individual remains untouched.

	FDA 4-grid			*MEFDA* 4-grid			*FDA* 5-grid			*MEFDA* 5-grid		
Seed	SR	Gen	Sdv	SR	Gen	Sdv	SR	Gen	Sdv	SR	Gen	Sdv
1	17	14.35	1.27	19	14.53	1.50	16	14.00	1.37	15	14.40	1.18
2	10	15.10	2.38	9	15.22	1.64	13	13.46	1.33	9	15.33	1.80
3	12	13.50	1.51	13	13.31	1.18	13	12.15	0.99	13	13.08	1.26
4	14	13.29	1.27	12	13.83	1.40	13	12.62	1.66	10	13.70	0.82
5	20	13.05	1.43	20	13.50	0.95	20	12.55	1.05	20	13.45	1.50
6	15	12.67	0.82	10	14.10	1.91	19	11.89	0.88	17	14.24	1.89
7	19	12.05	1.08	20	12.25	0.79	18	11.28	1.18	19	12.47	0.96
8	10	12.60	1.26	13	12.62	1.04	14	12.29	0.91	16	12.88	1.20
9	4	14.75	0.50	6	14.67	1.51	18	12.72	1.02	6	14.33	0.82
10	14	13.21	0.97	15	12.87	1.60	16	11.81	1.28	14	13.14	1.23

Table 6.10.: Results of *FDA* and *MEFDA* on the 10×10 Ising instances. Values and parameters are the same as in Table 6.9, except for 20 runs and population size $N = 1000$.

	FDA 4-grid			*MEFDA* 4-grid			*FDA* 5-grid			*MEFDA* 5-grid		
Seed	SR	Gen	Sdv	SR	Gen	Sdv	SR	Gen	Sdv	SR	Gen	Sdv
1	6	23.00	1.10	8	24.88	1.25	10	19.70	1.25	9	25.22	1.30
2	8	24.00	1.20	5	26.80	0.84	14	20.64	0.63	5	26.80	1.64
3	5	23.00	1.22	12	25.50	1.45	17	20.47	1.07	11	25.36	1.29
4	1	23.00	—	0	—	—	0	—	—	2	26.00	0.00
5	0	—	—	0	—	—	20	20.95	0.94	0	—	—
6	0	—	—	0	—	—	9	21.22	1.56	0	—	—
7	11	23.00	2.05	11	25.36	2.91	20	19.75	0.85	10	25.10	0.99
8	7	21.57	0.98	4	24.50	0.58	4	20.25	0.96	2	25.00	0.00
9	19	23.59	1.43	16	27.13	1.59	15	20.47	0.83	11	26.91	0.70
10	9	22.78	0.83	19	24.16	0.83	17	21.06	0.97	15	24.73	1.49

Table 6.11.: Results of *FDA* and *MEFDA* on the 15×15 Ising instances. Values and parameters are the same as in Table 6.9, except for 20 runs and population size $N = 6000$.

evaluations (10000 to 15000 for 10×10 and around 150000 for 15×15). Both are higher than the values of *BKDA*; Tables 6.6 and 6.7 or Fig. 6.17 show that *BKDA* can sample the optimum of the instances with 100 to 1000 trials for 10×10 and with 1000 to 10000 trials for 15×15. This comparison indicates that both methods (*EDA* and *BKDA*) scale similarly with the grid size.

Whereas on the 7×7 grid, *MEFDA* performs better than *FDA* on most instances, for large problems and correspondingly large population sizes it loses its advantage. Especially for 15×15, *FDA* with the pentavariate grid structure is the clear winner. For the large grids only 20 runs were performed, which makes the results more sensitive to fluctuation. The differences of the algorithms on instance rh15_9, for example, are not significant. But on most of the instances *FDA* with pentavariate grid is the most successful, most clearly on instance rh15_5.

The instances are of very different complexity. Instances rh15_4 and rh15_8 are very difficult for *FDA*. For *BKDA*, the most difficult instances are rh15_8 and rh15_10. On this grounds it cannot be judged definitely whether a hard instance for *FDA* is also hard for *BKDA* or not.

In Sect. 8.4, hybrid EDA with local search are applied on these Ising spin glass instances.

6.6. The Concave Convex Procedure

The Concave Convex Procedure (CCCP) [Yui02] is a variant of GBP. It also leads to a stationary point of the free energy. CCCP is slower than GBP, but unlike GBP it is guaranteed to converge. In the *BKDA*, CCCP can readily replace GBP when the need arises.

6.6.1. Convex and Concave Lagrangian

We now derive the CCCP update procedure, following [Yui02]. The starting point is a Lagrangian similar to (6.86). The only difference is that in GBP (the parent to child algorithm) consistency of neighboring regions is ensured by the more sophisticated consistency lemma (Lemma 6.8), whereas CCCP uses the straightforward consistency condition (6.73) which leads to the Lagrangian terms

$$\lambda_{PR}(\mathbf{x}_R) \left(\sum_{\mathbf{x}_P \setminus \mathbf{x}_R} q_P(\mathbf{x}_P) - q_R(\mathbf{x}_R) \right) \quad . \tag{6.149}$$

The Lagrangian to be minimized is

$$L = \sum_{R \in \mathcal{R}} c_R \left(\sum_{\mathbf{x}_R} q_R(\mathbf{x}_R) \beta E_R(\mathbf{x}_R) + \sum_{\mathbf{x}_R} q_R(\mathbf{x}_R) \log q_R(\mathbf{x}_R) \right) + \sum_{R \in \mathcal{R}} \gamma_R \left(1 - \sum_{\mathbf{x}_R} q_R(\mathbf{x}_R) \right) + \sum_{(P,R) \in E_{\mathcal{R}}} \sum_{\mathbf{x}_R} \lambda_{PR}(\mathbf{x}_R) \left(\sum_{\mathbf{x}_P \setminus \mathbf{x}_R} q_P(\mathbf{x}_P) - q_R(\mathbf{x}_R) \right) \tag{6.150}$$

The basic idea of CCCP is now to split up L in a convex and a concave part. The problematical part is the entropy term: For regions with $c_R > 0$, the entropy term is convex, for regions with $c_R < 0$ it is concave. The average energy and the constraints are linear in the q_R, so it does not matter where we put them.

To avoid an awkward case separation into convex and concave regions, we set

$$c_{\max} = \max_R c_R \tag{6.151}$$

and use this definition to split up L into a convex part

$$L_{\text{vex}} = \sum_{R \in \mathcal{R}} c_{\max} \left(\sum_{\mathbf{x}_R} q_R(\mathbf{x}_R) \beta E_R(\mathbf{x}_R) + \sum_{\mathbf{x}_R} q_R(\mathbf{x}_R) \log q_R(\mathbf{x}_R) \right) + \sum_{R \in \mathcal{R}} \gamma_R \left(1 - \sum_{\mathbf{x}_R} q_R(\mathbf{x}_R) \right) + \sum_{(P,R) \in E_{\mathcal{R}}} \sum_{\mathbf{x}_R} \lambda_{PR}(\mathbf{x}_R) \left(\sum_{\mathbf{x}_P \setminus \mathbf{x}_R} q_P(\mathbf{x}_P) - q_R(\mathbf{x}_R) \right) \tag{6.152}$$

and a concave part

$$L_{\text{ave}} = \sum_{R \in \mathcal{R}} (c_R - c_{\max}) \left(\sum_{\mathbf{x}_R} q_R(\mathbf{x}_R) E_R(\mathbf{x}_R) + \sum_{\mathbf{x}_R} q_R(\mathbf{x}_R) \log q_R(\mathbf{x}_R) \right) \tag{6.153}$$

It is easy to see that $L = L_{\text{vex}} + L_{\text{ave}}$.

6.6.2. Outer and Inner Loop

GBP equates the local beliefs of the regions to derive update equations for the messages $m^{\tau}_{P \to R}(\mathbf{x}_R)$. CCCP updates the beliefs and messages in turn. It consists of an *inner loop* in which the messages are updated until convergence, and an *outer loop* in which the current estimates of the beliefs are updated. The inner loop uses the iteration index τ (like GBP), and the outer loop uses the iteration index ξ.

The Outer Loop

For the outer loop iteration the ansatz is

$$\nabla L_{\text{vex}}^{\xi+1} + \nabla L_{\text{ave}}^{\xi} = 0 \tag{6.154}$$

where ∇L denotes[4] the vector of the partial derivatives of L with respect to the beliefs $q_R(\mathbf{x}_R)$. These derivatives are

$$\frac{\partial L_{\text{vex}}}{\partial q_R(\mathbf{x}_R)} = c_{\max}\left(\beta E_R(\mathbf{x}_R) + \log q_R(\mathbf{x}_R) + 1\right) - \gamma_R - \sum_{P|(P,R)\in E_{\mathcal{R}}} \lambda_{PR}(\mathbf{x}_R) + \sum_{C|(R,C)\in E_{\mathcal{R}}} \lambda_{RC}(\mathbf{x}_C) \tag{6.155}$$

and

$$\frac{\partial L_{\text{ave}}}{\partial q_R(\mathbf{x}_R)} = (c_R - c_{\max})\left(\beta E_R(\mathbf{x}_R) + \log q_R(\mathbf{x}_R) + 1\right) \ . \tag{6.156}$$

Inserting (6.155) and (6.156) into (6.154) yields

$$c_{\max}\left(\beta E_R(\mathbf{x}_R) + \log q_R^{\xi+1}(\mathbf{x}_R) + 1\right) - \gamma_R - \sum_{P|(P,R)\in E_{\mathcal{R}}} \lambda_{PR}(\mathbf{x}_R) + \sum_{C|(R,C)\in E_{\mathcal{R}}} \lambda_{RC}(\mathbf{x}_C) + (c_R - c_{\max})\left(\beta E_R(\mathbf{x}_R) + \log q_R^{\xi}(\mathbf{x}_R) + 1\right) = 0 \ . \tag{6.157}$$

Solving this for $q_R^{\xi+1}(\mathbf{x}_R)$ gives the update equations for the beliefs in the outer loop:

$$q_R^{\xi+1}(\mathbf{x}_R) = q_R^{\xi}(\mathbf{x}_R)^{\frac{c_{\max}-c_R}{c_{\max}}} \exp\left[-\frac{c_R}{c_{\max}}\beta E_R(\mathbf{x}_R) + \frac{\gamma_R - c_R}{c_{\max}} + \frac{1}{c_{\max}}\left(\sum_{P|(P,R)\in E_{\mathcal{R}}} \lambda_{PR}(\mathbf{x}_R) - \sum_{C|(R,C)\in E_{\mathcal{R}}} \lambda_{RC}(\mathbf{x}_C)\right)\right] \tag{6.158}$$

For the regions with $c_R = c_{\max}$ the previous belief $q_R^{\xi}(\mathbf{x}_R)$ disappears in this equation.

Like in GBP, we introduce messages

$$m_{PC}(\mathbf{x}_C) := e^{\frac{1}{c_{\max}}\lambda_{PC}(\mathbf{x}_C)} \tag{6.159}$$

and choose γ_R appropriately for normalization, which changes the update equation to

$$q_R^{\xi+1}(\mathbf{x}_R) \propto q_R^{\xi}(\mathbf{x}_R)^{\frac{c_{\max}-c_R}{c_{\max}}} e^{-\frac{c_R}{c_{\max}}\beta E_R(\mathbf{x}_R)} \frac{\prod_{P|(P,R)\in E_{\mathcal{R}}} m_{PR}(\mathbf{x}_R)}{\prod_{C|(R,C)\in E_{\mathcal{R}}} m_{RC}(\mathbf{x}_C)} \tag{6.160}$$

[4] ∇: nabla

The Inner Loop

The inner loop update equation for the messages can be derived by inserting (6.160) into the consistency equation (6.73):

$$\sum_{\mathbf{x}_P \setminus \mathbf{x}_R} q_P(\mathbf{x}_P) = q_R(\mathbf{x}_R) \tag{6.161}$$

$$\begin{aligned}\sum_{\mathbf{x}_P \setminus \mathbf{x}_R} q_P^{\xi}(\mathbf{x}_P)^{\frac{c_{\max}-c_P}{c_{\max}}} e^{-\frac{c_P}{c_{\max}}\beta E_P(\mathbf{x}_P)} \frac{\prod_{Q|(Q,P)\in E_{\mathcal{R}}} m_{QP}(\mathbf{x}_P)}{\prod_{C|(P,C)\in E_{\mathcal{R}}} m_{PC}(\mathbf{x}_C)} \\ = q_R^{\xi}(\mathbf{x}_R)^{\frac{c_{\max}-c_R}{c_{\max}}} e^{-\frac{c_R}{c_{\max}}\beta E_R(\mathbf{x}_R)} \frac{\prod_{Q|(Q,R)\in E_{\mathcal{R}}} m_{QR}(\mathbf{x}_R)}{\prod_{C|(R,C)\in E_{\mathcal{R}}} m_{RC}(\mathbf{x}_C)}\end{aligned} \tag{6.162}$$

The message $m_{PR}(\mathbf{x}_R)$ is independent of the summation variables $\mathbf{x}_P \setminus \mathbf{x}_R$, so it can be extracted from the sum. It appears in the denominator on the left side of (6.162) and in the numerator on the right side. This allows to solve the equation for this message:

$$m_{PR}(\mathbf{x}_R)^2 = \frac{\displaystyle\sum_{\mathbf{x}_P \setminus \mathbf{x}_R} q_P^{\xi}(\mathbf{x}_P)^{\frac{c_{\max}-c_P}{c_{\max}}} e^{-\frac{c_P}{c_{\max}}\beta E_P(\mathbf{x}_P)} \frac{\prod_{Q|(Q,P)\in E_{\mathcal{R}}} m_{QP}(\mathbf{x}_P)}{\prod_{C\neq R|(P,C)\in E_{\mathcal{R}}} m_{PC}(\mathbf{x}_C)}}{q_R^{\xi}(\mathbf{x}_R)^{\frac{c_{\max}-c_R}{c_{\max}}} e^{-\frac{c_R}{c_{\max}}\beta E_R(\mathbf{x}_R)} \frac{\prod_{Q\neq P|(Q,R)\in E_{\mathcal{R}}} m_{QR}(\mathbf{x}_R)}{\prod_{C|(R,C)\in E_{\mathcal{R}}} m_{RC}(\mathbf{x}_C)}} \tag{6.163}$$

With the abbreviations

$$g_R(\mathbf{x}_R) := q_R^{\xi}(\mathbf{x}_R)^{\frac{c_{\max}-c_R}{c_{\max}}} e^{-\frac{c_R}{c_{\max}}\beta E_R(\mathbf{x}_R)} \tag{6.164}$$

$$h_R(\mathbf{x}_R) := \frac{\prod_{Q|(Q,R)\in E_{\mathcal{R}}} m_{QR}^{\tau}(\mathbf{x}_R)}{\prod_{C|(R,C)\in E_{\mathcal{R}}} m_{RC}^{\tau}(\mathbf{x}_C)} \tag{6.165}$$

we can turn the fixed point equation (6.163) into the update equation

$$m_{PR}^{\tau,\mathrm{upd}}(\mathbf{x}_R) = m_{PR}^{\tau}(\mathbf{x}_R) \sqrt{\frac{\sum_{\mathbf{x}_P \setminus \mathbf{x}_R} g_P(\mathbf{x}_P) h_P(\mathbf{x}_P)}{g_R(\mathbf{x}_R) h_R(\mathbf{x}_R)}} \tag{6.166}$$

For the iteration of this formula, the same considerations as in Sect. 6.3.4 apply. Like in GBP, sequential update of the messages converges better than parallel update. In [Yui02], linear damping (6.103) with $\alpha = 0.1$ was used, but geometrical damping (6.104) works just as well.

6.6.3. Convergence of CCCP

If the inner loop (6.166) is iterated until convergence between single steps of the outer loop (6.160), convexity and concavity guarantee the algorithm to converge to a stationary

point of the free energy. In real application, it is not necessary to run the inner loop until convergence. In our experiments, we performed between five and ten steps of the inner loop before each update of the beliefs in the outer loop.

For our benchmark functions, we have always observed that CCCP converges to the same stationary point as GBP.

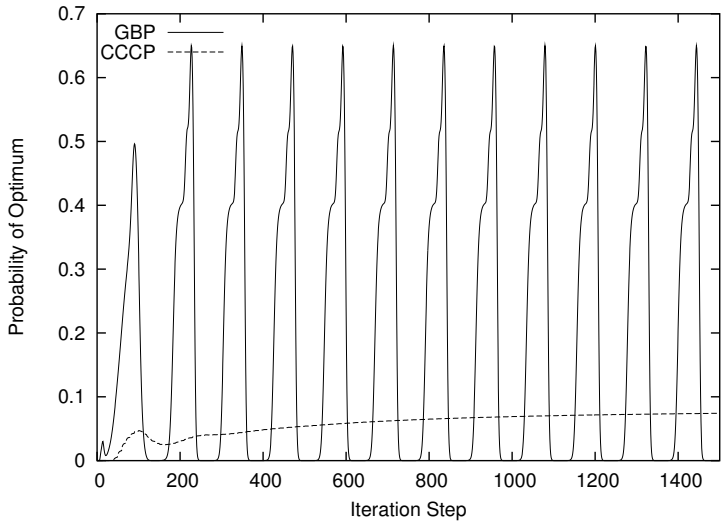

Figure 6.18.: Evolution of the probability of the optimum for the spin glass instance rh10_6, $n = 100$, $\beta = 10$, using the pentavariate grid. Given are the results of GBP and CCCP, both with geometrical damping $\alpha = 0.3$. For CCCP, one outer loop iteration after 10 inner loop steps. CCCP converges after 6647 steps.

Fig. 6.18 gives an example where CCCP is helpful. In Table 6.6 we have seen that the Ising spin glass instance rh10_6 is unstable with $\beta = 10$. Fig. 6.18 shows that GBP does not converge on this instance, but oscillates heavily. CCCP converges to the stationary point given in Table 6.6.

6.7. Summary

This chapter presented the use of loopy belief models in order to compute Boltzmann distributions. This research is inspired by statistical physics, so we first introduced the Ising spin glass model. Our fitness function is identified with a Hamiltonian energy. In this context, the minimum relative entropy principle is equivalent to minimization of the Gibbs free energy.

The loopy belief model used is the region graph, which can be understood as a generalization of the junction tree. We demonstrated how junction trees are equivalent to cycle-free region graphs. Generalized belief propagation (GBP) is an algorithm for minimizing the approximate Gibbs free energy within a region graph.

The new idea is to combine this technique with a factorization for the purpose of optimization. This is not an EDA, because it does not use populations and generations. Instead, the Boltzmann distribution is directly approximated, and its marginals are used for the factorization. The role of the inverse temperature is recognized and investigated.

The performance of this technique on circular and grid-like benchmark problems is analyzed. This gives insight into the dynamics of the generalized belief propagation algorithm and the suitability of the technique for solving Ising spin glass instances.

Sometimes GBP has difficulties to converge. In this case it can be replaced by a concave convex procedure (CCCP) which is more stable.

7. Learning Graphical Models For EDA

In this chapter we consider the topic of building a probability model – particularly a Bayesian network – from a given set of data. When there is no additive composition (see Sect. 2.4.4) known for the given problem, the dependencies of the variables can be learned from the data set. In the context of EDA, learning a Bayesian Network from the selected population and then sampling from this model are the basic steps of *LFDA* (Learning *FDA*, [MM99]).

Usually, Bayesian networks are learned using the "score and search" paradigm. This means that there is a score which measures the quality of the network and a local search which finds within a neighborhood the network which maximizes the score. In Sect. 7.1 we derive the BIC/MDL score and the hillclimber used in the *LFDA*.

Then some numerical results are presented which serve to prove that the algorithm is able to learn a correct structure. The success depends particularly on the population size. An analysis of the BIC/MDL space gives insight into the behavior of the learning algorithm.

7.1. Learning Graphical Models

In many optimization problems, an ADF structure of the objective function is known. But often, we have no structure information. Then an *EDA* can either use a pre-defined fixed structure (e. g. *UMDA* can be seen as *FDA* using an empty Bayesian network), or it uses the selected population to generate a Bayesian network. The network should capture the structure information, the independencies contained in the data. Generating a Bayesian network from data is called *learning*.

The intrinsic assumptions about the data made by this approach should be emphasized: We assume the selected population to originate from a Boltzmann distribution

$$p(\mathbf{x}) = \frac{1}{Z} e^{\beta f(\mathbf{x})} \tag{7.1}$$

with $f(\mathbf{x})$ being the fitness function. We assume that this Boltzmann distribution can be factorized polynomially, and that the conditional dependencies can be recognized from the data.

The model $M = (G, \Theta)$ that we intend to learn consists of a Bayesian network structure $G = (V, E)$ describing the conditional dependencies, and a set of conditional distributions $\Theta = \{p_M(x_i|\pi_i), i = 1, \ldots, n\}$, where Π_i are the parents of X_i in G. This model induces the probability distribution (3.19):

$$p_M(\mathbf{x}) = \prod_i p_M(x_i|\pi_i) \tag{7.2}$$

7.1.1. Relative Entropy or Log-Likelihood

Suppose that there is a collection of data $\mathfrak{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$. We want to learn a model which generates a factorized distribution most probable to sample the data $\mathfrak{X}$.

To this end we can use the principle of minimum relative entropy. From the population $\mathfrak{X}$ we estimate a distribution $p_{\mathfrak{X}}$. The simple, straightforward formula is

$$p_{\mathfrak{X}}(\mathbf{x}) = \frac{N(\mathbf{x})}{N} \ , \tag{7.3}$$

where $N(\mathbf{x})$ is the number of appearances of $\mathbf{x}$ in the population and N the population size. We will present the derivation with this formula, but Bayesian priors (see Sect. 4.3.2) can be used, too.

Following Sect. 4.1.3, we now assume that $p_{\mathfrak{X}}$ is the "true" distribution and we want to minimize the inefficiency of assuming the distribution p_M. (We will soon see that this assumption is insufficient, since the data is noisy.) Then the relative entropy to be minimized is:

$$\begin{aligned} D(p_{\mathfrak{X}} \| p_M) &= \sum_{\mathbf{x}} p_{\mathfrak{X}}(\mathbf{x}) \log \frac{p_{\mathfrak{X}}(\mathbf{x})}{p_M(\mathbf{x})} && (7.4)\\ &= \sum_{\mathbf{x}} p_{\mathfrak{X}}(\mathbf{x}) \log p_{\mathfrak{X}}(\mathbf{x}) - \sum_{\mathbf{x}} p_{\mathfrak{X}}(\mathbf{x}) \log p_M(\mathbf{x}) && (7.5)\\ &= -H(p_{\mathfrak{X}}) - \sum_{\mathbf{x}} p_{\mathfrak{X}}(\mathbf{x}) \log p_M(\mathbf{x}) && (7.6) \end{aligned}$$

The latter term is called the *Log-Likelihood* (LLH) of the data, given the model.

Definition 7.1. The **Log-Likelihood** of the data $\mathfrak{X}$ with the model M is

$$\mathcal{L}(\mathfrak{X}|M) = \sum_{\mathbf{x}} p_{\mathfrak{X}}(\mathbf{x}) \log p_M(\mathbf{x}) \tag{7.7}$$

This name is justified because the likelihood of the data is

$$\begin{aligned} p_M(\mathfrak{X}) &= \prod_{i=1}^{N} p_M(\mathbf{x}_i) && (7.8)\\ &= \prod_{\mathbf{x} \in \mathcal{D}} p_M(\mathbf{x})^{N(\mathbf{x})} \ . && (7.9) \end{aligned}$$

and the logarithm of this likelihood is

$$\begin{aligned} \log p_M(\mathfrak{X}) &= \sum_{\mathbf{x} \in \mathcal{D}} \log p_M(\mathbf{x})^{N(\mathbf{x})} && (7.10)\\ &= N \sum_{\mathbf{x} \in \mathcal{D}} p_{\mathfrak{X}}(\mathbf{x}) \log p_M(\mathbf{x}) && (7.11)\\ &= N \mathcal{L}(\mathfrak{X}|M) \ . && (7.12) \end{aligned}$$

$\mathcal{L}(\mathfrak{X}|M)$ is negative. Since $H(p_{\mathfrak{X}})$ is independent of the model, minimizing the relative entropy is equivalent to maximizing the log-likelihood.

Now we assume that M is a Bayesian network. Then for every variable X_i there is given a set of parents Π_i, and p_M is the factorization (3.19):

$$p_M(\mathbf{x}) = \prod_{i=1}^{n} p_M(x_i|\pi_i) \tag{7.13}$$

where the conditional distributions of the network are estimated using the given data, so (because of Lemma 2.6)

$$p_M(x_i|\pi_i) = p_{\mathfrak{X}}(x_i|\pi_i) \ . \tag{7.14}$$

Then we can derive the following lemma:

Lemma 7.1. *Let M be a Bayesian network model for the distribution $p_{\mathfrak{X}}$. Then*

$$\mathcal{L}(\mathfrak{X}|M) = -\sum_i H(X_i|\Pi_i) = \sum_i \sum_{x_i,\pi_i} p_{\mathfrak{X}}(x_i,\pi_i) \log p_{\mathfrak{X}}(x_i|\pi_i) \tag{7.15}$$

where π_i runs over all possible values of Π_i (the parents of X_i).

Proof. Using (7.13) and (7.14), we calculate

$$\begin{aligned}
\mathcal{L}(\mathfrak{X}|M) &= \sum_{\mathbf{x}} p_{\mathfrak{X}}(\mathbf{x}) \log p_M(\mathbf{x}) && (7.16)\\
&= \sum_{\mathbf{x}} p_{\mathfrak{X}}(\mathbf{x}) \log \prod_i p_M(x_i|\pi_i) && (7.17)\\
&= \sum_{\mathbf{x}} p_{\mathfrak{X}}(\mathbf{x}) \sum_i \log p_{\mathfrak{X}}(x_i|\pi_i) && (7.18)
\end{aligned}$$

Now we define $Q_i := \{X_1, \ldots, X_n\} \backslash (\{X_i\} \cup \Pi_i)$, all variables except X_i and its parents. So, we also split up the sum over $\mathbf{x}$, all values of $\mathbf{X}$, into a sum over x_i, π_i, q_i, where q_i runs over all values of Q_i. Then we can exchange the sums

$$\begin{aligned}
\mathcal{L}(\mathfrak{X}|M) &= \sum_i \sum_{x_i,\pi_i,q_i} p_{\mathfrak{X}}(\mathbf{x}) \log p_{\mathfrak{X}}(x_i|\pi_i) && (7.19)\\
&= \sum_i \sum_{x_i,\pi_i} \log p_{\mathfrak{X}}(x_i|\pi_i) \sum_{q_i} p_{\mathfrak{X}}(x_i,\pi_i,q_i) && (7.20)\\
&= \sum_i \sum_{x_i,\pi_i} p_{\mathfrak{X}}(x_i,\pi_i) \log p_{\mathfrak{X}}(x_i|\pi_i) && (7.21)\\
&= -\sum_i H(X_i|\Pi_i) && (7.22)
\end{aligned}$$

□

By adding an edge to a Bayesian network, the LLH will not decrease. This is stated in the following lemma.

Lemma 7.2. *Let $G = (V, E)$ be a Bayesian network, and $G^* = (V, E \cup \{(x_j, x_k)\})$ be the same network with an edge $x_j \to x_k$ added. Let $M = (G, \Theta)$ and $M^* = (G^*, \Theta^*)$. Then*

$$\mathcal{L}(\mathfrak{X}|M^*) = \mathcal{L}(\mathfrak{X}|M) + I(X_j, X_k|\Pi_k) \ , \tag{7.23}$$

where $I(X_j, X_k|\Pi_k)$ is the conditional mutual information of X_j and X_k given Π_k, in the distribution $p_{\mathfrak{X}}$.

Proof. Define $\Pi_k^* = \Pi_k \cup \{X_j\}$. From Lemma 7.1 follows

$$\mathcal{L}(\mathfrak{X}|M^*) = \sum_{i \neq k} \sum_{x_i, \pi_i} p_{\mathfrak{X}}(x_i, \pi_i) \log p_{\mathfrak{X}}(x_i|\pi_i) + \sum_{x_k, \pi_k^*} p_{\mathfrak{X}}(x_k, \pi_k^*) \log p_{\mathfrak{X}}(x_k|\pi_k^*) \tag{7.24}$$

$$= \sum_{i \neq k} \sum_{x_i, \pi_i} p_{\mathfrak{X}}(x_i, \pi_i) \log p_{\mathfrak{X}}(x_i|\pi_i) + \sum_{x_j, x_k, \pi_k} p_{\mathfrak{X}}(x_j, x_k, \pi_k) \log p_{\mathfrak{X}}(x_k|x_j, \pi_k) \tag{7.25}$$

$$= \mathcal{L}(\mathfrak{X}|M) - \sum_{x_k, \pi_k} p_{\mathfrak{X}}(x_k, \pi_k) \log p_{\mathfrak{X}}(x_k|\pi_k) + \sum_{x_j, x_k, \pi_k} p_{\mathfrak{X}}(x_j, x_k, \pi_k) \log p_{\mathfrak{X}}(x_k|x_j, \pi_k) \tag{7.26}$$

$$= \mathcal{L}(\mathfrak{X}|M) - \sum_{x_j, x_k, \pi_k} p_{\mathfrak{X}}(x_j, x_k, \pi_k) \log p_{\mathfrak{X}}(x_k|\pi_k) + \sum_{x_j, x_k, \pi_k} p_{\mathfrak{X}}(x_j, x_k, \pi_k) \log p_{\mathfrak{X}}(x_k|x_j, \pi_k) \tag{7.27}$$

$$= \mathcal{L}(\mathfrak{X}|M) + \sum_{x_j, x_k, \pi_k} p_{\mathfrak{X}}(x_j, x_k, \pi_k) \log \frac{p_{\mathfrak{X}}(x_k|x_j, \pi_k)}{p_{\mathfrak{X}}(x_k|\pi_k)} \tag{7.28}$$

$$= \mathcal{L}(\mathfrak{X}|M) + I(X_j, X_k|\Pi_k) \tag{7.29}$$

□

In the extreme case that X_j and X_k are conditionally independent given Π_k, the mutual information is zero, and $\mathcal{L}(\mathfrak{X}|M) = \mathcal{L}(\mathfrak{X}|M^*)$. But if there is only the slightest correlation between them, the LLH measure prefers M^* to M.

Remark 7.1. As a side note we mention another curious effect of this result: A variable that already has parents is more likely to receive another edge than other variables. This is due to the estimation of $I(X_j, X_k|\Pi_k)$ from the population using (7.3). The more parents a variable X_k has, the smaller is the sample size for estimating $p_{\mathfrak{X}}(x_k|\pi_k)$; so the noise of the estimate will be larger, leading to more spurious correlation. Bayesian priors (see Sect. 4.3.2) reduce this unwanted effect.

To combat the overfitting, we need a measure which prefers simple structures. A good measure with this property is the *Bayesian Information Criterion* (BIC) measure [Sch78], which is a special case of *Minimal Description Length* (MDL), introduced in [Ris78] and described in detail in [Ris89, Grü98].

7.1.2. Minimal Description Length

The MDL measure is rooted in coding theory. The principal idea is the following: Suppose that we have a set of data $\mathfrak{X}$, and we want to transmit the data in coded form over a transmission channel. Certainly it is favorable to code the data in a form that the length of the transmission, i. e. the *description length* is minimal. The earliest [Ris78] and clearest concepts are the *two-part codes*, consisting of an assumption about the data and the coding of the data using this assumption.

This idea can be applied to the problem of learning a Bayesian network [FG99]. The assumption is that the data was generated from a specific Bayesian network model $M = (G, \Theta)$. So, the coding of our data consists of two parts,

- the coding of the Bayesian network model M, consisting of the structure G and the probability distributions Θ, and
- the coding of the data $\mathfrak{X}$, using this model.

To put it in a formula, the description length is

$$\text{Length}(\mathfrak{X}) = \text{Length}(M) + \text{Length}(\mathfrak{X}|M) \tag{7.30}$$

The connection between a probability distribution and a coding, where probable points have short code lengths and less probable points longer code lengths, is formalized in the following lemma [CT89, Grü98]:

Lemma 7.3. *For all probability distributions $p(x)$ there exists a coding so that for every x the code length is*

$$\text{Length}(x) = -\log p(x) = \log \frac{1}{p(x)} \tag{7.31}$$

The expected keyword length is the entropy of the distribution:

$$E(\text{Length}(x)) = -\sum_x p(x) \log p(x) = H(p) \tag{7.32}$$

Conversely, if there is a code for x, there exists a probability distribution p fulfilling $\text{Length}(x) = -\log p(x)$ *for every x.*

Remark 7.2. This lemma neglects rounding effects and the need of code lengths to be natural numbers. If the coding is binary, so that the code lengths are given in *bits*, the entropy uses the binary logarithm.

We can use Lemma 7.3 to identify code lengths with probabilities. We now present the two parts of the coding. For reasons of clarity, we begin with the second part.

The Description of the Data Given the Model

Let $\mathfrak{X}$ be a data set of size N, $\mathfrak{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$. A Bayesian network model M induces a probability distribution p_M in the data space. From Lemma 7.3 follows the code length of the data:

$$\text{Length}(\mathfrak{X}|M) = -\sum_{i=1}^{N} \log p_M(\mathbf{x}_i) \tag{7.33}$$

If $N(\mathbf{x})$ is the number of times that $\mathbf{x}$ appears in the data, then we can further calculate

$$\begin{aligned} \text{Length}(\mathfrak{X}|M) &= -\sum_{\mathbf{x}} N(\mathbf{x}) \log p_M(\mathbf{x}) && (7.34)\\ &= -N \sum_{\mathbf{x}} \frac{N(\mathbf{x})}{N} \log p_M(\mathbf{x}) && (7.35)\\ &= -N\mathcal{L}(\mathfrak{X}|M) && (7.36) \end{aligned}$$

with Def. 7.1.

So we can also apply Lemma 7.1 and get

$$\text{Length}(\mathfrak{X}|M) = N \sum_{i=1}^{n} H(X_i|\Pi_i) \tag{7.37}$$

The Description of the Bayesian Network Model

A Bayesian network model consists of a graph G and a set of probability distributions. The graph can be coded simply by an incidence matrix $I_G \in \{0,1\}^{n\times n}$, with $I_G(i,j) = 1$ if X_i is a parent of X_j in the graph G. This coding needs n^2 bits.

Arguably, there exist more compact codings. Particularly if the graph is sparsely connected, it can be better to save a list of the parents for each node. Also, the fact that Bayesian networks are acyclic can be exploited in order to devise more compact codings. But since this part of the code is independent of N, for sufficiently large sample sizes the possible saving at this place is negligible.

The next part of the code is the length of the probability distributions $p_M(X_i|\Pi_i)$. To keep it simple, we will assume only binary variables. The generalization is straightforward and can be found in [Mah01, Jor99]. We need to save for every i only the probability $p_M(X_i = 1|\pi_i)$. There are $2^{|\Pi_i|}$ such probabilities, one for each assignment π_i of the parents.

The marginal probabilities are estimated from the data, so they are fractions given with a precision of $\frac{1}{N}$. They can be stored with a code length of $\log N$. However, [FY96] argue that the frequencies can be assumed to be distributed according to a Gaussian model, and that due to the central limit theorem it suffices to code only $\sqrt{N}$ possible frequency values. This results in a code length of $\frac{1}{2}\log N$.

Putting it all together, we get

$$\text{Length}(M) = \text{Length}(G) + \text{Length}(\Theta) \tag{7.38}$$

$$= n^2 + \frac{1}{2}\log N \sum_{i=1}^{n} 2^{|\Pi_i|} \tag{7.39}$$

The total description length is

$$\text{Length}(\mathfrak{X}) = \text{Length}(M) + \text{Length}(\mathfrak{X}|M) \quad = n^2 + \frac{1}{2}\log N \sum_{i=1}^{n} 2^{|\Pi_i|} + N \sum_{i} H(X_i|\Pi_i) \tag{7.40}$$

The first term n^2 is constant and independent of the network model, so it can be omitted. Thus we obtain the BIC measure (Bayesian Information Criterion):

$$\text{BIC}(M) = \frac{1}{2}\log N \sum_{i=1}^{n} 2^{|\Pi_i|} + N \sum_{i} H(X_i|\Pi_i) \tag{7.41}$$

This measure was originally derived (in a different manner) in [Sch78].

In the general *Minimal Description Length* measure, the two parts of the coding are weighted by a parameter α, to wit

$$\text{MDL}(M) = \alpha \log N \sum_{i=1}^{n} 2^{|\Pi_i|} + N \sum_{i} H(X_i|\Pi_i) \tag{7.42}$$

We call α the *structure penalty*, because it punishes complicated network structures. Setting $\alpha = 0$ is equivalent to the Log-Likelihood measure. The larger α is chosen, the sparser will be the resulting Bayesian network. Sensible values are between 0.1 and 2; the value 0.5 from the BIC measure is a good starting point.

7.1.3. LFDA: Learning Factorized Distribution Algorithm

Using the MDL measure, we can learn a Bayesian network from data and thus apply *FDA* on cases where no decomposition of the fitness function is known.

To learn a Bayesian network from data, the simplest algorithm is the greedy technique: Start with an empty graph, then add the edge to the network which minimizes MDL(M) (of course, do not add an edge that results in a cyclic network). This is repeated until no more improvement is possible.

Incorporating this technique into *FDA*, we obtain *LFDA* [MM99], Alg. 7.1.

In addition to the structure penalty, *LFDA* contains another technique to control the complexity of the network. A parameter *maxparents* is introduced, and edges are only permitted if the number of parents for each variable stays below *maxparents*.

Algorithm 7.1: *LFDA* – Learning Factorized Distribution Algorithm

```
1  t ⇐ 1. Generate an initial population with N individuals from the
   uniform distribution.
2  do {
3     Perform selection
4     G ⇐ (V, E) = ({X_1, ..., X_n}, ∅)
5     do {
6        bestmdl ⇐ MDL(G)
7        for all (X_i, X_j) ∉ E do {
8           E* ⇐ E ∪ {(X_i, X_j)}
9           if E* cycle-free and MDL(G* = (V, E*)) < bestmdl
                              and |Π*_j| ≤ maxparents
10             bestmdl ⇐ MDL(G*)
11             bestedge ⇐ (X_i, X_j)
12       }
13       if bestmdl < MDL(G)
14          E ⇐ E ∪ bestedge
15    } until no more improvement
16    Estimate the conditional probabilities p(x_i|π_i, t) from the selected
      points.
17    Generate new points according to p(x, t + 1) = ∏_{i=1}^{n} p(x_i|π_i, t).
18    t ⇐ t + 1.
19 } until stopping criterion reached
```

7.1.4. Other Learning Measures

The same idea was turned into an algorithm by several researchers at the same time [LL01]. During the same year, [EL99] developed the EBNA (Estimation of Bayesian Network Algorithm), [PGC99] the BOA (Bayesian Optimization Algorithm), and [MM99] the *LFDA* (Learning Factorized Distribution Algorithm).

EBNA comes in different variants, using MDL or also other measures. BOA prefers tournament selection, and a different Bayesian network measure called the *BDe score*. The BDe score [HGC95] comes from the theory of Bayesian learning and is a calculation of the posterior probability of a network, given the data. It was argued in [Bou94] that MDL and BDe are asymptotically equivalent, and that MDL is less sensitive to spurious correlations in the data. Therefore, this thesis will confine to MDL. A thorough discussion of these matters can be found in [Jor99].

There have also been attempts to learn other graphical models than Bayesian networks. In Sect. 5.2, we describe the PADA [SO00] and present some work for its improvement. In [San05] a Kikuchi approximation is learned from population data.

Furthermore, it has been pointed out that the simple greedy algorithm, adding one edge in each step, can be deceived and led to a suboptimal solution [XWC97]. There

exist more complicated algorithms for learning Bayesian networks. But the population sizes in *EDA* are rather small for intricate structure learning, and for the optimization purpose the network does not have to be perfect. Furthermore, since in every generation a new structure is learned from scratch, it seems like a waste of resources to use a too expensive learning algorithm.

7.2. Numerical Results

To demonstrate the learning algorithm, we apply it on the Deceptive-5 function. This function consists of blocks of five bits, on which the following fitness function is defined:

$$f_{\text{Dec5}}(u := x_i + x_{i+1} + x_{i+2} + x_{i+3} + x_{i+4}) = \begin{cases} 4 & \iff u = 0 \\ 3 & \iff u = 1 \\ 2 & \iff u = 2 \\ 1 & \iff u = 3 \\ 0 & \iff u = 4 \\ 5 & \iff u = 5 \end{cases} \tag{7.43}$$

In this function the five bits seem to be independent, with the value 0 preferable for the single bits. Only when all bits are 1, the global optimum is reached. It is difficult for evolutionary algorithms to recognize this dependency.

We present in Table 7.1 the result of *LFDA* on an instance of ten deceptive-5 blocks, so the dimension of the problem is $n = 50$.

Pop.	Succ.	Gene-	Generation 1		Generation 2		Generation 3	
Size	Rate	rations	edges	good	edges	good	edges	good
1000	0	—	29.21	10.46	41.35	15.18	47.80	17.79
3000	0	—	34.50	25.20	52.40	40.77	62.92	50.15
5000	56	7.34	51.33	44.76	76.09	69.90	86.73	81.03
6000	84	6.44	59.68	54.67	85.42	80.62	93.47	88.85
7000	99	5.85	68.80	64.25	92.51	88.84	97.07	93.44
10000	100	5.21	84.74	81.38	98.55	95.67	99.85	96.83

Table 7.1.: Result of *LFDA* on deceptive-5, 100 runs, $n = 50$, truncation selection with $\tau = 0.3$, *LFDA* structure penalty $\alpha = 0.5$. Given are the population size N, the success rate (the percentage of runs which find the optimum), the average number of generations required for this, and for the first three generations the average number of edges added to the model and the number of correct edges.

The success of *LFDA* in learning the correct structure as well as in finding the optimum depends heavily on the population size. We note that for $n = 50$ there are $\binom{50}{2} = 1225$ possible edges (not considering the direction), of which $10 \cdot \binom{5}{2} = 100$ are correct.

Pop. Size	Succ. Rate	Gene-rations	Generation 1 edges	Generation 1 good	Generation 2 edges	Generation 2 good	Generation 3 edges	Generation 3 good
1000	0	—	99.84	20.90	114.20	29.85	119.83	35.14
3000	63	7.21	93.50	51.66	114.86	78.00	120.67	87.31
5000	98	5.77	105.24	76.91	119.23	95.93	121.94	97.70
7000	100	5.29	112.81	90.16	119.71	99.05	121.82	99.30

Table 7.2.: Same as Table 7.1, except for $\alpha = 0.25$.

For small population sizes, fewer edges are added. This is plausible because a smaller population provides less confident dependencies. Also, the quality of the model improves with the population size. For very high population sizes, where the success rate approaches 100 %, almost all correct edges are added, and almost no superfluous edges (between different blocks of the deceptive function). During the run, the model improves slightly. The network which is built in the third generation is clearly better than the one for the first generation. This is no surprise, since the data has been generated with the previous network, and then the best individuals were selected.

In Table 7.2, identical experiments with a structure penalty of $\alpha = 0.25$ are presented. The results are similar to the results in Table 7.1. We see that due to the smaller α, more edges are added to the model. Also, the population size for solving the problem is smaller. In this case it seems that the superfluous edges are not problematic, as long as there are enough correct edges in the model.

Fig. 7.1 shows the influence of α on the behavior of *LFDA*. Three runs with different values of α (0.25, 0.5, and 1) are depicted in the MDL space; the x axis gives the complexity of the model, the y axis the log-likelihood. As can be seen, each Bayesian network begins at the bottom left corner, with a low log-likelihood and a small complexity. Then the complexity increases as edges are added to the network, while the log-likelihood improves first rapidly, then more slowly. The smaller α, the longer the trajectory of the run.

In the first generations, the population is very noisy. This noise cannot be captured by the Bayesian network, and the log-likelihood hardly increases during the run. In the latter generations, the situation changes:

- In the successful runs with $\alpha = 0.25$ and $\alpha = 0.5$, the log-likelihood begins almost as low as in the first generations, but then increases rapidly by adding the first edges. This indicates that the population contains a structure which is captured by the edges added to the Bayesian network.

- For $\alpha = 1$ the log-likelihood at the beginning of the generations is much higher, which indicates less noise in the population (note that for an empty network the log-likelihood is $-N \sum H(X_i)$). The log-likelihood does not increase very much during the run. This indicates a premature convergence of the population. It has lost the structure that has been found in the other runs. Indeed, this run is not successful.

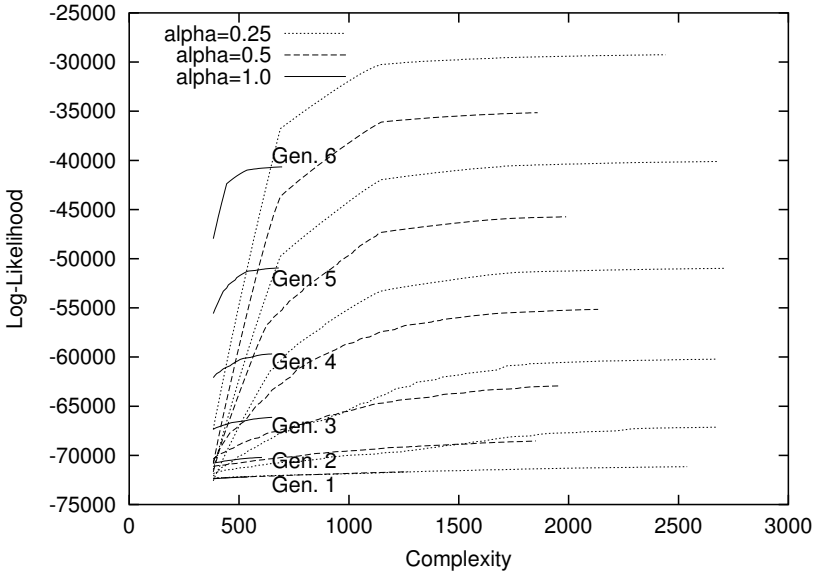

Figure 7.1.: Three runs of *LFDA* on Deceptive-5 with $n = 50$ for different values of α. $N = 7000$, $\tau = 0.3$. The trajectories of the first six generations in the MDL space are shown. The x axis is the complexity $\log N \sum 2^{|\Pi_i|}$, the y axis the log-likelihood $-N \sum H(X_i|\Pi_i)$.

We see that a too large value of α leads to premature convergence and loss of important structure. A too small value of α leads to a complicated Bayesian network structure; the marginals are difficult to estimate, and the algorithm becomes very costly. An analysis like the above can help to choose an appropriate α value.

7.3. Summary

In case that the dependency structure of the objective function is not known, and simple algorithms like *UMDA* fail, it is possible to learn a Bayesian network structure from the population data. This chapter presents methods to incorporate structure learning into *EDA*, using the principle of minimal description length. This leads to the *LFDA* and similar algorithms.

It was shown empirically how *LFDA* is able to recognize a dependency structure from data. When the population size approaches infinity, the correct *FDA* structure minimizes the BIC/MDL score. The influence of the structure penalty parameter α was analyzed. The effects of too large α (premature convergence) and too small α (overfitting) were investigated.

8. Combination With Local Hillclimbing

For many real-world problems, evolutionary optimization algorithms can run into difficulties because of the large search space. In order to reduce the search space, they have been combined with local hillclimbing. To each generated individual, a local hillclimber is applied. These hybrid algorithms (a combination of evolutionary optimization and local search) have been called *memetic* [HKS04].

A simple example is an algorithm that tries to improve the fitness with single bit flips, until a local optimum is reached. We emphasize that the term "local" in this context always refers to a neighborhood. E. g. for this simple hillclimber, the neighbors of a bit vector are all vectors that can be reached by single bit flips.

The problem of this simple algorithm is that is cannot traverse valleys in the fitness landscape. That means, it cannot flip two or more bits at the same time in order to improve the fitness. The most common approach for this problem is to increase the number of possible bit flips per iteration k. But this method is exponential in k. So a more sensible extension of the neighborhood is needed.

This chapter presents the Kernighan Lin hillclimber which was originally conceived for graph bipartitioning and TSP only [KL70, LK73, JM97]. We formulate it more generally, so it can be applied to arbitrary bit-string problems.

An iterated version [LMS02] of the Kernighan Lin hillclimber is presented, which is a simple yet surprisingly successful optimization algorithm. Then, Kernighan Lin is combined with *EDA*. The notions of information theory are used to analyze the results.

This chapter is organized as follows:

First we describe local hillclimbers, namely the simple bit-flip hillclimber and Kernighan Lin. Then local search algorithms using these hillclimbers are presented, as well as the hybrid algorithms which combine *EDA* with local search.

The next section is dedicated to the MAXSAT problem. First the problem is defined, and some design issues of the combination with evolutionary computation, as well as related work and SAT-specific hillclimbers are described.

Then the benchmark suite is presented. The difficulty of MAXSAT instances differs very much. We introduce the set of instances that we consider and choose particular instances for deeper analysis. We analyze the set of local maxima of the Kernighan-Lin algorithm and look for structure in this space, using methods of information theory. From this analysis we judge whether *EDA* is fit for this problem. Finally, the results of the iterated local search and hybrid algorithms on the benchmark problems are discussed.

The next section gives the results of the hybrid algorithms on the Ising spin glasses of Chapter 6. They are compared with the results of *BKDA*.

Finally, the third benchmark problem, the Kaufmann (n, k) problem, is introduced and then tackled by the algorithms. All algorithms presented in this thesis are applied

on this problem, and their behavior is discussed.

8.1. Local Hillclimbers

8.1.1. The Simple Hillclimber

The Simple Hillclimber performs single bit flips that improve the fitness. There are two common variants: Either the first bit flip with better fitness that is found is performed (greedy ascent), or all n bits are tried, and the best bit flip is accepted (steepest ascent), until no more improvement is possible.

Algorithm 8.1 implements the latter variant. In all algorithms $\mathbf{x}_{-j}$ is defined as $\mathbf{x}$ with bit x_j flipped.

Algorithm 8.1: Simple Hillclimber

```
1  Input: x
2  do {
3      j* ⇐ argmax_{j∈{1,...,n}} f(x_{-j})
4      if f(x_{-j*}) > f(x)
5          x ⇐ x_{-j*}
6  } until no more improvement
```

This hillclimber can be extended by allowing more than one bit flip per iteration. This increases the neighborhood of the search. But usually the neighborhood cannot be larger than two or at most three bit flips. Otherwise the search would quickly become too expensive.

8.1.2. The Kernighan-Lin Hillclimber

The Kernighan Lin hillclimber is an algorithm that can traverse large valleys in the fitness landscape. It was introduced in specialized forms for the graph bipartitioning [KL70] and traveling salesman problem [LK73, JM97]. Now, the basic idea will be generalized to arbitrary bit-string optimization problems.

In one pass, the algorithm flips bits which maximize the fitness among all possible bit flips (even if the new fitness is worse than the one we began with). A bit cannot be flipped twice in one pass, because that might result in a cycle.

This is done until a maximal number of bit flips ("maxflips") is reached. This maximal number of bit flips is the maximal size of a valley that can be traversed. Most commonly, we have set it to n, but if this is too slow, the value can be reduced.

We have now generated a series of vectors, each one bit flip away from its predecessor. From these, we finally choose the vector which maximizes the fitness, but only if it is better than the fitness we began with. Otherwise, the algorithm has reached a local optimum and stops.

A pseudocode description is given in algorithm 8.2.

Algorithm 8.2: *KLH−* Kernighan Lin Hillclimber

```
1  Input: x
2  do {
3     Mark all bits unflipped
4     for i = 1 to maxflips do {
5        j* ⇐ argmax_{j∈{1,...,n} unflipped} f(x_{−j})
6        x_i ⇐ x_{−j*}
7        Mark j* flipped
8     }
9     x* ⇐ argmax_{x_i} f(x_i)
10    if f(x*) > f(x)
11       x ⇐ x*
12 } until no more improvement
```

The Kernighan Lin algorithm is deterministic, it does not use random variables.

8.1.3. Complexity of Kernighan Lin

In the first experiments using the Kernighan Lin hillclimber, we have found that many standard benchmark problems for *EDA* pose no difficulty for *KLH*. Deceptive functions or the Iso-Circle are solved easily. This is due to the simple structure and symmetry of these functions: For example, many of these functions have the global optimum at $(1, 1, 1, \ldots, 1)$ and a local optimum at $(0, 0, \ldots, 0)$. In this case, the Kernighan Lin hillclimber can jump from the local optimum to the global optimum in one pass, if maxflips $= n$.

Therefore, we have chosen more difficult benchmark functions. First we will apply *KLH* on difficult MAXSAT instances. Then, we will investigate its performance on the Ising spin glass problem (see Sect. 6.1.1), comparing it to *BKDA*, and on the Kaufmann function, which has a random dependency structure.

The complexity of the Kernighan Lin hillclimber is $O(n \cdot \text{maxflips} \cdot \text{passes})$. This assumes that the fitness evaluation of $f(\mathbf{x}_{-j})$ can be done in constant time.

For Ising spin glass problems, this is the case: We only have to consider the up to four couplings in which the bit j appears. The random MAXSAT and Kaufmann problem both have a random dependency structure, but the expected number of clauses (MAXSAT) or connections (Kaufmann) in which a variable X_j appears can be assumed to be bounded.

8.2. Optimization with Hillclimbers

There are various ways to use such a hillclimber for optimization. Here we first describe their use for local search. Then we present their combination with evolutionary

algorithms.

8.2.1. Random Start and Iterated Mutation KLH

We examine two variants of local search: *Random Start KLH (RSKLH)* and *Iterated Mutation KLH (IMKLH)*.

RSKLH generates randomly a starting point and runs *KLH* on it. This is repeated G times.

IMKLH is similar, but instead of using a completely new random starting point, it applies mutation on the previously found local optimum. This technique is called iterated local search [LMS02]. R bit flips are performed, allowing repetition. After running *KLH* on the mutated individual, the better of the two is chosen for the next generation. If both have equal fitness, we choose the new one.

Algorithm 8.3: *IMKLH*– Iterated Mutation Kernighan Lin Hillclimber

```
1   Generate a random bit string x
2   Apply KLH on x
3   for g = 1 to G do {
4      y ⇐ x
5      for r = 1 to R do {
6         Choose a random i ∈ {1, ..., n}
7         Flip bit y_i
8      }
9      Apply KLH on y
10     if f(y) ≥ f(x)
11        x ⇐ y
12  }
```

In spite of its simplicity, this is a good heuristic algorithm. We find that it is much more successful than *RSKLH*. The parameter R should be chosen not too small (or the algorithm will have difficulties to move to a different optimum), but not too large either (or the algorithm will be essentially like *RSKLH*).

8.2.2. Memetic Algorithms: Evolutionary Optimization with Local Search

Whereas local search techniques embody a single random walker, evolutionary algorithms contain a population of points. The combination with local search has been called *memetic* or *hybrid algorithms* [HKS04].

The most canonic way to combine *EDA* and local search is to apply the local hillclimber on every generated point. The point is then replaced by the result of the hillclimber. This approach was followed here, too.

In contrast to this ("Lamarckian") variant there exists also another technique: The point is not replaced by the hillclimber result, but its fitness obtains the value of the hill-

climber outcome ("Baldwinian"). However, [HKS04] report that recent research prefers the Lamarckian approach. They also report on diversity preservance techniques, combination of different local searchers and ways to incorporate knowledge. These are several possibilities to extend the research.

8.3. The SAT and MAXSAT problem

The first benchmark function for this technique is the MAXSAT problem. First we give the definition of the problem, then an analysis of the search space for selected instances, and finally a comparison of the results for the different variants of *EDA* and *KLH*.

8.3.1. Definition of SAT

Definition 8.1 (SAT). An instance of the **SAT problem** is given by a binary formula in conjunctive normal form of the variables $x_1, \ldots, x_n$:

$$F(\mathbf{x}) = \bigwedge_{i=1}^{\gamma} C_i(\mathbf{x})$$

where γ is the number of clauses and

$$C_i(\mathbf{x}) = \bigvee_{j=1}^{l(i)} x_{k(i,j)}^{s(i,j)}$$

$l(i)$ is the number of literals in the clause C_i. $k(i,j) \in \{1, \ldots, n\}$ is the index of the variable, $s(i,j) \in \{0,1\}$ is its sign, defining

$$x_k^1 = x_k \qquad x_k^0 = \neg x_k$$

A solution of the instance is a vector $\mathbf{x}^*$ with $F(\mathbf{x}^*) = \text{true}$.
If $\max l(i) \leq 3$, we call the problem **3-SAT**.

8.3.2. SAT in Evolutionary Computation and MAXSAT

The straightforward coding of a SAT instance into an individual of evolutionary computation is to let each bit of the bit vector represent the truth value of one variable of the SAT problem. This is called the *bit-string representation*. Other representations were proposed, but turned out to be inferior [GMR02].

Next, we need to define a fitness function. The SAT problem is a "yes/no" problem: Either a formula is fulfilled or not. But a fitness function that has value "0" for unsatisfied and "1" for satisfied formulas will definitely give bad results, because this gives no indication how far away from a solution we are and in which direction to search next.

Therefore most commonly evolutionary computation does not treat the SAT problem, but the MAXSAT problem: The function to be maximized is the number of fulfilled clauses:

$$f(\mathbf{x}) = |\{C_i, i = 1, \ldots, \gamma | C_i(\mathbf{x}) = \text{true}\}| \tag{8.1}$$

Here we know that the optimum is $f_{\max} = \gamma$ (provided that the instance is satisfiable at all). When this value is reached, we know the instance to be satisfiable.

More sophisticated approaches include adaptive fitness functions, which guide the evolution according to information collected during the search, e. g. by emphasizing clauses that are difficult to fulfill. This was not pursued here.

The MAXSAT problem is a good benchmark for the combination with local search, because without the use of local search, it is very difficult for evolutionary algorithms. This is due to the huge and complicated search space, which can be reduced very much by constraining oneself to the local optima of a hillclimber. Also, the fitness landscape contains a lot of local optima. Suppose that only one clause is violated, so $f(\mathbf{x}) = \gamma - 1$. The evolutionary algorithm has no indication of how to fulfill this last clause. Local search helps on this matter.

8.3.3. SAT specific Hillclimbers: Walksat

In the community working on the MAXSAT problem, a number of hillclimbers have been crafted which are particularly fit for solving MAXSAT instances. An example is the simple algorithm used in [San04]. This hillclimber chooses an unsatisfied clause and then performs a bit flip which satisfies this clause.

A slight change to this hillclimber yields the algorithm GSAT [SLM92]: Here among the variables of the violated clause the one which gives the best fitness improvement is flipped. Then, the introduction of a noise parameter leads to the algorithm Walksat [SKC94, SKC95, Fuk04a]. Here with probability *noiseprob* (by default 0.5) a random variable of the violated clause is flipped, instead of the one with the best fitness improvement. The exact algorithm is given in Alg. 8.4.

Algorithm 8.4: Walksat

```
1  Input: x
2  do {
3      Choose randomly a violated clause C_i
4      if a variable in C_i can be flipped without violating another clause
5          flip such a variable (break ties randomly)
6      else
7          flip with probability noiseprob a random variable in C_i and with
           probability 1 − noiseprob one which leads to best fitness (break
           ties randomly)
8  } until no more improvement
```

This and other MAXSAT-specific elements can be combined in various ways, resulting in modern MAXSAT-solving algorithms [Fuk04b]. However, we are more interested in algorithms which are more generalizable, so that the concept can be used for other fitness functions, too. Kernighan Lin is such a general-purpose hillclimber.

8.3.4. MAXSAT Instances

We used for our analysis example instances of various sizes for 3-SAT which can be found in the Satlib [HS00]. The random instances found there are of size from 20 up to 250. Modern SAT-solving competitions use much larger instances; but for a first investigation, these instances are still widely used. It turned out that the instances of size 20 and 50 are very easy for our *KLH*. (Without *KLH* evolutionary computation already fails for the uf50 instances!) Therefore we have concentrated our analysis on the problems of size 100, 150, and 200.

The series uf100 consists of 1000 random 3-SAT instances of size $n = 100$ with $\gamma = 430$ clauses. To estimate their difficulty, 20 runs of *FDA* with *KLH* were performed on each. The histogram of success rates can be found in Fig. 8.1. It was found that most instances are very easy. With a very small population size of $N = 100$ and $\tau = 0.3$, 788 instances had a success rate of 100 %. Of these, 649 instances were always solved in the initial population (the points were generated randomly and then *KLH* applied to them – this is equivalent to *RSKLH* with $G = 100$ attempts).

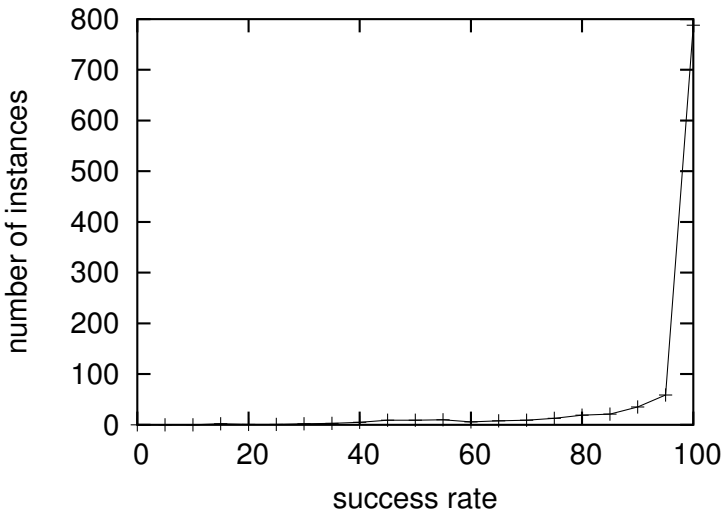

Figure 8.1.: Histogram of success rates for 20 runs of *FDA* with *KLH* on all 1000 instances of the uf100 series. Population size $N = 100$, truncation threshold $\tau = 0.3$.

These instances, which are obviously easily solved by *KLH* alone, are not interesting for our purposes. For deeper analysis, and for comparison of the different algorithms, the four instances given in Table 8.1 were chosen. The criteria for this choice were a high number of required generations and a varying success rate.

The same analysis was done for series uf150 with $n = 150$ variables and $\gamma = 645$ clauses. Here, a population size of $N = 200$ was used. Again, we found that most of the 100 instances were easy for *KLH*. 73 of them were solved in all 20 runs, of these 59 every time within the initial generation. The complete results on this data set are

Name	Success Rate	Average Generations
uf100-059.cnf	13 / 20	2.69
uf100-0129.cnf	20 / 20	2.45
uf100-0160.cnf	12 / 20	1.33
uf100-0285.cnf	8 / 20	3.25

Table 8.1.: Result of *FDA* with *KLH* on some instances of uf100. $N = 100$, $\tau = 0.3$.

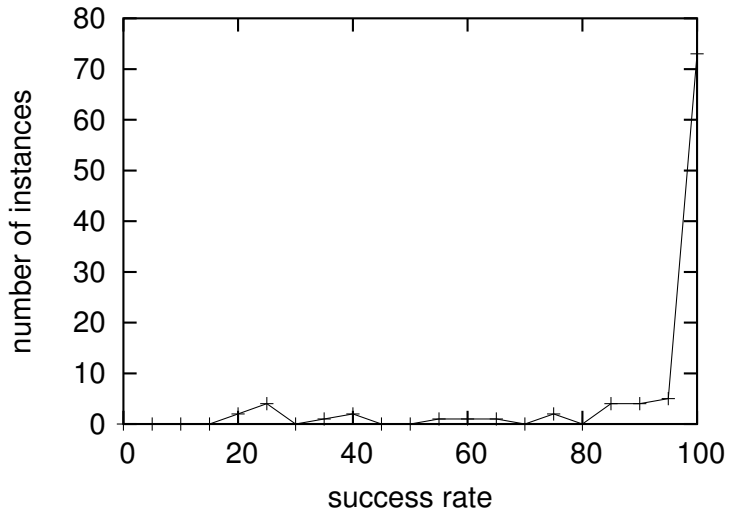

Figure 8.2.: Histogram of success rates for 20 runs of *FDA* with *KLH* on all 100 instances of the uf150 series. Population size $N = 200$, truncation threshold $\tau = 0.3$.

depicted in Fig. 8.2.

Our goals are the following:

- An analysis of the space of local optima of the *KLH* and
- Investigating whether the combination of *EDA* and local hillclimbing is worthwhile.

8.3.5. The local optima of KLH

We have chosen some well-suited instances from the uf150 series ($n = 150$), which are not too simple for the *KLH*. For the local optima analysis, we have performed 10000 runs of *KLH* on instance uf150-028, starting from initial random values. The results are paradigmatic for typical, not too simple MAXSAT instances.

Figure 8.3 shows a histogram of fitness values of these 10000 runs. We see that on this instance, *KLH* is able to find a solution (all 645 clauses fulfilled), but succeeds only in 13 of the 10000 runs. It is interesting to note that no two of the 10000 local optima are the same. So we see that the space of local optima is still very large.

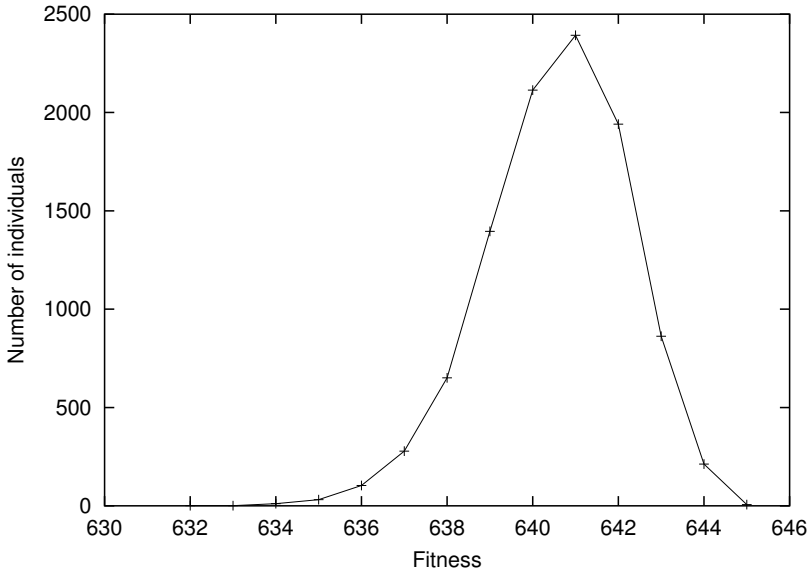

Figure 8.3.: Histogram of fitness values for 10000 runs of *KLH* on uf150-028.

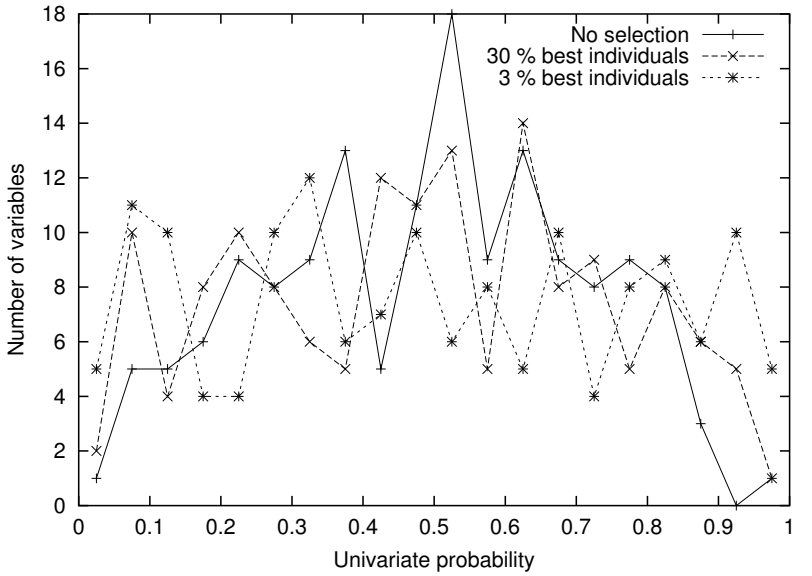

Figure 8.4.: Histogram of univariate probabilities for the 150 variables on uf150-028, averaged over 10000 runs of *KLH* (solid line). The dashed and dotted line show the same histogram, averaged over the 30 % and 3 % best individuals, respectively. Histogram bin width is 0.05.

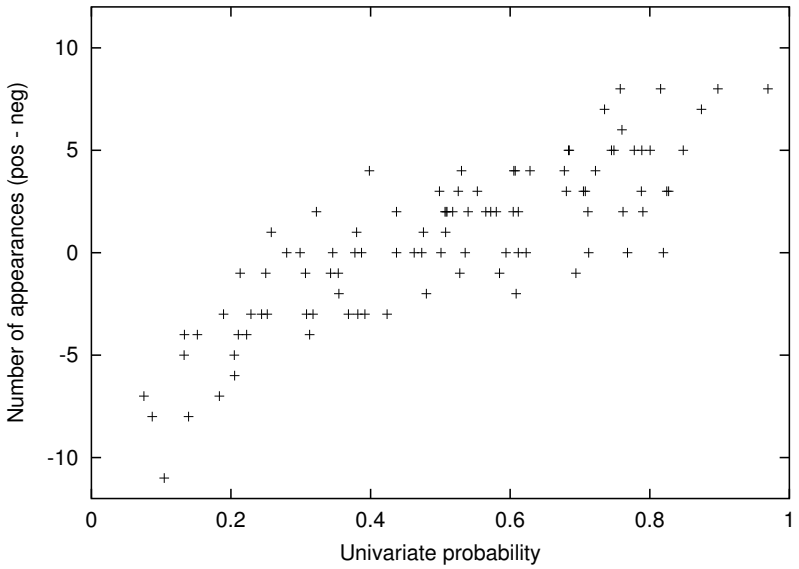

Figure 8.5.: Univariate probability of the 150 variables against number of appearances in the formula, where negated appearances count negative. 10000 runs on uf150-028.

We can regard the 10000 bit strings as a sample from a probability distribution over $\{0,1\}^n$. But care must be taken, since the sample size is quite small, as opposed to the space size $2^{150} \approx 10^{45}$. First we investigate the univariate marginal distributions. Figure 8.4 shows a histogram of the univariate probabilities for the 150 bits. We see that some bits are almost fixated (near 0 or 1), but most are not.

If we generate the same histogram, but use only the best individuals (performing truncation selection), we find that more variables have a probability near 0 or 1, but the effect is not very strong.

Figure 8.5 shows the connection between appearance of the variable in the formula (negated or non-negated) and its univariate probability within the *KLH* maxima. Against the univariate probability, we plot the number of positive occurrences minus the number of negated occurrences of the variable. We notice a clear dependence between these.

Next, we consider the bivariate marginal distributions. Of these, we show the mutual information between the variables (4.10). A histogram of these values for all combinations of variables is depicted in Fig. 8.6. It is separated in variables which appear together in a clause and variables which do not. We see that variables that share a clause tend to have a higher mutual information.

If we want to know whether the combination with *EDA* is promising, we should investigate whether selection of the best values has any effect.

On the univariate distributions, quite surprisingly, the effect is rather small. Truncation selection reduces the sample size, but other than that, there is hardly an effect to be seen. But since the space of local optima is still large, the univariate marginal

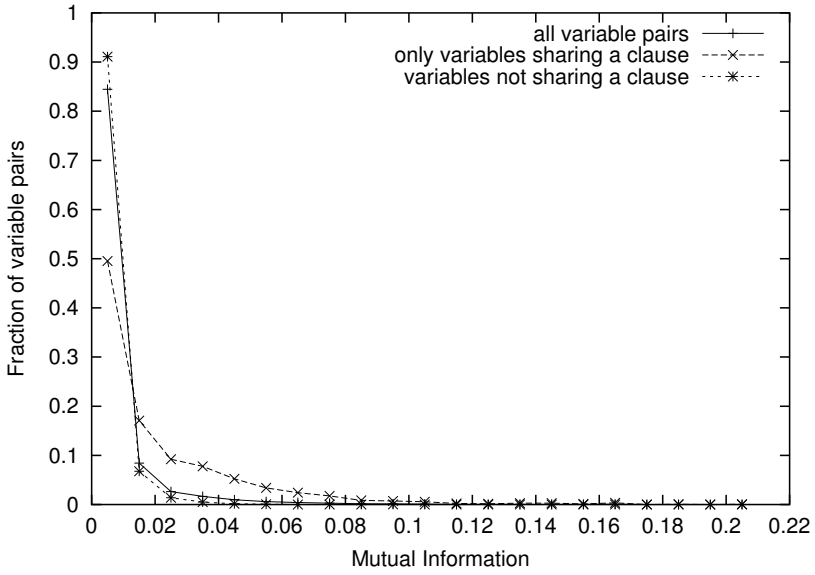

Figure 8.6.: Histogram of mutual informations for all $\binom{150}{2} = 11175$ pairs of variables. The dashed line is the histogram only for variable pairs that appear together in a clause (1779 pairs), the dotted line for those that do not (9396 pairs).

probabilities are not very expressive.

Fig. 8.7 shows for the population the minimum Hamming distance to one of the found optima, plotted against the fitness of the individual. Here we see that the individuals of low fitness values are all far away from an optimum. This indicates that it is a good idea to throw these away and go for the best ones, which have a better chance to be near to an optimum. This is an argument in favor of selection.

8.3.6. Cluster analysis

Evolutionary algorithms are most successful when the maxima are concentrated in a relatively small area and not randomly distributed. If they are evenly distributed in the fitness landscape, the assumption that "good values lead to better ones" does not hold.

Therefore we investigate the clustering of the MAXSAT solutions found by *KLH*. For clustering binary data, it has been found [Wil87, WD03] that Jarvis-Patrick Clustering [JP73] gives the best results. It has also the advantage that it does not need to calculate the center of a cluster (like the k-means algorithm), or for other reasons the average of a set of points, which gives questionable results on binary data[1].

The Jarvis-Patrick clustering algorithm takes two parameters, J and K. It consists of the following steps:

[1] It would at least require to extend the distance measure from binary, discrete data to the unit hypercube, which raises the question of which distance measure to use.

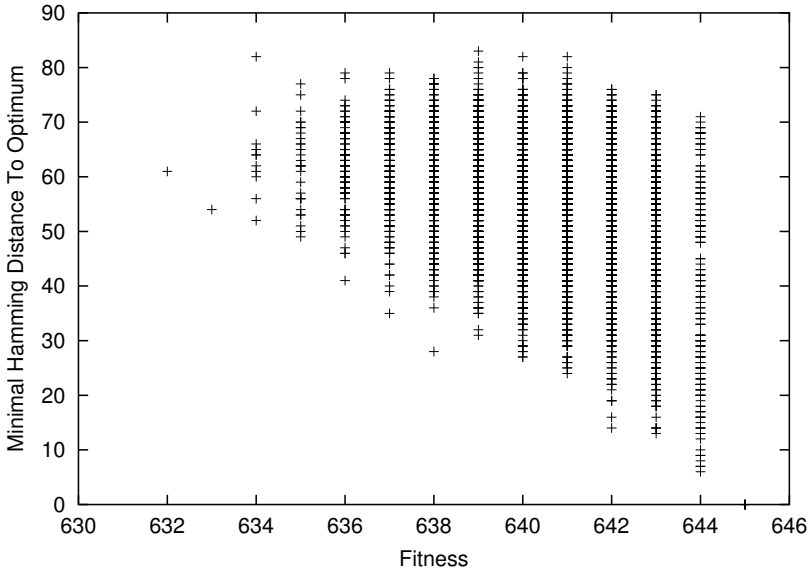

Figure 8.7.: Fitness and hamming distance to nearest optimum for the 10000 solutions

- For every data point, calculate the list of the J nearest neighbors of this point.
- Place two points in the same cluster if
  - they are in each other's neighborhood list and
  - their neighborhood lists have at least K points in common.

This algorithm was applied on the 30 solutions for uf150-028.cnf depicted in Fig. 8.8. It found $(J = 8, K = 3)$ that three solutions (17-19) form a cluster which has a hamming distance of 56 bits to the other solutions. It is also possible $(J = 8, K = 4)$ to divide solutions 1-16 from 20-30, but these clusters are only 5 bits apart.

The clustering indicates the so-called "Himalaya effect": The solutions are all concentrated in very few regions and not spread equally all over the search space. This is one

Figure 8.8.: Thirty solutions for uf150-028.cnf found by *KLH*. One row is one solution, one column is one bit, black bits are 1, white 0. Solutions are ordered lexicographically. It can be seen very well that solutions 17-19 are far away from the others.

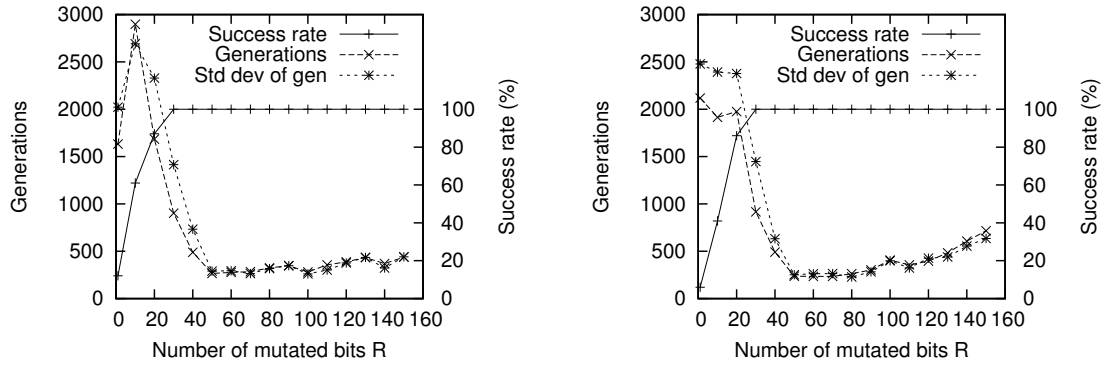

Figure 8.9.: Success rate and number of generations needed by *IMKLH* on problems uf100-059 (n=100) and uf150-028 (n=150) for different values of R (the number of bit flips for every restart). Mean over 100 runs.

of the basic axioms of evolutionary computation. Therefore, it is a good idea to work with probability distributions in the solution space.

8.3.7. Evaluation of IMKLH and RSKLH

The iterated mutation KLH (*IMKLH*) presented in Sect. 8.2.1 contains a parameter R, the number of bit flips after each run of KLH.

Figure 8.9 shows the success rate and the number of generations required until the optimum is found, averaged over 100 runs, for varying R. It can be seen that the algorithm fails for too small R, reaches its best performance with R between 50 and 80, and above this value gets worse again.

Very notable is the large standard deviation. Often it is even bigger than the average. This shows that the algorithm is unstable; it can get stuck in a local optimum and take a long time to find a global optimum.

The results of *RSKLH* on instance uf150-028 have already been shown in Fig. 8.3. To compare it with the results of *IMKLH*, we note that the number of runs until success is geometrically distributed with

$$p_{\text{geo}}(k) = (1-p)^k p \tag{8.2}$$

Given a success probability p, this is the probability of k failures before the first success. The expected value and standard deviation of the geometric distribution are

$$\mu(p_{\text{geo}}) = \frac{1-p}{p} \qquad \sigma(p_{\text{geo}}) = \frac{\sqrt{1-p}}{p} \tag{8.3}$$

From Fig. 8.3 we estimate $p = 0.0013$, which gives $\mu = 768.23$ and $\sigma = 768.73$. We note that the estimate acquired by 13 successes on 10000 runs is rather unstable, because of the small number of successes but it coincides with the limit of the curve in Fig. 8.9

for large R. The required generations are more than double the results of *IMKLH* with $R = 50$ or 80.

8.3.8. Iterated KLH and Walksat

We now present results of iterated local search on some more 3-SAT instances. In Table 8.2 *IMKLH* and Walksat are compared.

For Walksat the implementation by Henry Kautz and Bart Selman (Walksat version 45)[2] was used. This is an efficient, actively maintained implementation in C. It contains several different search heuristics, but we used it "out of the box", i. e. with the default parameters.

	IMKLH				Walksat		
Instance	SR	KL calls	stddev	phases	SR	flips	stddev
uf100-059	20	199.8	194.3	610.5	100	11953	12194.7
uf100-0129	20	166.4	138.9	506.8	100	7664	6777.1
uf100-0160	19	406.8	309.6	1205.6	100	8928	8525.9
uf100-0285	20	495.2	443.5	1418.6	100	19993	15730.6
uf150-028	20	196.9	181.0	679.8	100	11389	9387.9
uf200-01	20	1444.2	1130.6	5334.6	89	29843	40771.3
uf200-057	20	45.3	40.6	165.3	100	4665	3400.7

Table 8.2.: Results of *IMKLH* and Walksat on some 3-SAT instances. Given is the number of successful runs (SR), the number of KLH calls until success with its standard deviation and the average number of KLH phases; for Walksat the number of successful runs (SR), the average number of bit flips and its standard deviation. Parameters: *IMKLH* 20 runs, mutated bits $R = 50$ for $N = 100$, 80 for $N = 150$ and 100 for $N = 200$. Walksat 100 runs, noise probability 0.5.

It is not easy to compare the two local optimizers, because they work somewhat differently. Concerning the runtime, Walksat v45 is of course much faster than our C++ implementation of *KLH*. A C implementation, using C arrays instead of a binary variable class, and some intricate optimizations [Fuk04a] make Walksat v45 very efficient.

First we note that both techniques are very successful in finding optima of the MAX-SAT problems. Only once *IMKLH* failed to find a solution in 2000 generations, and in another instance Walksat succeeds only in 89 % of the runs.

IMKLH needs less KLH phases than Walksat needs bit flips. But if we remember that in every KLH phase maxflips $= n$ bit flips are attempted whereas Walksat considers only the 3 variables of a clause in each generation, we see that Walksat is better, especially for the $n = 100$ instances. This is no surprise, since Walksat is a specialized SAT solver and exploits better the problem structure. *IMKLH* can be speeded up by choosing a smaller value of maxflips. But we see that both algorithms perform in a similar scale.

[2] Available at http://www.cs.washington.edu/homes/kautz/walksat

	uf100-0129				uf100-059			
Algorithm	succ	gen	sdv	KLH	succ	gen	sdv	KLH
RSKLH	20/20	1.85	1.755	568.15	19/20	1.79	2.016	556.11
UMDA +KLH	20/20	0.95	0.686	389.05	12/20	0.83	0.835	365.83
FDA +KLH	20/20	0.45	0.605	289.55	12/20	1.17	1.267	432.17
LFDA +KLH	20/20	0.85	1.089	369.15	16/20	0.88	0.806	374.13
FDA j6+*KLH*	19/20	0.79	0.535	357.11	18/20	1.72	1.809	542.72
MEFDA +KLH	20/20	0.80	0.616	359.20	15/20	1.47	1.959	491.87

Table 8.3.: Comparison of the different hybrid algorithms (*EDA +KLH*). Given is the number of successful runs, the number of generations and its standard deviation, and the average number of *KLH* runs performed (including unsuccessful runs). $N = 200$, $\tau = 0.3$, break after 10 generations. The algorithms are random start *KLH*, *UMDA*, *FDA*, *LFDA* ($\alpha = 0.5$), *FDA* with join (maximal join size 6) and *MEFDA* (maximal join size 6).

	uf150-028 ($N = 200$)				uf200-01 ($N = 1500$)			
Algorithm	succ	gen	sdv	KLH	succ	gen	sdv	KLH
RSKLH	15/20	4.13	2.20	1314	9/20	2.89	2.03	9220
UMDA +KLH	16/20	2.06	1.12	926	5/20	1.20	0.45	9819
FDA +KLH	20/20	2.55	1.47	707	15/20	1.87	0.64	6222
LFDA +KLH	20/20	1.65	0.99	528	18/20	1.94	0.73	5173
FDA j6+*KLH*	20/20	2.40	1.27	678	13/20	1.69	0.75	6821

Table 8.4.: Comparison of the different hybrid algorithms (*EDA +KLH*) on instances uf150-028 (with population size $N = 200$) and uf200-01 (with $N = 1500$). 20 runs each. Values and parameters like in Table 8.3, except for uf200_01: break after 7 generations.

The standard deviations of both algorithms are quite high. Sometimes they are even larger than the mean. In this respect the local search methods are less stable than evolutionary algorithms, which commonly have a standard deviation of about one generation.

8.3.9. Results of FDA with KLH

The results of the hybrid algorithm of *FDA*, applying *KLH* on every generated point, is shown in Tables 8.3 and 8.4. We give also the number of calls to the KLH, because this is what takes most of the running time. For *FDA*, this is N for the initial population and $N - 1$ for the following generations, because being elitist, we do not touch the best solution so far. This gives $N + g(N - 1)$, where g is the number of generations needed. (If the initial population contains the solution, we set $g = 0$.)

We can see in the tables that the combination of *EDA* and *KLH* needs less calls to

KLH than *RSKLH* alone. Of the different *EDA* variants, it seems that *FDA* with join and *LFDA* perform better than conventional *FDA* and *UMDA*, but the difference is not very significant. The reason for this is that MAXSAT is a rather strongly connected problem. If a 3-SAT formula contains $\gamma \approx 4.3n$ clauses, this means it has $3\gamma \approx 13n$ literals. Each variable appears on average in 13 clauses, so it can be connected with up to 26 other variables.

This calculation indicates why *EDA*s with complicated structures – particularly *LFDA* which selects the most important connections – perform better. But the small difference between the different *EDA*s indicates also that most of the work is done by the Kernighan Lin hillclimber.

IMKLH performs better than the *EDA*s. It has a large success rate and needs few KLH calls, although with a large standard deviation. This is largely because *IMKLH* does not depend on a population size. Its number of required generations is often smaller than the population size needed for the *EDA*s.

8.4. Results of FDA and KLH on Ising Spin Glasses

For comparison, we also apply *FDA* with KLH on the spin glass instances of Sect. 6.5.5 and 6.5.6.

	FDA			*FDA* join			*MEFDA*		
Seed	SR	Gen	Sdv	SR	Gen	Sdv	SR	Gen	Sdv
1	100	0.810	0.706	100	0.730	0.709	99	0.737	0.737
2	99	0.303	0.524	94	0.383	0.551	93	0.376	0.550
3	100	0.030	0.171	100	0.010	0.100	100	0.040	0.197
4	100	0.180	0.479	100	0.150	0.386	100	0.180	0.386
5	98	0.673	0.797	98	0.867	1.090	92	0.891	1.153
6	100	0.070	0.256	99	0.121	0.358	100	0.180	0.411
7	100	0.000	0.000	100	0.000	0.000	100	0.020	0.141
8	100	0.690	0.775	99	0.444	0.673	100	0.480	0.627
9	96	0.385	0.587	98	0.388	0.586	98	0.418	0.824
10	100	0.150	0.386	99	0.131	0.339	100	0.170	0.378

Table 8.5.: Results of *FDA* with *KLH* on the 10×10 Ising instances of Sect. 6.5.5 and 6.5.6. Given is the number of successful runs, the generations and the standard deviation. $N = 100$, $\tau = 0.3$, 100 runs each.

Table 8.5 shows that the small instances pose no difficulties for *KLH*. All instances of size 10×10 are easily solved with a population of 100 points, almost always in the initial population, without even applying the evolutionary paradigm. Therefore the chosen flavor of *FDA* does not matter.

This result allows us to use for the 15×15 Ising instances only the *FDA* variant which was found optimal in Sect. 6.5.6. The results in Table 8.6 show a different picture. These

Seed	SR	Gen	Sdv	KL succ	phases
1	20	3.00	0.79	1197	3606
2	18	3.72	0.83	1413	3628
3	15	6.53	1.19	2253	5006
4	6	4.17	0.75	1546	4071
5	20	3.15	0.93	1242	3610
6	19	3.74	0.99	1417	4168
7	20	3.30	1.08	1287	3544
8	20	4.60	1.67	1675	4345
9	11	5.36	1.03	1904	4295
10	16	3.38	1.50	1309	3765

Table 8.6.: Results of *FDA* with pentavariate grid and *KLH* on the 15×15 Ising instances of Sect. 6.5.5 and 6.5.6. $N = 300$, $\tau = 0.3$, 20 runs each. Given is the number of successful runs, the number of generations and its standard deviation, and the number of runs and phases of *KLH* on the successful runs.

instances are not easy any more. For a *KLH* run the population size 300 is quite large and computationally expensive. Nevertheless some instances had a low success rate, and all needed between 3 and 7 generations on average.

It is interesting to note that the instance rh15_8 which was difficult in Table 6.7 and 6.11 was easily solved by *FDA* with *KLH*, whereas the instance rh15_4 is difficult for *EDA* but was solved by *BKDA* (see Table 6.8).

8.5. The Kaufmann Problem

8.5.1. Definition of Kaufmann

The Kaufmann (n, k) fitness function is an ADF with a completely random structure. It takes two parameters:

- n is the number of variables, but also the number of subfunctions.
- k is the dimension of the subfunctions.

A subfunction f_i, $i = 1, \ldots, n$, is generated by connecting the variable X_i with $k-1$ randomly chosen other variables $X_{j(i,\kappa)}$, $\kappa = 1, \ldots, k-1$. (Of course, repetitions of a variable are not allowed.) This way of building the structure ensures that every variable appears in at least one subfunction. The 2^k function values of such a k-variate subfunction are chosen randomly within the interval $[0; \frac{1}{n-k+1}]$.

The Kaufmann function is the sum of these subfunctions:

$$F_{\text{Kauf}}(\mathbf{x}) = \sum_{i=1}^{n} f_i(x_i, x_{j(i,1)}, \ldots, x_{j(i,k-1)}) \ . \tag{8.4}$$

Since the function values are continuous, the Kaufmann function has many more different local optima than MAXSAT. Whereas in MAXSAT there are generally many points with fitness $\gamma - 1$, for Kaufmann it is very unlikely that two local optima have the same fitness value.

8.5.2. Results of the Algorithms on Kaufmann

Tables 8.7 and 8.8 give the results on example Kaufmann (n, k) instances with $k = 3$ and $n = 200$ or $n = 400$, respectively.

Algorithm	SR	Avg fitness ± stddev	seconds
FDA +join, $N = 30000$	18	0.755829 ± 4.74807e-05	3124
FDA +join+*KLH* 20, $N = 100$	13	0.755755 ± 0.000150023	313
IMKLH 20, $R = 75$	12	0.755771 ± 8.75812e-05	1347
GBP 50gen, $N = 30000$, $\beta = 1000$	0	0.749058 ± 0.00143068	480

Table 8.7.: Results on Kaufmann (200,3). 20 runs each. SR gives the number of times that the best known point (fitness 0.755841) is found. Furthermore we give the average fitness over the 20 runs with its standard deviation and the total runtime of the 20 runs in seconds. The parameters are $\tau = 0.3$ for *FDA*, maxflips = 20 for *KLH*.

Algorithm	SR	Avg fitness ± stddev	seconds
FDA+join, $N = 30000$	3	0.752556 ± 0.000603656	13168
FDA+join+KLH20, $N = 100$	7	0.752997 ± 0.000518414	1464
IMKLH 20, $R = 100$	15	0.753165 ± 0.000747055	3088
CCCP 50gen, $N = 30000$, $\beta = 5000$	0	0.752291 ± 0.00183455	2229

Table 8.8.: Results on Kaufmann (400,3), same as in Table 8.7. For CCCP, one outer loop update after 5 inner loop iterations. The best point known for this instance has fitness 0.753505.

It can be seen that loopy belief propagation is less successful than the other methods. This is probably due to the bad structure of the region graph, automatically built using CVM. It was postulated in [YFW04] that a good region graph should have the sum of all counting numbers 1 (see p. 115). The region graphs used here have -154 for $(200, 3)$ and -161 for $(400, 3)$. Another problem is the choice of β. For small β the distributions are badly focused and therefore the fitness values are rather low, whereas for too large values the method becomes numerically unstable.

The other methods – *FDA* with subfunction join and a large population size, *FDA* with join and *KLH*, and *IMKLH*– are all more or less successful on this benchmark. Their average reached fitness is comparable. The runtime[3] should be regarded with care. It is very dependent of the parameters, e. g. the maximal number of generations

[3]Measured on an Intel Pentium M processor with 1.5 GHz.

and the stopping criteria, so it can only give a rough idea of the required effort. But it can be seen that the hybrid algorithm of *EDA* and *KLH* is the cheapest.

8.6. Summary

This chapter investigates the combination of *EDA* with local search. It is easily possible to incorporate a local hillclimber into *FDA* and its variants. We present the Kernighan-Lin hillclimber, which is for the first time formulated as a general-purpose technique.

It can be used in a stand-alone manner, as iterated local search, or combined with *EDA*. We investigated these variants using three benchmark problems: MAXSAT, Ising spin glasses, and Kaufmann (n, k). An analysis of the *KLH* local optima of MAXSAT sheds light on the properties and the behavior of *KLH*.

We have found that the iterated local search algorithm *IMKLH* is a good search heuristic, considering the simple design. The combination with *EDA* is similarly successful. The chosen flavor of *EDA* seems to be of minor importance.

The hybrid algorithm performs quite well on most regarded benchmark problems. For the grid-like Ising spin glass, *BKDA* is more efficient. But for the random-structure Kaufmann problem, the hybrid algorithm of *EDA* and *KLH* seems to be the most efficient.

9. Conclusion

9.1. Results and Conclusions

This thesis uses methods and notions from statistics, information theory and statistical physics to devise improvements and gain insights on the field of estimation of distribution algorithms (*EDA*). *EDA* is a class of evolutionary optimization algorithms which builds a probabilistic model in the search space from the selected population and uses this model to sample the next population.

- For the factorized distribution algorithm (*FDA*) we present a new method to build a probabilistic structure (a factorization) from a given additive structure of the objective function. By virtue of *merging subfunctions*, more dependencies between the variables can be exploited. The more powerful model leads to significant reduction in the required population size of *FDA*.

- The more complicated structure built by subfunction merge is also more difficult to estimate. The *maximum entropy principle* permits to reduce the noise in the distributions of the merged model. This leads to the maximum entropy *FDA* (*MEFDA*) which is favorable particularly for small populations where estimates are noisy.

- *Loopy probabilistic models* and the Kikuchi approximation constitute an extension of the applicable model class. Generalized belief propagation (GBP) learns a probabilistic model which approximately minimizes the relative entropy to the Boltzmann distribution of the objective function. In the Bethe Kikuchi distribution algorithm (*BKDA*) this is combined with a factorization in order to draw samples efficiently. On the Ising spin glass problem, this algorithm solves small instances (7×7) exactly. Larger instances (10×10 and 15×15) do not focus on the optimum, but can be solved using a reasonable sample size.

- The *inverse temperature* β of the Boltzmann distribution plays a vital role in this algorithm. If it is chosen too low, the optimum does not stand out sufficiently in the probability distribution; if it is too high, convergence is critical, numerically less stable, and the probability to sample the optimum is small, too. In some cases, the concave-convex procedure (CCCP) can replace generalized belief propagation to improve convergence.

- The relative entropy between a population distribution and a postulated model distribution lays the foundation of the BIC/MDL measure for *learning a model*

from population data within the learning *FDA* (*LFDA*). It can recognize the correct structure of the fitness function, if the population is sufficiently large. The mutual information of the variables is a good indicator of their dependency. An analysis of the BIC/MDL space gives insight in the behavior of the learning algorithm.

- The Kernighan Lin hillclimber is a powerful local search heuristics which can be applied in a stand-alone manner, as iterated local search, or within a hybrid algorithm, combined with evolutionary optimization. Analysis of *KLH* optima gives some indications why the combination of *EDA* and *KLH* is favorable.

  The MAXSAT problem is very densely connected and has only few distinct fitness function values. This makes it difficult for *EDA*, and therefore *EDA* plays only a minor role when solving this problem; most of the work is done by the *KLH*.

  The Ising and Kaufmann problems are different. Here the hybrid algorithm is successful, but not necessarily better than other *EDA* variants.

9.2. Outlook

Since *EDA* is combined with methods of information theory, it is advisable to use these methods also to estimate the dynamics of the algorithms, in order to understand how and why they work best. This has been demonstrated particularly in Chapter 7 (for learning a Bayesian network in the context of *EDA*) and in Chapter 8 (for the combination with local hillclimbing). Further analysis on this track should lead to interesting insights.

The subfunction merge algorithm (Alg. 5.1) yields a factorization which accounts for all variable connections. But often (e. g. for the SAT problem) the variables are too densely connected, and Alg. 5.1 produces an infeasible structure. In this case, connections were removed randomly, which is probably not the best solution.

A better criterion for the choice of the dependencies to be kept could be the deviance from linear regression of the variables in the fitness function. Variables which depend almost linearly in the subfunctions can be disjoined in the factorization without much information loss. Another possibility is to choose the most important connections using an algorithm similar to *LFDA*, but confined to the edges of the true dependency graph. Such a criterion for choosing dependencies could enlarge significantly the class of feasible problems for the subfunction merge algorithm. Another question is whether the merge algorithm, as well as *MEFDA* (using maximum entropy to estimate large distributions from smaller ones) could be applied fruitfully on *EDA*s which deduce the probabilistic structure from the population, like *LFDA*, *EBNA* or *BOA*.

A similar possible direction of future research applies to the Kikuchi approximation. Santana [San05] presents an algorithm to learn a region graph from a selected population; however, he does not perform generalized belief propagation, but estimates the marginals from the population and then applies Gibbs sampling on this structure. It is possible that on a region graph learned from population data GBP will give a poor result. Nevertheless it is an interesting and important research question whether *BKDA* and its sampling

technique can be combined with a region graph learning algorithm, in order to broaden its field of applicability.

Another important question is: What can we do if the optimum is not probable enough in the distribution learned by GBP? We have identified β as an important parameter with strong influence on this probability. Furthermore, if different region graph structures are possible (e. g. by rotating or mirroring the pentavariate grid), they often yield very different results, too. The CCCP algorithm is suitable when GBP has convergence difficulties, but the fixed point to which it converges is the same. Another possibility could be to devise a variant of the Gibbs free energy which leads to better fixed points (similar to BIC/MDL, which adds a structure penalty to the log-likelihood).

Our research on *LFDA* is just a first step. There are many possibilities for further research on this avenue, also for further information-theoretical analysis. Also, apart from the simple greedy hillclimber used in *LFDA*, many more complicated algorithms have been proposed for learning Bayesian networks from data. These could be used in *EDA*, too. The first step should be to allow removing or reversing edges in the learned graph. Another possibility is to take advantage of the previous generation's model instead of rebuilding the structure from scratch in every generation. Finally, *LFDA* uses only the distribution of the selected population to build the model structure, but the fitness values of the points are not exploited. It could be fruitful to take these into account, too.

The Kernighan Lin hillclimber is a very successful local search procedure. It outperforms single or multiple bit-flip hillclimbers, but it is also computationally more expensive. It would be interesting to investigate how to make it more efficient. Usually, *maxflips* (the maximal number of bits to be flipped in one phase) can be set to $n/2$ or less without severe performance loss. Another possibility is to consider only a subset of bits for flipping, possibly in a randomized manner. Is it possible to reduce the asymptotic running time of *KLH* without jeopardizing its good performance?

Finally, the presented methods for estimating probability distributions can be applied not only to optimization. For example, in [MH02b] we have applied maximum entropy to the stochastic analysis of cellular automata. Other disciplines could profit as well from an interdisciplinary research effort, which avoids "reinventing the wheel", i. e. solving problems which have already been tackled in other disciplines.

9.3. Choice of an Optimization Algorithm

Given an optimization problem, which algorithm should be used to solve it?

The *no free lunch* theorem states that there is no "best" optimization algorithm. For each objective function a different algorithm can perform best.

First, it is advisable to attempt a simple search heuristic (e. g. *IMKLH*), before considering more complicated algorithms. Certainly if *a priori* knowledge about the problem is given, like an additive structure of the fitness function, an algorithm should be chosen which can exploit this. For example, for grid structures like the Ising spin glass, *BKDA* with the pentavariate factorization is a good choice.

MAXSAT is a rather densely connected problem. For such a structure, a sensible heuristics for choosing the most important connections of the graphical model is advisable. This recommends *FDA* with merge or *LFDA* on this problem. Also, since the objective function (the number of fulfilled clauses) is not very informative for *EDA*, a local hillclimber like *KLH* is very helpful.

For the Kaufmann function, *KLH* is not indispensable; if k is not too large, normal *FDA* with the new subfunction merge and a large population can solve it, too. For larger k, the factorization sets become too large and must be pruned.

In this manner, every optimization problem has its specific properties which should be considered in the choice of the algorithm.

Bibliography

[ACP87] S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal of Algebraic and Discrete Methods*, 8(2):277–284, 1987.

[Alm95] R. G. Almond. *Graphical Belief Modeling*. Chapman & Hall, London, 1995.

[Alt95] L. Altenberg. The schema theorem and Price's theorem. In D. Whitley and M. Vose, editors, *Foundations of Genetic Algorithms 3*, pages 23–49, San Francisco, 1995. Morgan-Kaufman.

[AM01] S. M. Aji and R. J. McEliece. The generalized distributive law and free energy minimization. In *Proceedings of the 39th Annual Allerton Conference on Communication, Control, and Computing*, pages 672–681, 2001.

[Bal02] S. Baluja. Using a priori knowledge to create probabilistic models for optimization. *International Journal of Approximate Reasoning*, 31(3):193–220, 2002.

[Bar82] F. Barahona. On the computational complexity of Ising spin glass models. *J. Phys. A: Math. Gen.*, 15:3241–3253, 1982.

[BB72] U. Bertelè and F. Brioschi. *Nonserial Dynamic Programming*. Academic Press, New York, 1972.

[BC95] S. Baluja and R. Caruana. Removing the genetics from the standard genetic algorithm. In *Proc. of the 12th Intern. Conf. on Machine Learning*, Lake Tahoe, 1995.

[BIV97] J. S. De Bonet, Ch. L. Isbell, and P. Viola. MIMIC: Finding optima by estimating probability densities. In M.C. Mozer, M.I. Jordan, and Th. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 424–431, Cambridge, 1997. MIT Press.

[Bou94] R.R. Bouckaert. Properties of Bayesian network learning algorithms. In R. Lopez de Mantaras and D. Poole, editors, *Proc. Tenth Conference on Uncertainty in Artificial Intelligence*, pages 102–109, San Francisco, 1994. Morgan Kaufmann.

[Bré99] P. Brémaud. *Markov Chains. Gibbs Fields, Monte Carlo Simulation and Queues*. Springer, New York, 1999.

[CL68] C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Trans. on Information Theory*, IT14(3):462–467, 1968.

[Csi75] I. Csiszár. I-divergence geometry of probability distributions and minimization problems. *Annals of Probability*, 3:146–158, 1975.

[CT89] Th. M. Cover and J.A. Thomas. *Elements of Information Theory*. Wiley, New York, 1989.

[Cul98] J. C. Culberson. On the futility of blind search: An algorithmic view of "no free lunch". *Evolutionary Computation Journal*, 6(2):109–128, 1998.

[DR72] J. N. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43(5):1470–1480, 1972.

[DS40] W. E. Deming and F. F. Stephan. On a least square adjustment of a sampled frequency table when the expected marginal totals are known. *Ann. Math. Statist.*, 11:427–444, 1940.

[EL99] R. Etxeberria and P. Larrañaga. Global optimization using Bayesian networks. In A. Ochoa Rodriguez, M. R. Soto Ortiz, and R. Santana Hermida, editors, *Proc. of the Second Symposium on Artificial Intelligence Adaptive Systems*, pages 332–339, La Habana, Cuba, 1999. ICIMAF.

[FG99] N. Friedman and M. Goldszmidt. Learning Bayesian networks with local structure. In Jordan [Jor99], pages 421–459.

[Fog98] D. Fogel, editor. *Evolutionary Computation: The Fossil Record*. IEEE Press, Piscataway, NJ, 1998.

[Fuk04a] A. S. Fukunaga. Efficient implementations of SAT local search. In *Proceedings of SAT-2004 (Seventh International Conf. on Theory and Applications of Satisfiability Testing)*, 2004.

[Fuk04b] A. S. Fukunaga. Evolving local search heuristics for SAT using genetic programming. In *Genetic and Evolutionary Computation - GECCO 2004*, volume 3103 of *Lecture Notes in Computer Science*, pages 483–494. Springer, 2004.

[FY96] N. Friedman and Z. Yakhini. On the sample complexity of learning Bayesian networks. In *Proceedings of the 12th Conference on Uncertainty in AI*, pages 274–282, 1996.

[GC05] Y. Gao and J. Culberson. Space complexity of estimation of distribution algorithms. *Evolutionary Computation*, 13(1):125–143, 2005.

[Gei44] H. Geiringer. On the probability theory of linkage in Mendelian heredity. *Annals of Math. Stat.*, 15:25–57, 1944.

[GMR02] J. Gottlieb, E. Marchiori, and C. Rossi. Evolutionary algorithms for the satisfiability problem. *Evolutionary Computation*, 10(1):35–50, 2002.

[Gol87] D. E. Goldberg. Simple genetic algorithms and the minimal, deceptive problem. In L. Davis, editor, *Genetic Algorithms and Simulated Annealing*, pages 74–88. Pitman, London, 1987.

[Gol89] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, 1989.

[Grü98] P. Grünwald. *The Minimum Description Length Principle and Reasoning under Uncertainty*. PhD thesis, Universiteit van Amsterdam, 1998.

[HD96] C. Huang and A. Darwiche. Inference in belief networks: A procedural guide. *International Journal of Approximate Reasoning*, 15(3):225–263, 1996.

[Hec99] D. Heckerman. A tutorial on learning with Bayesian networks. In Jordan [Jor99], pages 301–354.

[Hen88] M. Henrion. Propagation of uncertainty by probabilistic logic sampling in Bayes' networks. In J. F. Lemmer and L. N. Kanal, editors, *Uncertainty in Artificial Intelligence 2*, pages 149–164, Amsterdam, 1988. North Holland.

[Hes03] T. Heskes. Stable fixed points of loopy belief propagation are minima of the Bethe free energy. In *Advances in neural information processing systems 15: proceedings of the 2002 conference*, pages 343–350, Cambridge, 2003. MIT Press.

[Hes04] T. Heskes. On the uniqueness of loopy belief propagation fixed points. *Neural Computation*, 16:2379–2413, 2004.

[HGC95] D. Heckerman, D. Geiger, and D. Chickering. Learning Bayesian networks: The combinations of knowledge and statistical data. *Machine Learning*, 20:197–243, 1995.

[HGN01] C. Van Hoyweghen, D. E. Goldberg, and B. Naudts. Building block superiority, multimodality and synchronization problems. In Lee Spector et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference 2001*, pages 694–701. Morgan Kaufmann, 2001.

[HKS04] W. E. Hart, N. Krasnogor, and J. E. Smith. *Recent Advances in Memetic Algorithms*, chapter Memetic Evolutionary Algorithms, pages 3–30. Springer, Berlin, 2004.

[HMP03] J. Harel, R. J. McEliece, and R. Palanki. Poset belief propagation - experimental results. In *Proceedings of the International Symposium on Information Theory*, page 177, 2003.

[Hol75] J.H. Holland. *Adaptation in Natural and Artificial Systems.* Univ. of Michigan Press, Ann Arbor, 1975.

[Hoy03] C. Van Hoyweghen. *Symmetry in the representation of an optimization problem.* PhD thesis, Department of Mathematics and Computer Science, University of Antwerp, Belgium, 2003.

[HS00] H. Hoos and Th. Stützle. SATLIB: An online resource for research on SAT. In I. P. Gent, H. v. Maaren, and T. Walsh, editors, *SAT 2000*, pages 283–292. IOS Press, 2000. SATLIB is available online at www.satlib.org.

[IK68] C. T. Ireland and S. Kullback. Contingency tables with given marginals. *Biometrika*, 1968.

[Isi25] E. Ising. Beitrag zur Theorie des Ferromagnetismus. *Zeitschr. f. Physik*, 31:253–258, 1925.

[Jay57] E. T. Jaynes. Information theory and statistical mechanics. *Phys. Rev*, 6:620–643, 1957.

[Jay78] E. T. Jaynes. Where do we stand on maximum entropy? In R. D. Levine and M. Tribus, editors, *The Maximum Entropy Formalism.* MIT Press, Cambridge, 1978.

[JJ94] F. V. Jensen and F. Jensen. Optimal junction trees. In *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence*, pages 360–366, Seattle, 1994.

[JM97] D. S. Johnson and L. A. McGeoch. The traveling salesman problem: a case study. In E. Aarts and J. K. Lenstra, editors, *Local Search in Combinatorial Optimization*, pages 215–310. Wiley, Chichester, 1997.

[Jor99] M. I. Jordan, editor. *Learning in Graphical Models.* MIT Press, Cambrigde, 1999.

[JP73] R.A. Jarvis and E.A. Patrick. Clustering using a similarity measure based on shared near neighbors. *IEEE Transactions on Computers*, C22:1025–1034, 1973.

[JP95] R. Jiroušek and St. Přeučil. On the effective implementation of the iterative proportional fitting procedure. *Computational Statistics & Data Analysis*, 19:177–189, 1995.

[Ker98] Gabriele Kern-Isberner. Characterizing the principle of minimum cross-entropy within a conditional-logical framework. *Artificial Intelligence*, 98(1-2):169–208, 1998.

[KGV83] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[Kik51] R. Kikuchi. A theory of cooperative phenomena. *Phys.Review*, 115:988–1003, 1951.

[KL70] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell Syst. Techn Journ.*, 2:291–307, 1970.

[Kul68] S. Kullback. Probability densities with given marginals. *Annals of Mathematical Statistics*, 39(4):1236–1243, 1968.

[Lau96] S. L. Lauritzen. *Graphical Models*. Clarendon Press, Oxford, 1996.

[Lew59] P. M. Lewis II. Approximating probability distributions to reduce storage requirements. *Information and Control*, 2:214–225, 1959.

[LK73] S. Lin and B. W. Kernighan. An efficient heuristic for the traveling salesman problem. *Operations Research*, 21:298–516, 1973.

[LL01] P. Larrañaga and J.A. Lozano. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Optimization*. Kluwer Academic Press, Boston, 2001.

[LMS02] H. R. Lourenco, O. Martin, and T. Stützle. Iterated local search. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, pages 321–353. Kluwer Academic Publishers, Norwell MA, 2002.

[LPHJ03] F. Liers, M. Palassini, A. K. Hartmann, and M. Jünger. Ground state of the Bethe lattice spin glass and running time of an exact optimization algorithm. *Physical Review B*, January 2003.

[Mah01] Th. Mahnig. *Populationsbasierte Optimierung durch das Lernen von Interaktionen mit Bayes'schen Netzen*. PhD thesis, Universität Bonn, 2001.

[Mey98] C.-H. Meyer. *Korrektes Schließen bei unvollständiger Information*. PhD thesis, Fernuniversität Hagen, 1998.

[MH02a] H. Mühlenbein and R. Höns. Stochastic analysis of cellular automata and the voter model. In S. Bandini, B. Chopard, and M. Tomassini, editors, *Cellular Automata. 5th International Conference on Cellular Automata for Research and Industry. ACRI 2002*, volume 2493 of *LNCS*, pages 92–103, Berlin, 2002. Springer.

[MH02b] H. Mühlenbein and R. Höns. Stochastic analysis of cellular automata with application to the voter model. *Advances in Complex Systems*, 5(2 & 3):301–337, 2002.

[MH05] H. Mühlenbein and R. Höns. The estimation of distributions and the minimum relative entropy principle. *Evolutionary Computation*, 13(1):1–27, 2005.

[MM72] A. Martelli and U. Montanari. Nonserial dynamic programming: On the optimal strategy of variable elimination for the rectangular lattice. *J. Math. Anal. Appl.*, 40:226–242, 1972.

[MM98] H. Mühlenbein and Th. Mahnig. Convergence theory and applications of the factorized distribution algorithm. *Journal of Computing and Information Technology*, 7(1):19–32, 1998.

[MM99] H. Mühlenbein and Th. Mahnig. FDA - a scalable evolutionary algorithm for the optimization of additively decomposed functions. *Evolutionary Computation*, 7(4):353–376, 1999.

[MM00] H. Mühlenbein and Th. Mahnig. Evolutionary algorithms: From recombination to search distributions. In L. Kallel, B. Naudts, and A. Rogers, editors, *Theoretical Aspects of Evolutionary Computing*, Natural Computing, pages 137–176. Springer Verlag, Berlin, 2000.

[MM01a] Th. Mahnig and H. Mühlenbein. A new adaptive Boltzmann selection schedule SDS. In *Proceedings CEC 2001*, pages 183–190, Piscataway, 2001. IEEE Press.

[MM01b] H. Mühlenbein and Th. 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.

[MM02] H. Mühlenbein and Th. Mahnig. Evolutionary optimization and the estimation of search distributions with applications to graph bipartitioning. *Journal of Approximate Reasoning*, 31(3):157–192, 2002.

[MMO99] H. Mühlenbein, Th. Mahnig, and A. Ochoa. Schemata, distributions and graphical models in evolutionary optimization. *Journal of Heuristics*, 5(2):213–247, 1999.

[MP96] 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, *Lecture Notes in Computer Science 1141: Parallel Problem Solving from Nature - PPSN IV)*, pages 178–187, Berlin, 1996. Springer-Verlag.

[MPV87] M. Mézard, G. Parisi, and M. A. Virasoro. *Spin glass theory and beyond*. World Scientific, Singapore, 1987.

[MT93] M. de la Maza and B. Tidor. An analysis of selection procedures with particular attention paid to proportional and Boltzmann selection. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 124–131, San Mateo, CA, 1993. Morgan Kaufman.

[Müh91] H. Mühlenbein. Evolution in time and space - the parallel genetic algorithm. In G. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 316–337, San Mateo, 1991. Morgan-Kaufman.

[Müh92] H. Mühlenbein. How Genetic Algorithms Really Work: Mutation and Hillclimbing. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature*, pages 15–26, Amsterdam, 1992. North-Holland.

[MV96] H. Mühlenbein and H.-M. Voigt. Gene pool recombination in genetic algorithms. In J. P. Kelly and I. H. Osman, editors, *Metaheuristics: Theory and Applications*, pages 53–62, Norwell, 1996. Kluwer Academic Publisher.

[MY02] R. J. McEliece and M. Yildirim. Belief propagation on partially ordered sets. In *Proceedings of the 15th Internatonal Symposium on Mathematical Theory of Networks and Systems (MTNS 2002)*, 2002.

[Nil86] N. Nilsson. Probabilistic logic. *Artificial Intelligence*, 28(1):71–87, 1986.

[OHSM03] A. Ochoa, R. Höns, M. Soto, and H. Mühlenbein. A maximum entropy approach to sampling in EDA - the single connected case. In A. Sanfeliu and J. Ruiz-Shulcloper, editors, *Proceedings of the 8th Iberoamerican Congress on Pattern Recognition (CIARP 2003)*, volume 2905 of *Lecture Notes in Computer Science*, pages 683–690. Springer, 2003.

[Ons44] L. Onsager. Crystal statistics. I. a two-dimensional model with an order-disorder transition. *Phys. Rev.*, 65(3& 4):117–149, 1944.

[PA05] P. Pakzad and V. Anantharam. Estimation and marginalization using Kikuchi approximation methods. *Neural Computation*, 17(8):1836–1873, August 2005.

[Pea88] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Mateo, CA, 1988.

[Pei36] R. Peierls. On Ising's model of ferromagnetism. *Proc. Cambridge Phil. Soc.*, 32:477–481, 1936.

[PG03] M. Pelikan and D.E. Goldberg. Hierarchical BOA solves Ising spin glasses and MAXSAT. In *Genetic and Evolutionary Computation Conference 2003*, volume 2724 of *Lecture Notes in Computer Science*, pages 1271–1282. Springer, 2003. Also IlliGAL Report No. 2003001.

[PGC99] M. Pelikan, D.E. Goldberg, and E. Cantú-Paz. BOA: The Bayesian optimization algorithm. In *Genetic and Evolutionary Computation Conference - GECCO 1999*, volume 1, pages 525–532. Morgan Kaufmann Publishers, 1999.

[PGOT03] M. Pelikan, D. E. Goldberg, J. Ocenasek, and S. Trebst. Robust and scalable black-box optimization, hierarchy, and Ising spin glasses. Technical Report 2003019, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, 2003.

[PM99] Martin Pelikan and Heinz Mühlenbein. The bivariate marginal distribution algorithm. In R. Roy, T. Furuhashi, and P. K. Chawdhry, editors, *Advances in Soft Computing - Engineering Design and Manufacturing*, pages 521–535, Berlin, 1999. Springer-Verlag.

[PV97] J. Paris and A. Vencovská. In defense of the maximum entropy inference process. *International Journal of Approximate Reasoning*, 17(1):77–103, 1997.

[Rat97] A. Ratnaparkhi. A simple introduction to maximum entropy models for natural language processing. Technical Report 97-08, Institute for Research in Cognitive Science, University of Pennsylvania, Philadelphia, 1997.

[Ris78] J. Rissanen. Modeling the shortest data description. *Automatica*, 14:465–471, 1978.

[Ris89] J. Rissanen. *Stochastic Complexity in Statistical Inquiry*. World Scientific Publishing Company, 1989.

[Rob18] R. B. Robbins. Some applications of mathematics to breeding problems III. *Genetics*, 3:375–389, 1918.

[Rob96] G. O. Roberts. Markov chain concepts related to sampling algorithms. In W. R. Gilks, S. Richardson, and D. J. Spiegelhalter, editors, *Markov chain Monte Carlo in Practice*, pages 45–58. Chapman & Hall, London, 1996.

[San04] R. Santana. *Probabilistic modeling based on undirected graphs in Estimation Distribution Algorithms*. PhD thesis, Institute of Cybernetics, Mathematics and Physics, Havana, 2004.

[San05] R. Santana. Estimation of distribution algorithms with Kikuchi approximations. *Evolutionary Computation*, 13(1):67–97, 2005.

[Sch78] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 7:461–464, 1978.

[Sha48] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, July and October 1948.

[SJ80] J. E. Shore and R. W. Johnson. Axiomatic derivation of the principle of maximum entropy and the principle of minimum cross entropy. *IEEE Transactions on Information Theory*, IT-26(1):26–37, 1980.

[SKC94] B. Selman, H. A. Kautz, and B. Cohen. Noise strategies for improving local search. In *AAAI-94*, pages 337–343, 1994.

[SKC95] B. Selman, H. A. Kautz, and B. Cohen. Local search strategies for satisfiability testing. In D. S. Johnson and M. A. Trick, editors, *Cliques, Coloring, and Satisfiability*, volume 26 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 521–532, Providence RI, 1995. American Math. Soc.

[SLM92] B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In *AAAI-92*, pages 440–446, 1992.

[SO00] M. Soto and A. Ochoa. A Factorized Distribution Algorithm based on polytrees. In *Proceedings of the 2000 Congress on Evolutionary Computation CEC00*, pages 232–237, La Jolla, California, 6-9 July 2000. IEEE Press.

[Sot03] M. Soto. *A Singled Connected Factorized Distribution Algorithm and its cost of evaluation*. PhD thesis, University of Havana, Havana, Cuba, July 2003.

[Sta86] R. Stanley. *Enumerative Combinatorics*, volume 1. Wadsworth & Brooks/Cole, Monterey, 1986.

[Sto67] T. Stoppard. *Rosencrantz & Guildenstern Are Dead*. Grove Press, New York, 1967.

[Sys93] G. Syswerda. Simulated crossover in genetic algorithms. In L. D. Whitley, editor, *Foundations of Genetic Algorithms 2*, pages 239–255, San Mateo, 1993. Morgan Kaufmann.

[Teh03] Y. W. Teh. *Bethe Free Energy and Contrastive Divergence Approximations for Undirected Graphical Models*. PhD thesis, University of Toronto, 2003.

[TW03] 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.

[Uff95] J. Uffink. Can the maximum entropy principle be explained as a consistency requirement? *Studies in History and Philosophy of Modern Physics*, 26B:223–261, 1995.

[WD03] D. Weininger and J. Delany. *Daylight Clustering Manual*. Daylight Chemical Information Systems, Inc., Version 4.82, June 2003. Available online at http://www.daylight.com/dayhtml/doc/cluster.

[Whi90] Joe Whittaker. *Graphical Models in Applied Multivariate Statistics*. John Wiley & Sons, New York, 1990.

[Wil87] P. Willett. *Similarity and Clustering in Chemical Information Systems*. Wiley, New York, 1987.

[WM95] D. Wolpert and W. Macready. No free lunch theorems for search. Technical Report SFI-TR-95-02-010, Santa Fe Institute, Santa Fe, NM, USA, February 1995.

[WPS+04] A. Wright, R. Poli, C. Stephens, W. B. Landgon, and S. Pulavarty. An estimation of distribution algorithm based on maximum entropy. In *Proceedings of GECCO 2004*, volume 3103 of *Lecture Notes in Computer Science*, pages 343–354. Springer, 2004.

[XWC97] Y. Xiang, S. K. M. Wong, and N. Cercone. A 'microscopic' study of minimum entropy search in learning decomposable Markov networks. *Machine Learning*, 26:65–92, 1997.

[YFW01] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. Technical Report 2001-22, Mitsubishi Electric Research Laboratories, August 2001.

[YFW02] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Constructing free energy approximations and generalized belief propagation algorithms. Technical Report 2002-35, Mitsubishi Electric Research Laboratories, August 2002.

[YFW04] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Constructing free energy approximations and generalized belief propagation algorithms. Technical Report 2004-040, Mitsubishi Electric Research Laboratories, May 2004. Substantially revised version of [YFW02].

[Yui02] A. L. Yuille. CCCP algorithms to minimize the Bethe and Kikuchi free energies: Convergent alternatives to belief propagation. *Neural Computation*, 14(7):1691–1722, 2002.

A. Appendix: Ising Spin Glass Instances

This appendix contains the solutions of the suite of random instances of the Ising problem used in the thesis. The instances are available at http://www.hoens.net/robin/ising in the format used by the spin glass server in Cologne[1]. It begins with a short header, giving the size and the random seed used within the FDA software package. Then there is one line for each connection, in the format $\mu\ \nu\ J_{\mu,\nu}$, with the indices $\mu, \nu \in \{1, \ldots, n\}$.

The coupling constants $J_{\mu,\nu}$ were chosen randomly between -1 and 1. For pairs of spins that transgress the boundary, the coupling is 0, because we consider the fixed-boundary model. (These connections are listed in the files anyway, because this is required by the Cologne spin glass server.)

The ground states of the given instances were calculated using the spin glass server in Cologne and verified using the FDA software. The solutions are given as the indices of the spins that should be set to 1; all others should be set to -1 (or vise versa). The last column gives the energy of the ground state.

A.1. Instances of Size 7×7

Seed	Spins up	Energy
1	2, 4, 8, 10, 15, 19, 21, 25, 26, 29, 32, 33, 35, 37, 39, 41, 42, 43, 44, 45, 46	−34.664884
2	1, 2, 6, 7, 8, 17, 21, 24, 30, 32, 35, 38, 39, 40, 41, 45, 46, 49	−35.02556
3	1, 2, 3, 7, 12, 13, 15, 17, 20, 21, 23, 24, 26, 27, 28, 29, 31, 33, 35, 36, 37, 39, 44, 47	−35.416631
4	1, 4, 5, 11, 15, 16, 18, 23, 26, 27, 28, 29, 30, 34, 35, 36, 38, 42, 43, 44, 45, 48	−33.737626
5	1, 2, 5, 7, 9, 10, 12, 13, 14, 16, 20, 26, 28, 30, 33, 34, 35, 40, 42, 43, 44, 45, 48, 49	−37.363193
6	4, 5, 7, 11, 12, 14, 16, 17, 18, 20, 25, 28, 29, 32, 33, 34, 37, 38, 39, 41, 43, 46, 47, 49	−31.408687
7	1, 2, 4, 5, 6, 11, 17, 20, 23, 24, 26, 30, 34, 35, 36, 37, 38, 40, 41, 42, 44, 45, 48	−34.474571
8	1, 3, 5, 10, 11, 12, 14, 15, 17, 18, 21, 22, 24, 25, 26, 27, 29, 33, 35, 39, 40, 41, 43, 49	−34.514092
9	1, 2, 9, 10, 14, 15, 16, 19, 21, 30, 31, 32, 33, 34, 36, 38, 41, 42, 43, 45, 47, 48	−32.884739
10	2, 3, 4, 5, 8, 17, 20, 21, 22, 23, 24, 25, 27, 28, 38, 39, 40, 41, 44, 45, 48	−35.54777

[1] http://www.informatik.uni-koeln.de/ls_juenger/research/sgs/sgs.html

A.2. Instances of Size 10×10

Seed	Spins up	Energy
1	1, 10, 13, 18, 19, 20, 22, 23, 25, 27, 28, 30, 31, 32, 34, 37, 38, 39, 42, 46, 49, 50, 51, 53, 56, 60, 65, 68, 69, 71, 72, 73, 74, 79, 92, 93, 96, 97, 98	−65.99706
2	1, 2, 5, 6, 8, 10, 11, 16, 17, 22, 26, 28, 30, 31, 32, 35, 36, 40, 42, 43, 47, 48, 50, 52, 54, 55, 57, 58, 59, 61, 62, 63, 64, 67, 69, 73, 75, 77, 82, 85, 86, 89, 90, 91, 92, 95, 98	−73.359045
3	5, 6, 7, 9, 10, 11, 12, 15, 17, 23, 24, 26, 29, 30, 31, 33, 34, 36, 40, 43, 44, 46, 47, 50, 54, 55, 59, 60, 62, 63, 64, 70, 71, 76, 78, 81, 82, 83, 84, 86, 87, 90, 94, 98, 99	−72.69937
4	1, 3, 5, 7, 8, 13, 14, 17, 26, 28, 31, 32, 33, 39, 40, 43, 44, 46, 47, 48, 49, 50, 52, 53, 54, 55, 56, 57, 59, 60, 63, 64, 65, 73, 75, 77, 78, 80, 81, 86, 88, 91, 92, 93, 94, 97, 98, 99, 100	−72.516619
5	1, 4, 8, 9, 10, 12, 19, 20, 22, 26, 27, 28, 29, 30, 34, 36, 37, 38, 40, 42, 44, 47, 49, 53, 54, 57, 58, 59, 64, 65, 67, 68, 69, 70, 71, 74, 75, 80, 87, 89, 90, 91, 93, 94, 100	−73.537427
6	1, 2, 4, 5, 7, 8, 10, 11, 12, 18, 19, 20, 21, 25, 27, 28, 30, 32, 33, 34, 38, 39, 40, 41, 45, 46, 48, 49, 55, 61, 63, 65, 68, 71, 72, 73, 74, 79, 84, 89, 90, 93, 95, 98, 99	−72.79642
7	1, 2, 3, 5, 6, 7, 8, 9, 12, 13, 14, 16, 18, 20, 24, 27, 29, 31, 34, 39, 40, 41, 45, 47, 50, 51, 52, 54, 59, 61, 68, 69, 72, 77, 81, 82, 84, 88, 89, 90, 94, 95, 97, 99, 100	−73.976806
8	1, 3, 4, 6, 7, 9, 10, 14, 15, 17, 21, 23, 24, 26, 29, 31, 34, 35, 37, 38, 39, 40, 41, 44, 45, 49, 50, 51, 52, 54, 55, 58, 60, 62, 64, 68, 69, 71, 72, 74, 75, 79, 80, 81, 82, 84, 89	−70.732175
9	4, 7, 8, 9, 11, 12, 13, 14, 15, 16, 18, 19, 20, 23, 24, 25, 27, 28, 29, 30, 35, 36, 41, 44, 45, 48, 49, 52, 54, 57, 59, 60, 62, 63, 64, 68, 77, 78, 80, 84, 87, 88, 89, 92, 93, 97, 99	−70.996549
10	1, 4, 5, 6, 9, 12, 13, 17, 18, 20, 21, 22, 25, 27, 28, 31, 34, 35, 36, 40, 43, 45, 48, 50, 53, 56, 58, 60, 62, 63, 65, 69, 72, 73, 76, 78, 80, 81, 82, 83, 84, 85, 89, 90, 92, 93, 95, 97, 98, 99	−76.212219

A.3. Instances of Size 15×15

Seed	Spins up	Energy
1	1, 3, 4, 6, 7, 9, 10, 12, 13, 14, 22, 25, 27, 29, 30, 32, 33, 37, 38, 40, 41, 42, 44, 45, 49, 50, 52, 53, 55, 60, 61, 62, 66, 68, 71, 72, 74, 77, 81, 82, 88, 89, 91, 92, 93, 94, 97, 99, 100, 102, 103, 105, 107, 110, 115, 116, 121, 122, 124, 128, 132, 133, 135, 136, 137, 139, 141, 143, 145, 148, 150, 151, 152, 158, 160, 163, 167, 169, 170, 171, 173, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 187, 190, 191, 193, 194, 195, 197, 198, 206, 207, 208, 209, 210, 211, 221, 222, 224, 225	−165.426135
2	3, 4, 6, 8, 10, 18, 19, 21, 22, 23, 24, 25, 26, 29, 31, 32, 37, 39, 40, 48, 49, 50, 51, 54, 56, 57, 58, 59, 60, 62, 63, 69, 71, 74, 81, 82, 83, 84, 88, 89, 90, 91, 92, 93, 95, 97, 101, 102, 103, 104, 107, 110, 111, 112, 113, 118, 120, 122, 123, 124, 127, 128, 129, 133, 136, 137, 140, 141, 142, 145, 147, 148, 149, 150, 153, 155, 157, 158, 166, 170, 172, 176, 177, 179, 180, 181, 182, 183, 185, 189, 202, 203, 204, 206, 207, 209, 210, 214, 216, 217, 218, 219, 223, 225	−170.998755
3	1, 2, 3, 4, 5, 6, 7, 8, 9, 12, 15, 17, 18, 22, 23, 24, 27, 30, 31, 34, 35, 36, 37, 40, 41, 42, 44, 47, 48, 49, 50, 52, 54, 56, 57, 58, 59, 60, 64, 73, 75, 84, 87, 88, 90, 91, 93, 94, 96, 97, 98, 100, 101, 103, 104, 106, 108, 109, 112, 115, 119, 123, 124, 125, 127, 130, 131, 133, 135, 136, 138, 142, 145, 147, 148, 150, 151, 152, 153, 160, 161, 167, 170, 176, 177, 178, 179, 180, 181, 182, 184, 185, 186, 191, 192, 194, 195, 196, 197, 200, 202, 203, 204, 206, 209, 210, 213, 217, 218, 223, 224	−167.659186
4	1, 3, 4, 5, 7, 9, 13, 15, 21, 22, 23, 25, 27, 28, 35, 36, 38, 40, 43, 49, 50, 51, 52, 53, 55, 56, 57, 58, 60, 61, 62, 63, 64, 66, 67, 71, 72, 74, 75, 76, 80, 81, 82, 83, 87, 93, 96, 99, 101, 104, 105, 108, 109, 110, 111, 115, 116, 119, 122, 123, 126, 127, 129, 133, 135, 137, 139, 140, 141, 144, 146, 148, 150, 151, 152, 153, 154, 155, 158, 159, 160, 161, 162, 163, 166, 169, 171, 172, 173, 175, 177, 179, 182, 185, 187, 189, 193, 194, 195, 197, 200, 204, 205, 209, 212, 213, 217, 218, 221, 222, 223	−164.473406
5	3, 5, 6, 7, 8, 9, 10, 11, 12, 15, 16, 23, 25, 26, 27, 28, 30, 31, 34, 39, 42, 43, 44, 45, 46, 47, 48, 49, 50, 57, 58, 59, 60, 62, 64, 65, 66, 67, 69, 73, 74, 75, 78, 79, 85, 90, 95, 97, 98, 100, 102, 111, 113, 115, 121, 122, 124, 127, 129, 130, 132, 135, 136, 138, 139, 141, 142, 143, 144, 146, 147, 149, 150, 151, 153, 154, 157, 161, 164, 169, 170, 171, 172, 177, 178, 180, 182, 183, 185, 187, 188, 189, 190, 193, 195, 197, 203, 204, 205, 208, 209, 211, 213, 214, 217, 218, 222, 224, 225	−168.539819
6	1, 2, 4, 5, 7, 12, 14, 16, 17, 18, 19, 22, 23, 27, 29, 30, 31, 34, 35, 37, 38, 41, 43, 44, 45, 46, 47, 48, 50, 54, 56, 57, 58, 60, 62, 65, 66, 69, 70, 73, 76, 79, 81, 83, 84, 86, 88, 94, 97, 100, 101, 105, 114, 117, 120, 123, 126, 127, 129, 130, 133, 139, 140, 141, 142, 144, 146, 148, 153, 154, 155, 157, 158, 159, 160, 161, 162, 167, 168, 170, 176, 178, 180, 182, 191, 193, 194, 195, 196, 198, 199, 201, 203, 206, 207, 209, 210, 211, 212, 213, 214, 216, 217, 218, 221, 224	−168.823523
7	4, 5, 11, 16, 17, 18, 21, 22, 23, 26, 27, 29, 30, 33, 34, 35, 38, 46, 48, 51, 52, 53, 54, 55, 56, 59, 60, 61, 64, 65, 67, 75, 76, 77, 78, 79, 81, 86, 87, 88, 90, 92, 95, 96, 97, 98, 99, 100, 101, 102, 106, 109, 112, 116, 118, 119, 120, 121, 122, 123, 124, 125, 127, 133, 135, 137, 138, 139, 140, 141, 142, 144, 146, 154, 155, 157, 158, 168, 169, 175, 176, 177, 179, 180, 183, 184, 185, 189, 190, 193, 194, 199, 202, 205, 209, 210, 212, 213, 214, 216, 218, 219, 220, 221, 222, 223, 224	−169.817999
8	2, 3, 4, 6, 7, 8, 9, 15, 16, 17, 18, 19, 22, 23, 24, 25, 26, 27, 30, 32, 35, 36, 39, 40, 45, 47, 50, 51, 55, 58, 59, 66, 68, 72, 73, 75, 78, 79, 81, 83, 84, 88, 91, 94, 97, 98, 101, 104, 105, 108, 109, 113, 115, 116, 118, 119, 120, 124, 125, 128, 129, 130, 132, 134, 136, 137, 138, 141, 142, 144, 145, 149, 157, 158, 161, 166, 167, 168, 172, 174, 175, 176, 178, 180, 184, 186, 187, 188, 190, 191, 193, 197, 198, 199, 200, 201, 203, 207, 209, 210, 212, 213, 215, 216, 220, 222, 223, 224	−169.024191
9	1, 2, 6, 11, 12, 13, 14, 17, 21, 24, 25, 26, 27, 29, 33, 35, 36, 38, 39, 40, 51, 53, 56, 57, 60, 61, 65, 67, 68, 69, 73, 77, 80, 84, 88, 89, 90, 92, 93, 94, 98, 99, 100, 101, 102, 104, 105, 108, 110, 112, 113, 114, 118, 120, 124, 125, 128, 129, 130, 131, 132, 137, 140, 142, 144, 146, 154, 155, 159, 160, 163, 165, 166, 169, 170, 171, 172, 173, 174, 185, 189, 190, 192, 193, 195, 196, 198, 201, 203, 212, 215, 217, 220, 221, 222, 223, 224, 225	−166.844277
10	1, 3, 5, 6, 9, 12, 13, 17, 18, 20, 22, 26, 27, 29, 31, 32, 34, 36, 40, 41, 43, 47, 48, 52, 53, 54, 55, 57, 60, 61, 62, 64, 66, 70, 73, 75, 80, 84, 85, 86, 87, 88, 89, 92, 93, 94, 95, 99, 100, 101, 102, 105, 107, 108, 110, 111, 113, 114, 116, 123, 124, 128, 130, 133, 134, 135, 136, 140, 142, 145, 147, 148, 149, 151, 156, 160, 162, 164, 165, 166, 167, 168, 169, 170, 173, 177, 179, 180, 185, 191, 193, 195, 200, 201, 204, 207, 208, 211, 214, 216, 217, 218, 219, 220, 222, 223	−162.117506