Theoretical Computer Science 633 (2016) 4–18

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

MARLEDA: Effective distribution estimation through Markov random fields

Matthew Alden [a],[*], Risto Miikkulainen [b]

[a] Institute of Technology, University of Washington, Tacoma, WA 98402, United States
[b] Department of Computer Science, The University of Texas at Austin, Austin, TX 78741, United States

A R T I C L E   I N F O

Article history:
Received 27 December 2014
Received in revised form 20 July 2015
Accepted 27 July 2015
Available online 30 July 2015

Keywords:
Probabilistic modeling
Estimation of distribution algorithms
Markov random field
Markov chain Monte Carlo

A B S T R A C T

Estimation of Distribution Algorithms (EDAs) combine genetic algorithms with statistical modeling in order to learn and exploit the structure of search domains. Such algorithms work well when the EDA's statistical model matches the structure of the domain. Many EDAs use statistical models that represent domain structure with directed acyclic graphs (DAGs). While useful in many areas, DAGs have inherent restrictions that make undirected graph models a viable alternative for many domains. This paper introduces a new EDA, the Markovian Learning Estimation of Distribution Algorithm (MARLEDA), that makes effective use of this idea by employing a Markov random field model. MARLEDA is evaluated on four combinatorial optimization tasks, OneMax, deceptive trap functions, the 2D Rosenbrock function, and 2D Ising spin glasses. MARLEDA is shown to perform better than standard genetic algorithms and a DAG-based EDA. Improving the modeling capabilities of EDAs in this manner brings them closer to effective applications in difficult real-world domains, such as computational biology and autonomous agent design.

1. Introduction

In recent decades computers have become increasingly indispensable tools for researchers in many fields including mathematics, engineering, biology, and physics. Computational science transforms physical and mathematical problems into computational problems by defining a computational model. These models can be studied, adjusted, and experimented with to generate predictions about their real-world counterparts. In turn, real-world experiments help form the basis for new computational models. There is an interplay between real-world and computational research; each informs and refines the other. While a computational formulation of real-world problems is useful and convenient, there are drawbacks.

The key difficulty for many computational problems, such as DNA sequence alignment, selecting key attributes for data mining, or optimizing antenna design, is that no analytic solution methods exist. Alternatively, known analytic solutions may be too computationally expensive to be practical. Without the ability to construct an ideal solution efficiently, we are often reduced to searching for desirable solutions. Search algorithms are, in many cases, a relatively easy and effective means of producing near-optimal solutions to difficult problems. However, in complex domains search is often inefficient, taking too long to be practical.

Evolutionary search seeks to harness the creativity and problem-solving ability of biological evolution. The Genetic Algorithm (GA) [11,20], modeled after natural evolution, combines simple operations such as crossover and mutation to form

[*] Corresponding author.
E-mail addresses: mealden@uw.edu (M. Alden), risto@cs.utexas.edu (R. Miikkulainen).

http://dx.doi.org/10.1016/j.tcs.2015.07.049
0304-3975/© 2015 Elsevier B.V. All rights reserved.

a generic search system. However, nature retains two important advantages over such an algorithm: massive parallelism and deep time. Simple operations may be sufficient to discover complex systems in such a luxurious environment, but generating comparable results with reasonable computational resources requires greater sophistication.

Estimation of distribution algorithms (EDAs) [4,8,27,31] are a new and powerful approach to search. The main idea is to combine statistical modeling with evolutionary search. These algorithms exploit statistically identifiable structure within data to produce better solutions or to produce solutions more quickly. The statistical models can be defined a priori, injecting prior information into the search process, or learned as part of the algorithm's operation. The power of these algorithms therefore depends on two factors: (1) how appropriate the statistical model is to the domain, and (2) how well the system can learn it.

The majority of EDAs incorporate models that organize dependencies as directed acyclic graphs (DAGs), such as Bayesian networks [22,32]. There are many established learning mechanisms for DAGs and simple methods for sampling the resulting model. However, directed graph models are not necessarily the most natural representation of domain structure for all problems. For example, a DAG cannot represent, by definition, bi-directional or cyclic dependencies.

Several researchers have proposed utilizing undirected graph models in EDAs [15,40,43]. In particular, Markov random fields (MRFs) have been shown to be a promising basis for such models. However, learning and sampling MRFs is more difficult than DAGs, and has so far constrained their implementation.

This paper introduces the Markovian Learning Estimation of Distribution Algorithm (MARLEDA), a general-purpose EDA that can learn and use a Markov random field model. The MRF model is constructed to reflect the interactions of the search domain's parameters, allowing MARLEDA to identify good solutions intelligently. MARLEDA is compared experimentally to a standard GA and one of the most competent EDAs, the Bayesian Optimization Algorithm (BOA) [34], on several finite-valued combinatorial optimization tasks. MARLEDA performs well in these tasks, demonstrating not only the power of the system but also suggesting that EDAs are ready for application to real-world problems.

2. Background

A finite-valued combinatorial optimization task consists of a candidate solution space, $\mathbf{X}$, and an objective function, $f : \mathbf{X} \to \mathbb{R}$. The goal is to find a solution, $\mathbf{x}^* \in \mathbf{X}$, that either maximizes or minimizes the objective function, depending on the task. Candidate solutions are composed of many individual parameters, each with a finite number of possible values, thus the size of the solution space is combinatorial in the number of task parameters.

Difficult combinatorial optimization tasks often lack analytic solution methods and have solution spaces that are very large, making a systematic search computationally intractable. It is sometimes possible, however, to solve them approximately using probabilistic search methods. In the following sections two such methods are briefly reviewed, genetic algorithms and estimation of distribution algorithms. The basic theory behind MARLEDA's statistical model, Markov random fields and Pearson's χ^2 test, is also presented.

2.1. Genetic algorithms

Genetic algorithms (GAs) [11,20] are a well-established search method that has been successfully applied to a wide range of computational problems, such as planning, engineering design, and control. The basic principle underlying GAs make them well suited for combinatorial optimization tasks. GAs perform a parallel search, maintaining a *population* of candidate solutions that evolve over time. Evolution is guided by the objective function of the task, commonly called the *fitness function* or *fitness metric*. In keeping with GA conventions, "high-fitness" refers to desirable fitness function scores, i.e. high or low values depending on the direction of optimization. Similarly, "low-fitness" refers to undesirable fitness scores. The GA search mechanism is intended to visit candidate solutions of ever improving fitness at each iteration of the algorithm.

Following the biological analogy, candidate solutions are encoded in artificial *chromosomes*. A chromosome is composed of a set of *genes*, each representing a parameter of the optimization task (in practice there may be an additional mapping between a chromosomal encoding and a candidate solution, but for notational simplicity the space of chromosomes and the space of candidate solutions are assumed to be the same). The value of each gene, its *allele*, is one of the possible values for the associated task parameter.

The classic selection, recombination, and mutation operations govern the search behavior of GAs. These operators work well provided high-fitness chromosomes are located "near" other high-fitness chromosomes or their recombinations. GAs perform very well on tasks where such assumptions are true, i.e. tasks with *building blocks* [11]. However, GAs are less effective on tasks with more complicated parameter relationships, i.e. domains where combinations of genes must be correctly set to achieve high fitness.

2.2. EDAs

Estimation of distribution algorithms (EDAs) [4,8,27,31] address the building block problem by statistically inferring dependencies among genes. These dependencies are expressed in a statistical model, which can then be sampled to produce a candidate solutions. Through the sampling process, EDAs are likely to preserve high-fitness combinations of alleles, making the search process more efficient.

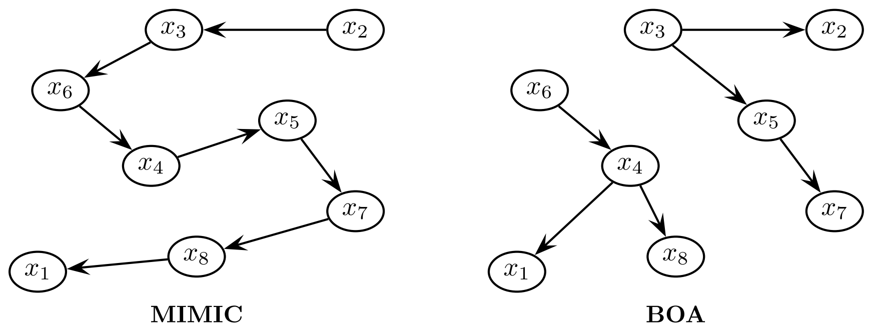

Fig. 1. Example learned gene-dependency structures for a fictional combinatorial optimization task with eight parameters. While both DAG-based methods, MIMIC and BMDA enforce very different global organization of dependencies. Domains that do match the assumed organization will be difficult to optimize, therefore the choice of statistical model is critically important when addressing a specific optimization task.

EDAs operate similarly to GAs, but sampling of the statistical model replaces the recombination and mutation operations:

1. The initial population of chromosomes, $\mathcal{P}(0)$, is uniformly sampled from $\mathbf{X}$ and the model, $\mathcal{M}(0)$, is initialized.
2. At iteration t, a subcollection, $\mathcal{P}'(t) \subseteq \mathcal{P}(t)$, of high-fitness chromosomes is selected.
3. $\mathcal{M}(t)$ is created to model the members of $\mathcal{P}'(t)$.
4. A collection of new chromosomes, $\mathcal{C}(t)$, is produced by sampling $\mathcal{M}(t)$.
5. A subcollection, $\mathcal{R}(t) \subseteq \mathcal{P}(t)$, of low-fitness chromosomes is selected.
6. $\mathcal{P}(t+1) \leftarrow \mathcal{P}(t) - \mathcal{R}(t) + \mathcal{C}(t)$.
7. Unless termination criteria are met, return to step 2.

The simplest EDAs employ univariate models [3,4,8,17,27,45], which do not identify gene dependencies. These models essentially record the marginal frequency of alleles for every gene. Let x_i be the ith gene of chromosome $\mathbf{x}$. New chromosome are constructed by sampling the modeled distribution

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

Bivariate EDAs [4,5] model dependencies between pairs of genes. Formally, the model is

$$P(\mathbf{x}) = \prod_{i=1}^{n} P\left(x_i|\text{parent-of}(x_i)\right). \tag{2}$$

Bivariate EDAs are distinguished by the restrictions placed on the parent–child relationship. For example, the Mutual Information Maximization for Input Clustering algorithm (MIMIC) [8] learns a chain of gene dependencies, while the Bivariate Marginal Distribution Algorithm (BMDA) [36] learns a forest of tree dependencies, as illustrated in Fig. 1.

Multivariate EDAs [1,10,28–30,38] model dependencies among larger sets of genes. One of the most successful multivariate EDAs is the Bayesian Optimization Algorithm (BOA) [34] and its hierarchical extension (hBOA) [33]. BOA and hBOA use a Bayesian network as the basis for the statistical model, capturing dependencies organized into a directed acyclic graph (DAG). The Bayesian network model is constructed to approximately minimize the Bayesian–Dirichlet difference [7, 19] between the model and the chromosome population, thus promoting an accurate model. The combination of a flexible model and a strong learning mechanism has made BOA an outstanding algorithm in its class. Consequently, BOA is used for comparison with MARLEDA on the benchmark experiments presented in Section 4.

The above bivariate and multivariate EDAs are based on directed graph models (or equivalent). Undirected graph models have been explored in a number of algorithms including the Extended Compact Genetic Algorithm (EcGA) [15], the Markov Network Factorized Distribution Algorithm (MN-FDA) [40], the Markov Network Estimation of Distribution Algorithm (MN-EDA) [41], and Distribution Estimation Using Markov Random Fields (DEUM) [42,43]. Undirected graph models have been shown to be superior to their directed graph model counterparts for many optimization tasks, making them strong candidates for further research.

The greatest challenge in using undirected graph models is the difficultly in learning and sampling the model. While Markov random fields have seen practical applications in fields such as physics and image processing [24,46], such applications do not traditionally involve learning. However, EDAs must be able to learn the MRF neighborhood system (defined in the next section) in order to maximize their effectiveness. Consequently, heuristic schemes have been developed for learning MRF neighborhood systems [40–42]. While these schemes draw on statistical methods, they generally lack the rigorous theoretical foundation available to directed graph methods.

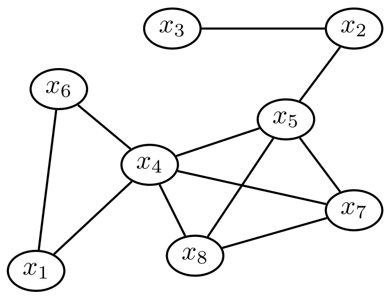

Fig. 2. An example undirected graph model for a fictional combinatorial optimization task with either parameters. Each variable (node) statistically depends on the variables adjacent to it. In contrast the DAG models of Fig. 1, this undirected graph model has no natural ordering of its nodes, making learning and sampling the model more difficult.

Sampling undirected graph models is difficult because the modeled variables are treated as inherently homologous. Directed graph models, and the conditional probability distributions encoded therein, define a natural ordering of the nodes of a DAG, such as the ordering achieved by "tracing" the dependency chain in Fig. 1(a). This ordering makes it possible to evaluate the model efficiently via standard graph traversal algorithms. In contrast, undirected graph models express relations that hold across many sets of variables simultaneously and provide no natural node ordering. Consequently, learning and sampling undirected graph models is considerably more costly than directed graph models.

To make these processes tractable, existing algorithms artificially constrain their undirected graph models. These constraints typically take the form of limits on the complexity of the model or conversions of the model to simpler forms. For example, the DEUM algorithm uses a univariate MRF model and its extension, Is-DEUM, employs a bivariate MRF model [42]. The MN-FDA and MN-EDA methods use junction graph and Kikuchi approximations to factorize the structure of their MRF models [40,41]. In effect, the MRF models are simplified or mixed with directed relations to reduce their computational demands.

Such constraints make it practical to learn and sample the MRF models. However, they make the models less flexible and the overall search process potentially less effective. Therefore, the main contribution of this paper is to develop mechanisms for learning and sampling a Markov random field, potentially resulting in more efficient search.

2.3. Markov random fields

A Markov random field defines the joint probability distribution of a set of random variables in terms of *local characteristics*, i.e. joint or conditional probability distributions of subsets of the random variables. Let $\{X_1, \dots, X_n\}$ be a set of finite random variables and let $\mathbf{X}$ be the space of *configurations* of $\{X_1, \dots, X_n\}$, i.e.

$$\mathbf{X} = \prod_{i=1}^{n} X_i. \tag{3}$$

A probability measure, P, is a *random field* with regard to $\mathbf{X}$ if it is strictly positive and normalized, i.e.

$$\begin{cases} P(\mathbf{x}) > 0, \quad \forall \mathbf{x} \in \mathbf{X} \\ \sum\limits_{\mathbf{x} \in \mathbf{X}} P(\mathbf{x}) = 1. \end{cases} \tag{4}$$

A *neighborhood system*, ∂, defines a set of neighbors for each random variable such that

$$\begin{cases} i \notin \partial(i) \\ i \in \partial(j) \iff j \in \partial(i). \end{cases} \tag{5}$$

The Markov property then induces an MRF on P given ∂ such that

$$P(x_i | x_j, i \neq j) = P\left(x_i | x_k, k \in \partial(i)\right). \tag{6}$$

The neighborhood system and Markov property together imply that each random variable is statistically dependent on all random variables inside its neighborhood, and statistically independent of all random variables outside its neighborhood (given the neighborhood set). The neighborhood system can be interpreted as an undirected graph, such as the one in Fig. 2.

MRFs can be equivalently defined in terms of a set of potential functions over maximal cliques in the neighborhood system. However, the MRF learning and sampling mechanisms described in Sections 3.2 and 3.3 are more readily understood in terms of the above definitions. In particular, this definition allows the MRF to be readily used as an EDA model. First, it is defined in terms of conveniently computable conditional probabilities, like those commonly used in EDAs. Second, each gene x_i is conveniently modeled by a random variable X_i, which makes the candidate solution and configuration spaces equivalent ($\mathbf{X}$). Third, the neighbor relation can apply to any pair of random variables; there are no ad hoc constraints on the structure of the neighborhood system. This paper further shows that the MRF neighborhood system can be learned using a metric for statistical dependence, χ^2, which will be described next.

2.4. Pearson's χ^2 test

Statistical hypothesis tests are commonly used to measure the significance of statistical comparisons. For example, Student's t-test [37] determines if the means of two Gaussian distributions are statistically distinct. The test results in a *confidence level* (or *p-value*) that is usually compared with a chosen threshold, e.g. 0.95 or 0.99, to decide whether the difference in means is statistically significant.

The confidence level can also be used as a direct measure of statistical dependence. This approach leads to Pearson's χ^2 test, which is a non-parametric statistical hypothesis test measuring the degree of similarity between two distributions of nominal (orderless) data. When used with data sampled from nominal random variables, the test measures the degree of dependence among the random variables. Pearson's χ^2 test compares two frequency distributions (FDs), typically an observed FD and an expected FD:

$$\chi^2 = \sum_{\mathbf{x}\in\mathbf{X}} \frac{\left(\mathrm{F_{obs}}(\mathbf{x}) - \mathrm{F_{exp}}(\mathbf{x})\right)^2}{\mathrm{F_{exp}}(\mathbf{x})}. \tag{7}$$

If $\mathrm{F_{exp}}$ is constructed to represent the null hypothesis (of independence) among a set of random variables $\{X_m, \ldots, X_t\}$ and differs notably (confidence level near 1) from $\mathrm{F_{obs}}$, then $\mathrm{F_{obs}}$ is assumed to be the product of dependencies among $\{X_m, \ldots, X_t\}$.

Like many statistical hypothesis tests, the confidence level of Pearson's χ^2 test depends on the degrees of freedom of the hypothesis in addition to the χ^2 value. Degrees of freedom are generally defined as the number of independent parameters of a system, i.e. the number of estimate variables minus the number of independent constraints on those variables. Consider an example contingency table showing the joint frequency distribution of two nominal random variables, X_i and X_j, each with four possible values:

X_i	X_j			
	α	β	γ	δ
α	4	5	10	3
β	18	5	3	9
γ	9	7	7	2
δ	34	8	12	15

Modeling this frequency distribution requires 16 estimate variables, one for each cell of the table. The sum of samples in each row and column provides eight constraints on those estimate variables. However, only seven of the constraints are independent, since the total number of samples can be obtained by summing the number of samples across all rows or, equivalently, across all columns. There are therefore $16 - 7 = 9$ degrees of freedom in the model, i.e. given

More generally, the degrees of freedom, δ, for any two-dimensional contingency table with r rows and c columns can be calculated as

$$\delta = rc - (r + c - 1) = rc - r - c + 1 = (r - 1)(c - 1). \tag{8}$$

This derivation is valid when the frequency distribution is well covered, and extends naturally to joint frequency distributions of more than two random variables.

However, derivation (8) overestimates the true degrees of freedom of systematically sparse frequency distributions, like those typically occurring during population-based search. Consider the following modification of the previous contingency table:

X_i	X_j			
	α	β	γ	δ
α	4	5	0	0
β	18	5	0	0
γ	9	7	0	0
δ	34	8	0	0

In this example, two possible values for X_j, γ and δ, are missing from the frequency distribution. In effect, insufficient sampling has systematically reduced the domain of X_j. Modeling the frequency distribution no longer requires 16 estimate variables, but only eight. The other "missing" eight no longer need to be modeled because their value is systematically zero. The degrees of freedom of the model is reduced accordingly, from nine to three. Any comparisons between frequency distributions that are *both* missing the two rightmost columns should be based on three degrees of freedom, otherwise the χ^2 test might fail to identify a correlation in the data. This solution is included in the MARLEDA method (described in the next section), making it possible to use χ^2 to construct an accurate MRF despite sparse sampling.

3. The MARLEDA method

The MARLEDA search algorithm is designed to overcome the limitations of previous EDAs. By using a more general and flexible model, learned in an efficient way, MARLEDA has the potential to be a more effective search method. MARLEDA still follows the procedural framework for EDAs outlined in Section 2.2. The following sections detail the major components of MARLEDA, with parameters of the MARLEDA algorithm shown in **highlighted** font, and analyze the computational complexity of the system.

3.1. Selection

Each chromosome in MARLEDA's population is composed of a set of genes, each corresponding to a parameter of the combinatorial optimization task. The number of genes is the same for all chromosomes and fixed at **Genes**. All concrete statistics regarding genes are calculated using members of the current population, $\mathcal{P}(t)$, where $|\mathcal{P}(t)| =$ **PopSize**. To bias the statistics toward favorable population members, a subcollection $\mathcal{P}'(t) \subseteq \mathcal{P}(t)$ of the current population is chosen via tournament selection [12]. The top **Parents** · **PopSize** (where **Parents** $\in (0, 1]$) high-fitness chromosomes compete in tournaments of size **TournSize**. A total of **PopSize** chromosomes are selected for membership in $\mathcal{P}'(t)$.

3.2. The MRF model

MARLEDA uses a set of nominal random variables, $\{X_1, \ldots, X_n\}$, to model the nominal genes, $\{x_1, \ldots, x_n\}$, of a combinatorial optimization task. Statistical dependencies among the random variables, and therefore among the genes, are recorded in a neighborhood system, thus forming an MRF model.

The neighbor relation between any two random variables is grounded in an observable statistical dependence within the members of $\mathcal{P}'(t)$. Like many EDAs, MARLEDA tests for these dependencies to learn its model. Consider a "partial" MRF whose neighborhood system does not yet fully capture all the observable dependencies. Let X_i and X_j be non-neighbors in the current neighborhood system, each with their own "partial" set of neighbors. If a dependence between X_i and X_j is observable, the neighborhood system should be updated to make X_i and X_j neighbors. Conversely, if X_i and X_j began as neighbors and a dependence is not observable, they should become non-neighbors.

Pearson's χ^2 test is used to compute the confidence level of dependence between two genes. The two frequency distributions compared are

$$\mathrm{F}_{\mathrm{obs}}(i, j) = \mathrm{F}\left(x_i, x_j | x_k, k \in \partial(i)\right) \quad \text{and} \tag{9}$$

$$\mathrm{F}_{\mathrm{exp}}(i, j) = \frac{\mathrm{F}\left(x_i | x_k, k \in \partial(i)\right) \cdot \mathrm{F}\left(x_j | x_k, k \in \partial(i)\right)}{|\mathrm{F}\left(x_k, k \in \partial(i)\right)|}, \tag{10}$$

where $\mathrm{F}_{\mathrm{obs}}$ is the joint frequency distribution of x_i and x_j, given x_i's neighbors, as observed within $\mathcal{P}'(t)$. $\mathrm{F}_{\mathrm{exp}}$ is the joint frequency distribution of x_i and x_j, given x_i's neighbors, under the assumption that x_i and x_j are independent, i.e. the product of the marginal FDs of x_i and x_j as observed within $\mathcal{P}'(t)$. (Note: when using binary chromosomes χ^2 is adjusted using Yates' correction; [47].)

Intuitively, the above procedure measures how much information is gained by making x_i and x_j neighbors. If $\mathrm{F}_{\mathrm{obs}}$ and $\mathrm{F}_{\mathrm{exp}}$ differ, x_i depends on x_j and the two should be made neighbors. Similarly, if x_i and x_j began as neighbors, the gain in remaining neighbors can be computed by temporarily removing their neighbor status and performing the same test. (Note: although the MRF neighbor relation is symmetrical, $\mathrm{F}_{\mathrm{obs}}$ and $\mathrm{F}_{\mathrm{exp}}$ are not symmetrical about x_i and x_j. Ideally, the reciprocal test should also be performed, with only two successes or two failures suggesting a change in neighbor status. A single test is performed in MARLEDA for simplicity, and it works well in practice.)

MARLEDA constructs the MRF neighborhood system via a greedy search approach. Beginning with a trivial neighborhood system, $\partial(i) = \emptyset$. At each iteration **ModelAdds** pairs of non-neighbors are tested. If the confidence level of the pair is at least **ModelAddThresh**, the model is updated to make the pair neighbors. Similarly, **ModelSubs** pairs of neighbors are tested, and if the confidence level is below **ModelSubThresh** the pair is made non-neighbors. The order of all tests is randomized.

The two threshold values, **ModelAddThresh** and **ModelSubThresh**, represent how strict of the neighbor relation is within the MRF neighborhood system. While statistical hypothesis tests are typically used with very strict confidence levels, 0.95, 0.99, or higher, in MARLEDA it is possible to use more relaxed confidence levels, since even partial correlations in the data are beneficial to the sampling process. During preliminary experimentation it was determined that **ModelAddThresh** $= 0.8$ and **ModelSubThresh** $= 0.6$ work well across a spectrum of optimization tasks.

As mentioned in Section 2.4, a degrees of freedom term, $\delta(i, j)$, must be computed for each χ^2 test between a pair of genes x_i and x_j. Let $\mathbf{A}$ be the set of alleles possible for each (and all) genes. Derivation 8 in Section 2.4 defines the degrees of freedom parameter as

$$\delta(i, j) = (|\mathbf{A}| - 1)^{|\partial(i)|+2}. \tag{11}$$

The above calculation is adjusted in two situations. First, when one or more alleles for a gene are not represented in $\mathcal{P}'(t)$, that gene no longer contributes a full $|\mathbf{A}| - 1$ degrees to the above calculation. The actual number of alleles represented for each gene and adjusted degrees of freedom are

$$\mathbf{A}(i) = \left\{a \in \mathbf{A} : \exists \mathbf{x} \in \mathcal{P}'(t) : x_i = a\right\} \tag{12}$$

$$\delta(i, j) = \prod_{k \in \{i,j\} \cup \partial(i)} \max(|\mathbf{A}(k)| - 1, 1). \tag{13}$$

Second, the degrees of freedom term naturally grows exponentially in the size of the neighborhood. The minimum χ^2 value necessary to demonstrate dependence grows exponentially as well. However, the candidate solution space is generally poorly represented by the chromosomes in $\mathcal{P}'(t)$, i.e. $|\mathcal{P}'(t)| \ll |\mathbf{X}|$. There are a finite number of samples in $\mathcal{P}'(t)$ from which F_{obs}, F_{exp}, and consequently χ^2, are computed. This constraint places an upper limit on the computable χ^2 value, thus truncating its exponential growth and making it increasingly difficult to expand the neighborhood system. This restriction provides a natural limit to the growth of the neighborhood system, helping prevent learning of unfounded dependencies.

However, when $|\mathbf{A}| = 2$ the degrees of freedom calculation trivially collapses to one. Instead of exponential growth in the size of a gene's neighborhood, there is no growth. The χ^2 term, however, continues to grow, and this mismatch between degrees of freedom and χ^2 makes it too easy to "pass" Pearson's χ^2 test. Consequently, neighborhoods can expand and become maximal without true statistical support. To combat this problem, when $|\mathbf{A}| = 2$ the computed degrees of freedom is artificially inflated to restore exponential growth as follows

$$\delta(i, j) = \left\lfloor \prod_{k \in \{j\} \cup \partial(i)} \max(|\mathbf{A}(k)| - 0.25, 1) \right\rfloor. \tag{14}$$

All genes but one, x_i, contribute at most 1.75 "virtual" degrees to the calculation. Values ranging from 1.5 to 1.75 worked well in preliminary experiments.

Pearson's χ^2 test provides a convenient method for learning the local neighbor relations of a Markov random field. Though this greedy construction procedure is not guaranteed to build an optimal MRF, it captures the high-fitness features of the population. Constructing the model is only the first step; the model must then be sampled to combine the high-fitness features into new chromosomes.

3.3. Generating chromosomes

New chromosomes are created in MARLEDA by sampling the MRF model. Sampling is performed via a Markov chain Monte Carlo process:

```
1. $\mathbf{x}^{\text{new}} \leftarrow$ a random chromosome from $\mathcal{P}'(t)$.
2. Randomly select a gene $x_i^{\text{new}}$.
3. Compute $P\left(x_i | x_k, k \in \partial(i)\right)$.
4. $x_i^{\text{new}} \leftarrow$ sample from $P\left(x_i | x_k, k \in \partial(i)\right)$.
5. Unless termination criteria are met, return to step 2.
```

Ideally, the sampling process continues until the allele distribution of the new chromosome stabilizes. The number of iterations needed before convergence depends on the specifics of the joint probability distribution encoded by the MRF and thus may not be known a priori. However, a good rule of thumb is to allow the sampler to "burn-in" for at least several thousand iterations. In MARLEDA, the sampling process executes for a fixed number of iterations specified by parameter **MonteIters**. After sampling, genes are mutated with probability **Mutation**.

The complete random field on the configuration space $\mathbf{X}$ is not available, hence $P\left(x_i | x_k, k \in \partial(i)\right)$ is sometimes undefined. In such cases, a "relaxed" conditional probability is used. In effect, undefined regions of the configuration space are approximated by nearby well-defined regions. Under normal conditions

$$P\left(x_i | x_k, k \in \partial(i)\right) = \frac{\mathrm{F}\left(x_i | x_k, k \in \partial(i)\right)}{|\mathrm{F}\left(x_k, k \in \partial(i)\right)|}. \tag{15}$$

When $\mathrm{F}\left(x_k, k \in \partial(i)\right)$ contains no samples, a first-order relaxation is calculated, incorporating all subsets of $\partial(i)$ of size $|\partial(i)| - 1$

$$P^{(1)}\left(x_i | x_k, k \in \partial(i)\right) = \frac{\bigcup_{\partial'(i) \subset \partial(i)} \mathrm{F}\left(x_i | x_k, k \in \partial'(i), |\partial'(i)| = |\partial(i)| - 1\right)}{\sum_{\partial'(i) \subset \partial(i)} |\mathrm{F}\left(x_k, k \in \partial'(i), |\partial'(i)| = |\partial(i)| - 1\right)|}. \tag{16}$$

If the first-order relaxation is also undefined, the second-order relaxation incorporating all subsets of $\partial(i)$ of size $|\partial(i)| - 2$ is evaluated, and so on, until a valid probability distribution is found. In the worst case, the entire neighborhood $\partial(i)$ is ignored

$$P^{(|\partial(i)|)}\left(x_i | x_k, k \in \partial(i)\right) = P\left(x_i\right), \tag{17}$$

and the current sampling iteration degenerates to sampling from the marginal distribution of x_i, as univariate EDAs do.

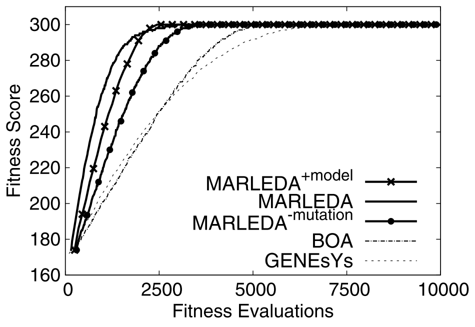

Fig. 3. Learning curves of the median best population member in 100 independent trials for an instance of OneMax with 300 bits. All algorithms easily find the optimum solution, with MARLEDA demonstrating a moderate advantage in learning time.

3.4. Replacement

The $\textbf{Replaced} \cdot \textbf{PopSize}$ (where $\textbf{Replaced} \in (0, 1]$) chromosomes in the population with the lowest fitness are replaced by newly created chromosomes. This step implements an elitist strategy by which the top $(1 - \textbf{Replaced}) \cdot \textbf{PopSize}$ chromosomes are preserved between iterations.

The processes described in this section are the essential components of the MARLEDA algorithm. In the next section their performance is tested on several combinatorial optimization tasks, and found to perform well compared to the standard GA and an advanced EDA.

4. Experimental results

In this section MARLEDA's performance is tested on four combinatorial optimization tasks. The first three tasks, OneMax, deceptive trap functions, and the Rosenbrock function, are standard benchmark tasks for optimization algorithms. The last task, optimization of Ising spin glass systems, is a difficult optimization task in statistical physics. MARLEDA's performance is compared against two readily available optimization suites: the GENEsYs [2] implementation of a standard GA, to provide an expected performance baseline, and the Bayesian optimization algorithm [34, BOA] with decision graphs, to provide a comparison with a state-of-the-art search method.

In addition to the GENEsYs, BOA, and MARLEDA algorithms, two variants of MARLEDA are evaluated where applicable. The first MARLEDA variant, MARLEDA$^{-\text{mutation}}$, disables mutation. Most EDAs lack a traditional mutation operator, thus MARLEDA$^{-\text{mutation}}$'s performance illuminates mutation's contribution to MARLEDA and its relevance in EDA methods. The second variant, MARLEDA$^{+\text{model}}$, disables MRF learning and uses a fixed MRF neighborhood system based on the known domain structure of the task. By removing the need for model learning, this variant evaluates MARLEDA's ability to successfully exploit an ideal model.

In order to fairly gauge each algorithm's search capability, all algorithms were subject to a limit on the number of fitness function evaluations permitted during each experiment. So that each algorithm could best use this limited resource, each algorithm's parameters were tuned to optimize final solution quality. Tuning proceeded via simple gradient ascent: beginning with reasonable or documented parameter settings, slightly perturbed settings were evaluated, continuing until no further improvement in solution quality was achieved. In cases where multiple parameter settings resulted in equivalent solution quality, preference was given to those producing more rapid progress. While this procedure does not guarantee that the resulting parameter settings documented in this section are optimal, they reliably lead to good performance. All algorithm parameter settings are included in Appendix A.

4.1. OneMax

The OneMax problem is an extremely simple optimization problem for binary strings. The goal is to maximize the number of "on" bits, i.e. maximize

$$f(\mathbf{x}) = \sum_{i=1}^{n} x_i.$$

There are no dependencies among genes for this task, thus OneMax is not a particularly interesting problem for EDAs. It is included only to provide a performance baseline for the next optimization problem, deceptive trap functions.

In this experiment, binary strings of 300 bits were optimized, with each algorithm limited to 10,000 fitness function evaluations. Fig. 3 shows the median best fitness score during the course of evolution over 100 independent trials of each algorithm. While all algorithms easily find the optimal OneMax solution, it is interesting to note the effect of mutation on the three instances of MARLEDA. The parameter tuning process discovered that standard MARLEDA performed best

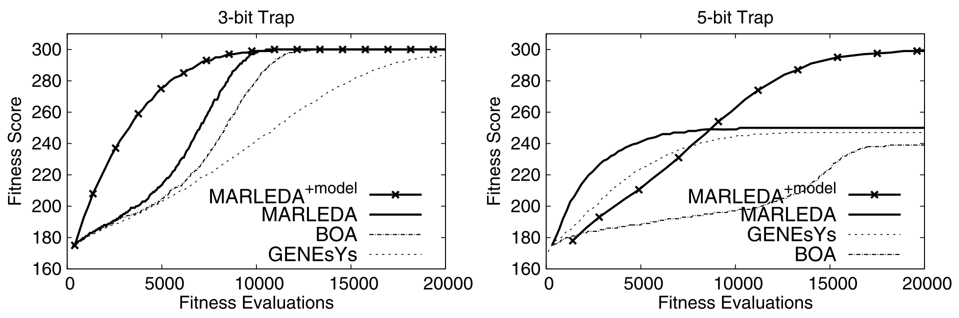

Fig. 4. Learning curves of the median best population member in 100 independent trials for 300-bit instances of deceptive trap function. The differences in median best fitness at the end of evolution are statistically significant (as computed by the Wilcoxon rank-sum test with a confidence greater then 99%) in the 5-bit trap domain for all algorithm pairs except GENEsYs/MARLEDA. In the 3-bit trap scenario, the modeling capabilities of the EDAs allow then to outperform the standard GA. In the 5-bit trap scenario, the EDAs only achieve marginal solution gains over the GA. However, when provided an accurate model of the domain, MARLEDA$^{+\text{model}}$ can readily find the optimal solution in the majority of trials. As domain complexity increases and the potential to become trapped in local optima rises, utilizing an accurate model is critically important.

with a small amount of mutation. Without mutation, MARLEDA$^{-\text{mutation}}$'s rate of progress is slightly worse than that of standard MARLEDA. Interestingly, when provided the true univariate model (i.e. no dependencies among genes) of the domain, MARLEDA$^{+\text{model}}$ performed best without mutation.

MARLEDA$^{+\text{model}}$ is able to find the optimum solution with several hundred fewer evaluations than MARLEDA. However, the rate of progress is initially worse than that of MARLEDA. The coincidental dependencies between bits that standard MARLEDA identifies initially boost its performance but then hinder it as “off” bits are mistakenly preserved. Without an ideal model, mutation contributes to MARLEDA's performance on this, albeit simple, task.

4.2. Deceptive trap functions

Deceptive trap functions [9] are multimodal functions designed such that local gradient information will tend to lead optimization algorithms toward local optima and away from global optima. Search algorithms must therefore have the capacity to escape local optima if there is to be any chance of identifying global optima. For EDAs, this means learning the deceptive elements in order to avoid local “trap” optima.

A standard class of trap function is a variant of OneMax where blocks of bits have two local optima, only one of which contributes to the global optimum of the function. Let α be the block size of the trap. Then

$$u(\mathbf{x},k) = \sum_{i=\alpha(k-1)+1}^{\alpha k} x_i,$$

$$f_\alpha(\mathbf{x},k) = \begin{cases} \alpha - u(\mathbf{x},k) - 1 & \text{if } u(\mathbf{x},k) < \alpha \\ \alpha & \text{if } u(\mathbf{x},k) = \alpha, \end{cases} \quad \text{and}$$

$$f(\mathbf{x}) = \sum_{i=1}^{\frac{n}{\alpha}} f_\alpha(\mathbf{x},i).$$

Within each block of α bits, only one of the 2^α possible bit combinations is part of the global optimum. All other bit combinations guide search toward local trap optima. As the trap size increases the components of the global optimum become more rare and thus more difficult for search to discover. For small trap sizes this problem presents an interesting challenge for optimizers, which must weigh the abundant evidence of trap optima against the scarce evidence of the global optimum.

In this experiment, binary strings of 300 bits were optimized for traps of size three bits and five bits. Each algorithm was limited to 20,000 fitness function evaluations. Fig. 4 shows the median best fitness score during the course of evolution over 100 independent trials of each algorithm. In the case of traps of size three, all algorithms routinely discover the optimum solution, though the rate of progress is notably slower than in OneMax.

For traps of size five, all algorithms ignorant of the true structure of the domain are unable to find the globally optimal solution. The algorithms instead identify trap optima, where each block of five bits has converged to its second local optimum. Resulting fitness scores are at least $300 \cdot \frac{4}{5} = 240$, because a few of the blocks have been correctly optimized by chance. However, when MARLEDA$^{+\text{model}}$ is provided the true structure of the domain, where each bit is dependent on the four other bits in its block, avoiding the trap optima and identifying the global optimum ceases to be a problem.

The proportionally slower improvement of MARLEDA$^{+\text{model}}$ is an artifact of the larger population used in that experiment, necessitating more fitness function evaluations per generation. With a greater trap size, more chromosomes are needed per generation to accurately cover each trap block. This observation is consistent with previous work where the necessary

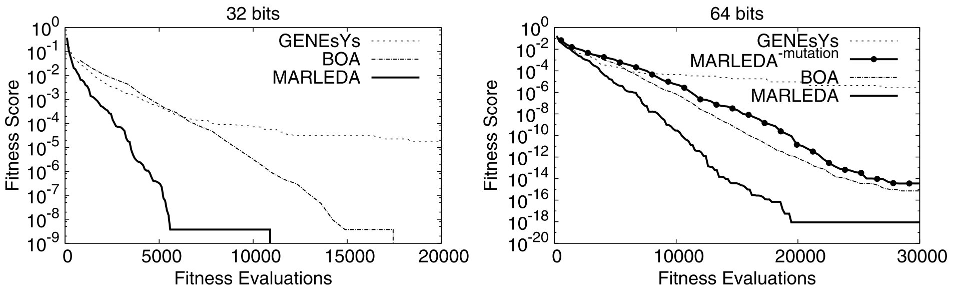

Fig. 5. Learning curves of the median best population member in 100 independent trials for Rosenbrock instances of 32 and 64 bits. The differences in median best fitness at the end of evolution are statistically significant (as computed by the Wilcoxon rank-sum test with a confidence greater then 99%) in the 32-bit domain for GENEsYs/BOA, GENEsYs/MARLEDA$^{-\text{mutation}}$, and GENEsYs/MARLEDA. Differences are statistically significant in the 64-bit domain for all algorithm combinations except MARLEDA$^{-\text{mutation}}$/BOA. In this domain of relatively low deception, the EDAs are able to learn and exploit domain structure, allowing them to find results vastly superior to more localized GA search.

population size was shown to increase exponentially with the size of the trap [13,16]. Consequently, the ignorant algorithms would be likely to find the optimum solution if permitted additional fitness function evaluations by an order of magnitude or more. Without such evaluations, the parameter tuning process discovered that those algorithms with mutation operators (GENEsYs and MARLEDA) were best served by maximizing the total number of generations of evolution, by minimizing population size, and relying on mutation for search. This process resulted in some unintuitive parameter shifts, such as the *decrease* in population size for MARLEDA from 450 on traps of size three to 300 on traps of size five. Similar consequences of the fitness function evaluation limit occur on the following optimization tasks as well.

When provided sufficient fitness evaluations, the EDAs learned and exploited the domain structure to increase search efficiency. However, when evaluations were limited, as was forced upon MARLEDA and BOA in the 5-bit trap experiment, MARLEDA's performance degraded the most gracefully. As in the OneMax problem, mutation contributes to MARLEDA's search capability. However, on this deceptive task mutation remains useful even when the true domain structure is known. The results of this and the OneMax experiment suggest that mutation is a useful component in EDA methods.

4.3. The Rosenbrock function

The Rosenbrock function [39] is a numerical function definable for any number of dimensions. In two dimensions, the goal is to minimize the function

$$f(x, y) = (1 - x)^2 + 100\left(y - x^2\right)^2.$$

The function has a global minimum at $f(1, 1) = 0$, but when x and y are encoded in binary the resulting discretization of the domain produces many local minima near the curve $y = x^2$. In addition, there are many overlapping low-order dependencies among the bits of x and y. The parameters x and y are encoded in a binary chromosomes as fixed-point numbers in the range $[0, 4]$ whose values are then translated to the target domain $[-2, 2]$. Since this domain is much less deceptive than trap functions, EDAs should be able to exploit the domain structure to perform search efficiently and distinguish themselves from GAs.

The experiments include chromosomes of 32 bits (16 bits each for x and y) and 64 bits (32 bits each for x and y). Each run of the experimental algorithms was limited to 20,000 fitness function evaluations. Fig. 5 shows the median best fitness score found by each algorithm over 100 independent trials. Both MARLEDA and BOA perform well on this task, with MARLEDA demonstrating a distinct advantage in both learning rate and final quality. In the 32-bit domain, MARLEDA's and BOA's median best chromosome quickly approaches the second lowest fitness score possible, represented by the flat region of the respective fitness curves near 15,000 evaluations. This score corresponds to the most deceptive of the local minima in the domain, with a Hamming distance of 29 bits from the global minimum. The vertical segment of the fitness curves shows the point where the median best chromosome is the global optimum. Due to the logarithmic scale of the graphs, the learning curves appear to "fall off" the graph.

The 64-bit domain shows even greater separation between MARLEDA, BOA, and GENEsYs. The encoding of the two numerical coordinates presents a significant hurdle for local search methods such as GENEsYs. While the fitness landscape of the Rosenbrock function is smooth in numerical space, it is quite rough in configuration space. A small change in numerical space may result in a large change in configuration space, and vice versa. The structure-exploiting approach of the EDAs allowed them to find better solutions, with MARLEDA performing significantly better than BOA. Its ability to exploit the structure of the configuration space to correctly optimize many bits at once is crucial to good performance.

It is interesting to note that without mutation on 64-bit Rosenbrock, MARLEDA$^{-\text{mutation}}$ performs very similarly to BOA, which also has no mutation operator. Mutation proves to be useful yet again.

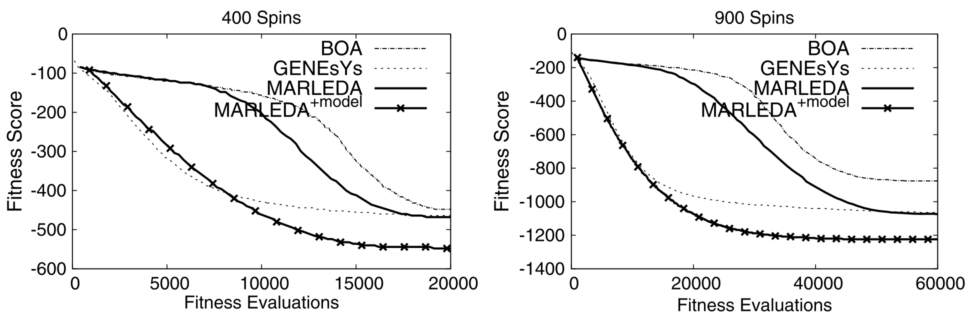

Fig. 6. Representative learning curves of the median best fitness in 100 independent trials for an instance of an Ising spin glass system of 400 spins and 900 spins. The differences in median best fitnesses at the end of evolution are statistically significant (as computed by the Wilcoxon signed-rank test, with a confidence greater than 99%) for all algorithm combination except GENEsYs/MARLEDA in both the 400 spin and 900 spin domains. MARLEDA's Markov random field model naturally represents the relationships in the spin glass domain, resulting in improved performance compared to BOA.

4.4. Ising spin glasses

Ising spin glasses [21] are a model of magnetic materials developed in statistical physics and have been extensively used as EDA benchmarks. A set of *spins*, $\{s_1, \dots, s_n\}$, exist in one of two states, $+1$ or -1, and each spin is coupled to a set of neighboring spins. An instance of an Ising spin glass system is defined by a set of coupling constants, $\{J_{i,j} : i, j \in \{1, \dots, n\}\}$, that encapsulate the neighbor relationships. A coupling constant $J_{i,j}$ is non-zero if s_i and s_j are neighbors. In these experiments, coupling constants are restricted to values of $+1$, 0, and -1. The goal is to find a set of spin states that minimizes the Hamiltonian of the system:

$$H = -\sum_{i,j=1}^{n} J_{i,j} s_i s_j .$$

Minimizing the Hamiltonian implies that neighboring spins should tend to exist in the same state if their coupling constant is $+1$ and in differing states if their coupling constant is -1. However, conflicting coupling constants among groups of spins prevent this rule from being applied fully. Consequently, groups of spins may have locally optimal states that are quite different from the globally optimal spin states, or *ground states*. This property makes spin glass systems a difficult partially deceptive search task. However, since each spin's contribution to the Hamiltonian directly depends on only the other spins with which it shares a non-zero coupling coefficient, there is an obvious analogue between non-zero coupling coefficients and an MRF neighborhood system.

To test the scalability of MARLEDA, the Ising spin glass systems used in these experiments contain the most parameters of any optimization experiment presented in this paper. Though there is plenty of domain structure for EDAs to exploit, the volume of information will likely make learning an effective model difficult. Five hundred instances of Ising spin glass systems with 400 spins and 900 spins were randomly generated. Each instance was arranged in a two-dimensional square lattice (20×20 or 30×30) with periodic boundary conditions. Each spin was neighbored by the adjacent spin above, below, to the left, and to the right. The coupling constants for neighboring spins were uniformly sampled from $\{+1, -1\}$, with all other coupling constants set to 0, thus each instance was drawn from the region of spin glass system space known to contain a comparatively high density of difficult instances [25,26]. The set of spin states was encoded in a binary chromosome with one bit per spin state.

Each run of the experimental algorithms was limited to 20,000 fitness function evaluations in the 400 spin domain and 60,000 fitness function evaluations in the 900 spin domain. Each algorithm was run 100 times on each of the 1000 randomly generated spin glass instances. Fig. 6 shows the median best fitness score over the 100 trials found by each algorithm on *one* particular instance. Nearly all instances resulted in similar learning curves, thus Fig. 6 is representative.

All algorithms ignorant of the true domain structure discover solutions of nearly the same quality, with the exception of BOA on system of 900 spins. Unlike the Rosenbrock function, two-dimensional lattice spin glass systems are amenable to local search techniques, shown by GENEsYs's good performance. However, local search does not lead to global optima. The optimal fitness score for each instance was determined using the Spin Glass Ground State Server at the University of Köln [23]. The solutions routinely discovered by MARLEDA and GENEsYs have fitness scores only 80%–85% of optimal. The deceptive qualities of this domain were not completely overcome. The differences in final performance between MARLEDA and BOA represent 4%–7% of the optimal fitness score in the 400 spin domain and 15%–18% in the 900 spin domain.

The EDAs exhibit a curiously slow start, which is caused by poor initial learned models. The complexity of the domain coupled with the relatively large number of parameters make it difficult for the EDAs to identify dependencies among parameters. The learned models therefore did not promote high-fitness chromosomes during sampling and tended to reproduce low-fitness aspects of the population. However, once the models were sufficiently refined solution quality improved rapidly.

In contrast to BOA and standard MARLEDA, MARLEDA$^{+\text{model}}$ performed very well. The lattice structure of the spin glass systems forms a natural MRF neighborhood system. When provided with this system, MARLEDA$^{+\text{model}}$ was able to routinely discover the ground state of systems of 400 spins and come to within 1%–2% of the ground state of systems of 900 spins. Though these experiments are constructed differently, the performance results are consistent with the experiments of [44]; exploiting the structure of spin glass systems is key to solving them efficiently. Using an accurate model, MARLEDA$^{+\text{model}}$ successfully scaled up to this large optimization tasks. This result also suggests that improving MARLEDA's model learning procedure is the single greatest opportunity for increasing MARLEDA's effectiveness.

5. Discussion and future work

The experiments presented in this paper show that the Markov random field model employed by MARLEDA is a promising alternative to the DAG-based methods commonly used in EDAs. It allows MARLEDA to perform better than BOA in a variety of test domains.

Good performance on the Ising spin glass domain is particularly encouraging because it is a difficult real world problem. Moreover, the comparison of GAs and EDAs is particularly insightful for two reasons. First, it suggests that it may be possible to improve performance of EDAs by combining them with local search techniques, as suggested by [35]. Second, it demonstrates the cost associated with building the probabilistic model. Both MARLEDA and BOA must discover reliable models before evolution can progress rapidly. This process takes place during the first 8,000–13,000 fitness evaluations, in the 400 spin domain, or the first 20,000–30,000 fitness evaluations, in the 900 spin domain, while the fitness improves only slightly (Fig. 6). Once the model is sufficiently accurate, performance improves very rapidly, faster than that of GENEsYs.

MARLEDA's MRF model, which naturally represents the dependencies of the spin glass domain, enables MARLEDA to learn an effective statistical model more quickly than BOA. Both algorithms initially optimize small, localized groups of spins before addressing the larger system. This progression requires each algorithm to learn a model organized around small groups of spins. As evolution progresses, the model must expand to allow optimization across larger sets of spins, forming numerous intricate sets of dependencies. For BOA's directed graph model, this process causes many learned dependencies supporting locally good fitness to be lost, either to preserve the acyclic property of the graph or to change the direction of dependency. MARLEDA's model, on the other hand, does not need to destroy learned dependencies in favor of new ones, allowing more rapid optimization across the entire system.

The MARLEDA approach can be further improved in two main ways. First, the MRF neighborhood system can be constructed in a more principled manner. For example, at each iteration of the algorithm a set of new neighbor relations forming a (partial) minimum spanning tree across non-neighbor random variables could be added to the neighborhood system. Such an approach would bias the model learning process towards global connectivity, potentially improving performance on some tasks.

Second, the MRF could be sampled more efficiently and more accurately. The current Monte Carlo technique does not scale well as population size increases. Alternatives include using the Metropolis–Hastings algorithm [18] and sampling via the equivalent Gibbs random field [6,14]. Such methods should provide better computational performance and sampling accuracy, resulting in a more powerful algorithm overall.

EDAs' ability to tackle difficult combinatorial optimization problems makes them strong candidates for application to real-world problems, such as sensor network layout or schedule optimization, in the near future. Not only can EDAs exploit known structure and relationships within problem domains, but they can also reveal new structure as they operate, possibly providing the experimenter with new insight into the domain. Domains of daunting complexity, such as those in computational biology, could greatly benefit from techniques that inherently learn and exploit the underlying statistical structure of the domain. The volumes of biological data regarding molecular sequences and structures of proteins, DNA, and RNA are particularly promising domains. Applying MARLEDA to molecular structure prediction is therefore a most interesting direction for future work.

6. Conclusions

This paper presents the Markovian Learning Estimation of Distribution Algorithm, a new estimation of distribution algorithm based on Markov random fields. An MRF model allows MARLEDA to efficiently handle optimization tasks involving complex structural dependencies. Experiments with four combinatorial optimization tasks suggest that MARLEDA can indeed successfully learn and use such a model, resulting in improved computational search. The approach should be particularly powerful in complex optimization tasks in nature, such as those in computational biology.

Acknowledgements

This research was supported in part by NSF under grant EIA-0303609.

Appendix A.

The following algorithm parameters were used for the OneMax experiments:

GENEsYs: population size = 200, full population selection, uniform crossover, elitism, Whitley rank selection with $\alpha = 2.0$, mutation rate = 0.004, crossover rate = 1.0, and generation gap = 1.0.
BOA: population size = 200, offspring percentage = 10, tournament size = 1, and max incoming links = 1.
MARLEDA: **PopSize** = 150, **Parents** = 0.6, **TournSize** = 3, **ModelAdds** = 3000, **ModelAddThresh** = 0.8, **ModelSubs** = 2000, **ModelSubThresh** = 0.6, **MonteIters** = 1500, **Mutation** = 0.01, and **Replaced** = 0.1.
MARLEDA$^{-\text{mutation}}$: **PopSize** = 300, **Parents** = 0.65, **TournSize** = 2, **ModelAdds** = 3000, **ModelAddThresh** = 0.8, **ModelSubs** = 2500, **ModelSubThresh** = 0.6, **MonteIters** = 1500, and **Replaced** = 0.05.
MARLEDA$^{+\text{model}}$: **PopSize** = 250, **Parents** = 0.3, **TournSize** = 3, **MonteIters** = 750, **Mutation** = 0.0, and **Replaced** = 0.4.

The following algorithm parameters were used for the three-bit deceptive trap function experiments:

GENEsYs: population size = 250, full population selection, uniform crossover, elitism, Whitley rank selection with $\alpha = 2.0$, mutation rate = 0.0, crossover rate = 1.0, and generation gap = 1.0.
BOA: population size = 250, offspring percentage = 10, tournament size = 1, and max incoming links = 1.
MARLEDA: **PopSize** = 450, **Parents** = 0.85, **TournSize** = 3, **ModelAdds** = 3000, **ModelAddThresh** = 0.8, **ModelSubs** = 2000, **ModelSubThresh** = 0.6, **MonteIters** = 1500, **Mutation** = 0.0, and **Replaced** = 0.15.
MARLEDA$^{+\text{model}}$: **PopSize** = 400, **Parents** = 0.65, **TournSize** = 4, **MonteIters** = 1500, **Mutation** = 0.01, and **Replaced** = 0.6.

The following algorithm parameters were used for the five-bit deceptive trap function experiments:

GENEsYs: population size = 100, full population selection, uniform crossover, elitism, Whitley rank selection with $\alpha = 2.0$, mutation rate = 0.005, crossover rate = 0.5, and generation gap = 1.0.
BOA: population size = 600, offspring percentage = 10, tournament size = 5, and max incoming links = 3.
MARLEDA: **PopSize** = 300, **Parents** = 0.1, **TournSize** = 3, **ModelAdds** = 3000, **ModelAddThresh** = 0.8, **ModelSubs** = 2000, **ModelSubThresh** = 0.6, **MonteIters** = 1500, **Mutation** = 0.015, and **Replaced** = 0.65.
MARLEDA$^{+\text{model}}$: **PopSize** = 1400, **Parents** = 0.7, **TournSize** = 3, **MonteIters** = 2500, **Mutation** = 0.005, and **Replaced** = 0.5.

The following algorithm parameters were used for the 32-bit Rosenbrock experiments:

GENEsYs: population size = 200, full population selection, uniform crossover, elitism, Whitley rank selection with $\alpha = 1.9$, mutation rate = 0.007, crossover rate = 0.9, and generation gap = 1.0.
BOA: population size = 800, offspring percentage = 80, tournament size = 2, and max incoming links = 10.
MARLEDA: **PopSize** = 400, **Parents** = 0.85, **TournSize** = 4, **ModelAdds** = 3000, **ModelAddThresh** = 0.8, **ModelSubs** = 2000, **ModelSubThresh** = 0.6, **MonteIters** = 1000, **Mutation** = 0.0, and **Replaced** = 0.7.

The following algorithm parameters were used for the 64-bit Rosenbrock experiments:

GENEsYs: population size = 220, full population selection, uniform crossover, elitism, Whitley rank selection with $\alpha = 1.9$, mutation rate = 0.005, crossover rate = 1.0, and generation gap = 1.0.
BOA: population size = 650, offspring percentage = 70, tournament size = 2, and max incoming links = 12.
MARLEDA: **PopSize** = 450, **Parents** = 0.8, **TournSize** = 4, **ModelAdds** = 3000, **ModelAddThresh** = 0.8, **ModelSubs** = 2000, **ModelSubThresh** = 0.6, **MonteIters** = 1000, **Mutation** = 0.03, and **Replaced** = 0.8.
MARLEDA$^{-\text{mutation}}$: **PopSize** = 700, **Parents** = 0.6, **TournSize** = 3, **ModelAdds** = 3000, **ModelAddThresh** = 0.8, **ModelSubs** = 2000, **ModelSubThresh** = 0.6, **MonteIters** = 1000, and **Replaced** = 0.8.

The following algorithm parameters were used for the 400-spin Ising spin glass experiments:

GENEsYs: population size = 100, full population selection, uniform crossover, elitism, Whitley rank selection with $\alpha = 2.0$, mutation rate = 0.006, crossover rate = 1.0, and generation gap = 1.0.
BOA: population size = 500, offspring percentage = 13, tournament size = 3, and max incoming links = 2.
MARLEDA: **PopSize** = 400, **Parents** = 0.85, **TournSize** = 4, **ModelAdds** = 3000, **ModelAddThresh** = 0.8, **ModelSubs** = 2000, **ModelSubThresh** = 0.6, **MonteIters** = 1000, **Mutation** = 0, and **Replaced** = 0.7.
MARLEDA$^{+\text{model}}$: **PopSize** = 900, **Parents** = 0.75, **TournSize** = 2, **MonteIters** = 2400, **Mutation** = 0.005, and **Replaced** = 0.25.

The following algorithm parameters were used for the 900-spin Ising spin glass experiments:

GENEsYs: population size = 130, full population selection, uniform crossover, elitism, Whitley rank selection with $\alpha = 2.0$, mutation rate = 0.0025, crossover rate = 1.0, and generation gap = 1.0.
BOA: population size = 700, offspring percentage = 75, tournament size = 3, and max incoming links = 6.

MARLEDA: **PopSize** = 700, **Parents** = 0.75, **TournSize** = 3, **ModelAdds** = 3500, **ModelAddThresh** = 0.8, **ModelSubs** = 2000, **ModelSubThresh** = 0.6, **MonteIters** = 1500, **Mutation** = 0, and **Replaced** = 0.9.

MARLEDA$^{\text{+model}}$: **PopSize** = 1000, **Parents** = 0.95, **TournSize** = 3, **MonteIters** = 2400, **Mutation** = 0.004, and **Replaced** = 0.1.

References

[1] M. Alden, A.-J. van Kesteren, R. Miikkulainen, Eugenic evolution utilizing a domain model, in: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO, Morgan Kaufmann, San Francisco, CA, 2002, pp. 279–286.

[2] T. Bäck, A User's Guide to GENEsYs 1.0, Department of Computer Science, University of Dortmund, 1992.

[3] S. Baluja, Population-based incremental learning: a method for integrating genetic search based function optimization and competitive learning, technical report CMU-CS-94-163, Carnegie Mellon University, Pittsburgh, PA, 1994.

[4] S. Baluja, S. Davies, Using optimal dependency-trees for combinatorial optimization: learning the structure of the search space, technical report CMU-CS-97-107, Carnegie Mellon University, 1997.

[5] S. Baluja, S. Davies, Fast probabilistic modeling for combinatorial optimization, in: Proceedings of the 15th National Conference on Artificial Intelligence and the 10th Annual Conference on Innovative Applications of Artificial Intelligence, AAAI Press, Menlo Park, CA, 1998.

[6] J. Besag, Spatial interaction and the statistical analysis of lattice systems, J. R. Stat. Soc. Ser. B. Stat. Methodol. 36 (1974) 192–236.

[7] G. Cooper, E. Herskovits, A Bayesian method for the induction of probabilistic networks from data, Mach. Learn. 9 (1992) 309–347.

[8] J. De Bonet, C. Isbell, P. Viola, MIMIC: finding optima by estimating probability densities, in: Advances in Neural Information Processing, vol. 9, MIT Press, Cambridge, MA, 1997.

[9] K. Deb, D. Goldberg, Analyzing deception in trap functions, in: Foundations of Genetic Algorithms, Morgan Kaufmann, San Francisco, CA, 1992, pp. 93–108.

[10] R. Etxeberria, P. Larrañaga, Global optimization with Bayesian networks, in: Proceedings of the Second Symposium on Artificial Intelligence, 1999, pp. 332–339.

[11] D. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison–Wesley, Reading, MA, 1989.

[12] D. Goldberg, K. Deb, A comparative analysis of selection schemes used in genetic algorithms, in: Foundations of Genetic Algorithms, Morgan Kaufmann, San Francisco, CA, 1990, pp. 69–93.

[13] D. Goldberg, K. Deb, J. Clark, Genetic algorithms, noise, and the sizing of populations, Complex Systems 6 (1992) 333–362.

[14] J. Hammersley, P. Clifford, Markov field and finite graphs and lattices, 1971, unpublished.

[15] G. Harik, Linkage learning via probabilistic modeling in the EcGA, technical report 99010, IlliGAL, University of Illinois at Urbana-Champaign, 1999.

[16] G. Harik, E. Cantú-Paz, D. Goldberg, B. Miller, The gambler's ruin problem, genetic algorithms, and the sizing of populations, Evol. Comput. 7 (3) (1999) 231–253.

[17] G. Harik, F. Lobo, D. Goldberg, The compact genetic algorithm, in: Proceedings of the IEEE Conference on Evolutionary Computation, vol. 3, 1998, pp. 523–528.

[18] W. Hastings, Monte Carlo sampling methods using Markov chains and their applications, Biometrika 57 (1970) 97–109.

[19] D. Heckerman, D. Geiger, M. Chickering, Learning Bayesian networks: the combination of knowledge and statistical data, Mach. Learn. 20 (1995) 197–243.

[20] J. Holland, Adaptation in Natural and Artificial Systems: an Introductory Analysis with Applications to Biology, Control and Artificial Intelligence, University of Michigan Press, Ann Arbor, MI, 1975.

[21] E. Ising, Beitrag zur Theorie des Ferromagnetismus, Z. Phys. 31 (1925) 253–258.

[22] F. Jensen, Bayesian Networks and Decision Graphs, Springer, 2001.

[23] Köln (University of Köln, Germany, 2004), Spin glass ground state server, http://www.informatik.uni-koeln.de/ls_juenger/research/spinglass/, 2004.

[24] S. Li, Markov Random Field Modeling in Image Analysis, 2nd edn., Springer, 2001.

[25] F. Liers, M. Jünger, G. Reinelt, G. Rinaldi, Computing exact ground states of hard Ising spin glass problems by branch-and-cut, in: New Optimization Algorithms in Physics, Wiley, 2004, pp. 47–68.

[26] F. Liers, M. Palassini, A. Hartmann, M. Jüenger, Ground state of the Bethe-lattice spin glass and running time of an exact optimization algorithm, Phys. Rev., B Solid State 68 (9) (2003) 094406.

[27] H. Mühlenbein, The equation for response to selection and its use for prediction, Evol. Comput. 5 (3) (1997) 303–346.

[28] H. Mühlenbein, T. Mahnig, The factorized distribution algorithm for additively decompressed functions, in: Proceedings of the Congress on Evolutionary Computation, 1999, pp. 752–759.

[29] H. Mühlenbein, T. Mahnig, FDA – a scalable evolutionary algorithm for the optimization of additively decomposed functions, Evol. Comput. 7 (4) (1999) 353–376.

[30] H. Mühlenbein, T. Mahnig, Evolutionary computation and beyond, in: Foundations of Real-World Intelligence, CLSI Publications, Stanford, CA, 2001, pp. 123–186.

[31] H. Mühlenbein, G. Paaß, From recombination of genes to the estimation of distributions, I: binary parameters, in: Proceedings of the 4th International Conference on Parallel Problem Solving from Nature, PPSN IV, Springer-Verlag, London, UK, 1996, pp. 178–187.

[32] J. Pearl, Causality, Cambridge University Press, 2000.

[33] M. Pelikan, D. Goldberg, Hierarchical Bayesian optimization algorithm = Bayesian optimization algorithm + niching + local structures, in: Optimization by Building and Using Probabilistic Models, OBUPM 2001, 2001, pp. 217–221.

[34] M. Pelikan, D. Goldberg, E. Cantú-Paz, BOA: the Bayesian optimization algorithm, in: Proceedings of the Genetic and Evolutionary Computation Conference, vol. I, GECCO, Morgan Kaufmann, San Francisco, CA, 1999, pp. 525–532.

[35] M. Pelikan, A. Hartmann, K. Sastry, Hierarchical BOA, cluster exact approximation, and Ising spin glasses, technical report Missouri Estimation of Distribution Algorithms Laboratory (MEDAL) No. 2006002, University of Missouri in St. Louis, St. Louis, MO, 2006.

[36] M. Pelikan, H. Mühlenbein, The bivariate marginal distribution algorithm, in: Advances in Soft Computing – Engineering Design and Manufacturing, Springer-Verlag, London, UK, 1999, pp. 521–535.

[37] W. Press, S. Teukolsky, W. Vetterling, B. Flannery, Numerical Recipes in C, second edition, Cambridge University Press, Cambridge, UK, 1992.

[38] J. Prior, Eugenic evolution for combinatorial optimization, Master's thesis, Department of Computer Sciences, The University of Texas at Austin, Austin, TX, 1998.

[39] H. Rosenbrock, An automatic method for finding the greatest or least value of a function, Comput. J. 3 (1960) 175–184.

[40] R. Santana, A Markov network based factorized distribution algorithm for optimization, in: Machine Learning: ECML 2003, Springer, 2003, pp. 337–348.

[41] R. Santana, Estimation of distribution algorithms with Kikuchi approximations, Evol. Comput. 13 (1) (2005) 66–97.

[42] S. Shakya, DEUM: a framework for an estimation of distribution algorithm based on Markov random fields, Ph.D. thesis, The Robert Gordon University, Aberdeen, UK, 2006.

[43] S. Shakya, J. McCall, D. Brown, Using a Markov network model in a univariate EDA: an empirical cost-benefit analysis, in: Proceedings of the Genetic and Evolutionary Computation Conference, vol. I, GECCO, 2005, pp. 727–734.

[44] S. Shakya, J. McCall, D. Brown, Solving the Ising spin glass problem using a bivariate EDA based on Markov random fields, in: Proceedings of the Congress on Evolutionary Computation, 2006, pp. 908–915.
[45] G. Syswerda, Simulated crossover in genetic algorithms, in: Foundations of Genetic Algorithms, Morgan Kaufmann, San Francisco, CA, 1992, pp. 239–255.
[46] G. Winkler, Image Analysis, Random Fields and Markov Chain Monte Carlo Methods, second edition, Springer, 2003.
[47] F. Yates, Contingency table involving small numbers and the χ^2 test, Suppl. J. R. Stat. Soc. 1 (1934) 217–235.