Noname manuscript No.
(will be inserted by the editor)

DICE: Exploiting *All* Bivariate Dependencies in Binary and Multary Search Spaces

Fergal Lane · **R. Muhammad Atif Azad** · **Conor Ryan**

the date of receipt and acceptance should be inserted later

Abstract Although some of the earliest *Estimation of Distribution Algorithms* (EDAs) utilized bivariate marginal distribution models, up to now, all discrete bivariate EDAs had one serious limitation: they were constrained to exploiting only a limited $O(d)$ subset out of all possible $O(d^2)$ bivariate dependencies.

As a first we present a family of discrete bivariate EDAs that can learn and exploit *all* $O(d^2)$ dependencies between variables, and yet have the same run-time complexity as their more limited counterparts. This family of algorithms, which we label DICE (*DI*screte *C*orrelated *E*stimation of distribution algorithms), is rigorously based on sound statistical principles, and particularly on a modelling technique from statistical physics: *dichotomised multivariate Gaussian distributions*.

Initially [19], DICE was trialled on a suite of combinatorial optimization problems over binary search spaces. Our proposed *dichotomised Gaussian* (DG) model in DICE significantly outperformed existing discrete bivariate EDAs; crucially, the performance gap increasingly widened as dimensionality of the problems increased.

In this comprehensive treatment, we *generalise* DICE by successfully extending it to multary search spaces that also allow for *categorical* variables. Because correlation is not wholly meaningful for categorical variables, interactions between such variables cannot be fully modelled by correlation-based approaches such as in the original formulation of DICE. Therefore, here we extend our original DG model to deal with such situations.

We test DICE on a challenging test suite of combinatorial optimization problems, which are defined mostly on multary search spaces. While the two versions of DICE outperform each other on different problem instances, they both outperform all the state-of-the-art bivariate EDAs on almost all of the problem instances. This further illustrates that these innovative DICE methods constitute a significant step change in the domain of discrete bivariate EDAs.

Keywords Dichotomised Gaussian models · Bivariate Estimation of Distribution Algorithms · Combinatorial Optimization

F. Lane · C. Ryan
CSIS Department, University of Limerick, Ireland
E-mail: {Fergal.Lane, Conor.Ryan}@ul.ie

R. M. Atif Azad
School of Computing and Digital Technology, Birmingham City University, UK
E-mail: atif.azad@bcu.ac.uk

1 Introduction

Estimation of Distribution Algorithms (EDAs), often also called *Probabilistic Model Building Genetic Algorithms* (PMBGAs), are an important optimization paradigm within Evolutionary Computation. They are stochastic optimization methods that guide the search for a global optimum by building and sampling explicit probabilistic models. Traditional search operators like mutation and crossover are instead replaced by a probabilistic model.

Many EDA variants have been developed for both continuous and discrete problem domains. Some of the earliest EDAs used relatively simple univariate models, for example, the *Univariate Marginal Distribution Algorithm* (UMDA) [25], *Population-Based Incremental Learning* (PBIL) [1] and the *Compact Genetic Algorithm* (cGA) [14]. Such uncomplicated models were, obviously, unlikely to suffice to tackle more difficult and intractable problems. There has been a long and natural progression towards the use of more complex models able to capture complicated problem dependencies and structures.

EDAs utilizing models that could capture and exploit bivariate marginal distributions appeared relatively early in

the development of this field. Such bivariate models are potentially much more expressive than univariate models. For discrete spaces, a well known example of a bivariate EDA would be *Mutual Information Maximizing Input Clustering* (MIMIC) [7].

However, successively more expressive models have been investigated. Perhaps the most popular general EDA modelling approach has been graphical models and Bayesian networks. Some discrete search space examples would be the *Bayesian Optimization Algorithm* (BOA) [27] and the *Estimation of Bayesian Networks Algorithm* (EBNA) [9], and, in continuous search spaces, the *Estimation of Gaussian Network Algorithm* (EGNA) [20]. However, increasing expressiveness has almost invariably gone hand in hand with added computational costs and configurational complexities.

In this paper, we explore a novel bivariate EDA approach for discrete search spaces, based on dichotomised multivariate Gaussian distributions, that is no longer hindered by this constraint. These models have the attractive property of capturing and exploiting all $O(d^2)$ bivariate interactions between the d variables of a problem domain. As far as these authors are aware, all bivariate marginal distribution models previously used in discrete space EDAs were restricted to using just $O(d)$ of these bivariate interactions.

The structure of the paper is as follows. Section 2 describes previously developed bivariate EDAs. The *dichotomised Gaussian* (DG) model that DICE utilizes has its origins in the literature on the simulation of correlated multivariate Bernoulli variables. Section 3 begins with a brief survey of this literature. The bulk of this section, however, is devoted to a detailed description of the DG model and its implementation. Section 4 describes an extended version of the basic DG model developed to cope with categorical variables in multary search spaces. Section 5 details our suite of combinatorial optimization problem domains, the bivariate EDA algorithms we compare them with, and other experimental details. Section 6 presents results and analysis for these experiments. Finally, in section 7, we give our conclusions and lay out some ideas for future work.

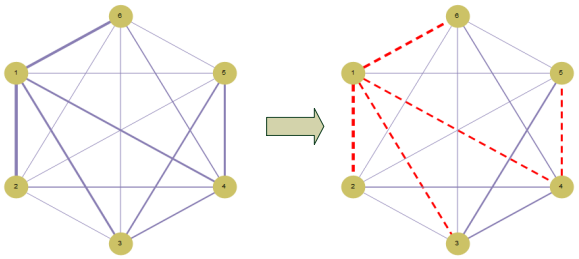
2 Bivariate EDAs

2.1 Continuous Bivariate EDAs

In continuous spaces, models capable of efficiently capturing all bivariate marginal distributions are readily available and easy to use, e.g. the family of multivariate Gaussian distributions. EDAs like EMNA and the *Covariance Matrix Adaptation Evolution Strategy* (CMA-ES)[1] [13] , which utilize such models have, therefore, long been widely available and used. The *Estimation of Gaussian Network Algorithm* (EGNA)[20] is another continuous EDA that, at least in part, uses Gaussian distribution techniques. When there has been a need for even more expressive continuous models, several past authors have utilized *Gaussian Mixture Models* (GMMs) based on weighted mixtures of several multivariate Gaussian distributions [22].

[1] Strictly speaking, CMA-ES does not quite fall into the canonical EDA framework. However, it shares almost all of the core features of a typical EDA.

Fig. 1: The relative thicknesses of edges in the first graph represent relative interaction strengths between six search space variables. Dashed red edges in the second graph represent the resulting constructed MST. Only edges/dependencies in the tree are exploited in these approaches.



2.2 Discrete Bivariate EDAs

For discrete spaces, the only bivariate EDAs available up-to-now have used restricted bivariate interaction models. MIMIC, mentioned earlier, greedily constructs a sequential chain of the most significant $O(d)$ individual bivariate marginal distributions. Building such chains or trees is the principal modelling strategy in existing discrete bivariate EDAs. *Combining Optimizers with Mutual Information Trees* (COMIT) [2], and the *Bivariate Marginal Distribution Algorithm* (BMDA) [28] are other examples. COMIT was essentially an extension of MIMIC. Whereas, for d-dimensional problems with d dependent variables, MIMIC greedily constructed a sequential chain of $O(d)$ individual bivariate marginal distributions, COMIT used a more general $O(d)$ dependency tree structure. MIMIC, COMIT and BMDA are all based on chains or trees of $d-1$ individual bivariate marginal distributions.

COMIT, in effect, builds a Chow-Liu tree [5]. This procedure scores all pairwise interaction densities based on an estimate of their *mutual information* (MI). An efficient *Minimum Spanning Tree* (MST) algorithm is then applied to a matrix of these scores to greedily construct an MST. Fig. 1 illustrates this process. The cost of this MST algorithm for arbitrary dense matrices is $O(d^2)$, which also gives the overall cost for the model building procedure. This tree along with the $d-1$ pairwise bivariate distributions associated with its edges is then traversed to generate new individuals. Chow-Liu trees can be viewed as a particularly simple and constrained form of Bayesian network (where a child can have

at most one parent). However, even if this approach identifies and makes use of the $O(d)$ out of $O(d^2)$ most significant pairwise interactions, it still is the case that the vast majority are discarded. This may increasingly impact on model accuracy as problem dimensionality increases. The BMDA algorithm also uses essentially the same procedure. The principal difference is that it uses Pearson's chi-squared estimator instead of mutual information to score bivariate variable interactions.

In [31], an interesting variation on such tree-based algorithms is given. Their algorithm, *EDA based on Mixtures* (EDAM), simply uses random trees (avoiding this costly $O(d^2)$ step). A mixture of ten such random trees was used as their EDA model in experiments. In essence, this algorithm reduces computational cost at the expense of some model accuracy. Despite randomly constructing the dependency trees, they claim their algorithm, nonetheless, performed similarly to MIMIC on tests.

A small number of authors have previously used multivariate Gaussian copula models in EDAs. An example of this, which has some relevance to our work is [16]. There are some similarities between our approach and the algorithm given in section III of that paper. Their algorithm is potentially capable of learning and exploiting all the bivariate marginal dependencies in the continuous problem domains examined in their paper. However, their method is not practical as a technique for discrete search spaces.

2.3 Computational Cost

A seeming advantage of previous MST techniques is the relatively low $O(d^2)$ cost in building such a model (the main cost is the construction of the MST). However, even if these models identify and use only the $O(d)$ most significant interactions, all $O(d^2)$ bivariate interaction strengths for a population of size n must still be estimated and scored (with computational cost $O(d^2n)$). This is the primary computational bottleneck for MIMIC and other past discrete bivariate EDAs.

The cost complexity of our new *dichotomised Gaussian* (DG) bivariate EDA model, which will be described in detail later, is $O(\max(d,n)\,d^2)$ per generation. However, generally, to have a reasonable chance of accurately estimating all $O(d^2)$ bivariate interaction parameters, it is expected that the population size n should at least be d. Therefore, the computational cost of the DG model normally is still of the same order as this unavoidable $O(d^2n)$ bivariate EDA generational cost; hence, computationally DICE is *not* more complex than past discrete bivariate EDAs despite exhaustively exploiting all the bivariate dependencies. This is a significant result.

3 The Dichotomised Gaussian Model

3.1 Simulation of Correlated Multivariate Bernoulli Variables

The literature concerning simulation (random generation) of correlated binary vectors has a decades long history. A multivariate Bernoulli variable (essentially a randomly generated bit string) is, in principle, fully specified by $2^d - 1$ parameters (in effect, the individual probabilities for every possible bit string of length d it can generate). There is usually little practical hope of accurately learning so many parameters from data. A more realistic goal is to learn the univariate marginal distributions of the variables and the more limited number of $O(d^2)$ correlations between those variables. Then, one constructs a multivariate Bernoulli variable with those same univariate distribution and correlation characteristics.

Usually, the ideal choice would be to use the maximum entropy distribution for which the means and correlations for the set of d binary variables are constrained to the desired target values. In this case, the maximum entropy distribution is actually the well-known Ising model. Unfortunately, it is not at all straightforward or cheap to find the particular Ising model that fits a desired set of mean and correlation constraints. It is also difficult and expensive (indeed NP-complete beyond a certain temperature) to sample binary vectors from such a model. Therefore, many other more practical methods have been proposed for simulating such correlated binary vectors.

Examples would include [4] that proposed a method using look-up tables of size $O(d^3)$, [21] that introduced two methods – one based on setting up a linear programming problem and another based on Archimedean Copulas, [11] that introduced an "iterative proportional fitting algorithm", and [17] that represents a more recent copula approach to this problem.

3.2 The Dichotomised Gaussian Model

The particular technique, the *dichotomised Gaussian* (DG) model, that we have elected to use has been described and utilized in a number of fields (though never previously in optimization or with EAs), with the first description being possibly [8]. The approach uses *dichotomised multivariate Gaussian distributions* where the real values in the vector produced by an appropriate multivariate Gaussian are *dichotomised* (converted into discrete integral values) via a set of appropriately constructed thresholds. A more recent exposition of this method of dichotomising via thresholding can be found in [24] (applying it to the biological problem of simulating neuronal spike trains). Those authors also argue that this model is "near maximum-entropy". This method

can also easily be extended to the more general case of generating multary vectors with any given correlation structure and associated set of univariate marginal distributions.

3.2.1 Randomly Generating Multary Strings

The DG method randomly generates multary search space strings $\omega \in \Omega$ where each gene position can choose from two or more possible allele values. Here Ω is a d-dimensional multary search space $\Omega = \prod_{i=1}^{d} \mathbb{Z}_{a_i}$ where $\mathbb{Z}_{a_i} = \{0,\ldots,a_i-1\}$, so that each $a_i \geq 2$ specifies the *arity* of the i$^{\text{th}}$ dependent variable (gene) ω_i. So, using this notation, the i$^{\text{th}}$ gene ω_i (out of d possible genes) can take on one out of of a_i possible allele values (ranging from 0 up to a_i-1). At the heart of the DG model is a d-dimensional multivariate normal distribution $\mathcal{N}(0,\Sigma)$ and an associated set of thresholds:

$$\mathcal{T} = \left\{ t_k^b \mid k \in \{1,\ldots,d\}, b \in \{0,\ldots,(a_k-2)\} \right\}$$

The multivariate normal distribution is used to randomly generate a d-dimensional continuous vector $x \in \Re^d$. Each individual variable x_i will have a univariate marginal distribution in $\Re$ with the usual Gaussian bell-shaped distribution (with a mean of 0 and a variance of 1). The threshold values in $\mathcal{T}$ are used to convert this d-dimensional randomly generated continuous vector x into a d-dimensional discrete multary string ω. Each variable ω_i, with arity a_i , will have a_i-1 increasing threshold values associated with it: $t_i^0 < t_i^1 \cdots < t_i^{a_i-2}$which partition $\Re$ into a_i disjoint intervals:

$$K_i^0 = \left(-\infty, t_i^0\right], K_i^1 = \left(t_i^0, t_i^1\right], \ldots, K_i^{a_i-1} = \left(t_i^{a_i-2}, \infty\right)$$

So, for example, if x_i happens to fall in the particular interval K_i^c, then we set $\omega_i = c$.

The basic operation of this method is, therefore, straightforward: we first generate multivariate normal vectors according to $\mathcal{N}(0,\Sigma)$ and then dichotomise these using the set of thresholds $\mathcal{T}$ to produce random multary strings.

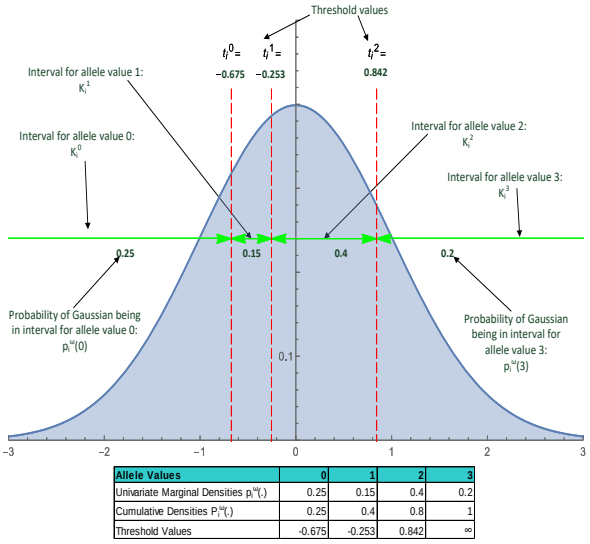
3.2.2 Configuring the DG Model

The goal of the DG method is to allow the random generation of multary search space vectors $\omega \in \Omega$ so that these conform to a set of desired univariate marginal density functions $\{p_i^\omega(c)\}_i$, where $p_i^\omega(c) = \Pr\{\omega_i = c\}$, and according to a target set of variable (gene) correlations r_{ij} given in a $d \times d$ gene correlation matrix $(R)_{i,j} = r_{ij}$. Usually, the $\{p_i^\omega(c)\}_i$ are empirical univariate marginal densities estimated from normalized allele frequency counts in the population, and R is a sample correlation matrix estimated from a population of selected search space points. Once we have our target estimates for R and $\{p_i^\omega(c)\}_i$, then three further steps need to be followed to properly configure the DG model (see Algorithm 1):

Algorithm 1 Steps to configure the DG model

1: Calculate the set of threshold values $\mathcal{T}$. These are chosen to replicate the target univariate marginal densities $p_i^\omega(c)$. Calculating $\mathcal{T}$ has computational complexity $O(a)$ where $a = \Sigma_{i=1}^{d} a_i$ and empirically estimating the $\{p_i^\omega(c)\}_i$ from the population has cost $O(dn)$.
2: Calculate the correlation matrix Σ. The multivariate normal $d \times d$ correlation matrix $(\Sigma)_{i,j} = \rho_{ij}$ is constructed to ensure that the correlations $\text{corr}(\omega_i, \omega_j)$ in the generated multary strings ω match the target gene correlation matrix R. Calculating Σ has cost $O(a^2)$ and empirically estimating R has cost $O(d^2 n)$.
3: If necessary, use a nearest correlation matrix algorithm to repair Σ (to reconcile any conflicting gene correlation targets). This has cost $O(d^3)$.

Fig. 2: Illustration of calculating thresholds for a gene ω_i with four possible allele values and specific occurrence probabilities for each of these alleles.



Allele Values	0	1	2	3
Univariate Marginal Densities $p_i^\omega(.)$	0.25	0.15	0.4	0.2
Cumulative Densities $P_i^\omega(.)$	0.25	0.4	0.8	1
Threshold Values	-0.675	-0.253	0.842	∞

3.2.3 Calculating the Threshold Values

We set the thresholds to exactly replicate the target univariate marginal densities $\{p_i^\omega(c)\}_i$. Let $P_i^\omega(c) = \Pr\{\omega_i \leq c\}$ be the target *cumulative distribution function* (CDF) for variable ω_i, which we can calculate as $P_i^\omega(c) = \sum_{k=0}^{c} p_i^\omega(c)$. If we were thresholding simple uniform $\text{U}(0,1)$ distribution variables (rather than normal variates), then these CDF density values could be directly used as the thresholds. However, since we are dichotomising normal distribution variables, then we must first use the standard inverse normal CDF function $\Phi^{-1}(x)$ to map these to suitable corresponding normal threshold values:

$$t_i^0 = \Phi^{-1}(P_i^\omega(0)), \ldots, t_i^k = \Phi^{-1}(P_i^\omega(k)), \ldots$$

It can be easily verified that the univariate marginal densities of the resulting randomly generated multary vectors ω will indeed equal $\{p_i^\omega(c)\}_i$. See Fig. 2 for a concrete example.

3.2.4 Replicating Gene Correlations

In step 1, we have already replicated the univariate marginal densities of our target discrete distribution. If we went no further and simply used a vector of d independent normal variates with diagonal Σ, in our multivariate normal distribution $\mathscr{N}(0,\Sigma)$, then, in effect, we would have constructed a univariate EDA like PBIL or UMDA. However, $(\Sigma)_{ij} = \rho_{ij}$ has $O(d^2)$ free parameters for us to tune. After empirically estimating the matrix $(R)_{i,j} = r_{ij}$ of correlations between the search space variables in the selected population, we then adjust each ρ_{ij} so that the resulting correlation $\mathrm{corr}(\omega_i,\omega_j)$ between variables i and j in the generated random multary vector ω equals the desired target value r_{ij}.

We can deal with each pair of variables ω_i and ω_j separately in turn. Their thresholds in $\mathscr{T}$ have already been calculated. Unfortunately, we cannot simply set ρ_{ij} equal to r_{ij}. However, for any possible value of ρ_{ij}, we can efficiently calculate what the resulting $\mathrm{corr}(\omega_i,\omega_j)$ would be.

Only cheap-to-compute functions from the standard mathematical toolbox are needed. We use the standard bivariate normal CDF function $\Psi_2(x,y;\rho)$ to calculate bivariate marginal CDFs for the generated vector ω as:

$$P_{ij}^{\omega}(b,c) = \mathrm{Prob}\left\{\omega_i \le b \cap \omega_j \le c\right\} = \Psi_2\left(t_i^b, t_j^c; \rho_{ij}\right), i \neq j$$

For convenience, if we also define $P_{ij}^{\omega}(b,c)$ to be 0 whenever $b < 0$ or $c < 0$, then the bivariate marginal densities $\mathrm{p}_{\mathrm{ij}}^{\omega}(\mathrm{b},\mathrm{c}) = \mathrm{Prob}\left\{\omega_i = b \cap \omega_j = c\right\}$ for ω can be expressed and easily calculated as:

$$\begin{aligned}\mathrm{p}_{\mathrm{ij}}^{\omega}(\mathrm{b},\mathrm{c}) &= P_{ij}^{\omega}(b,c) - P_{ij}^{\omega}(b-1,c) - \\ &\quad P_{ij}^{\omega}(b,c-1) + P_{ij}^{\omega}(b-1,c-1)\end{aligned}$$

From these density values, the gene correlations can be calculated as:

$$\mathrm{corr}(\omega_i,\omega_j) = \sum_{b=0}^{a_i-1}\sum_{c=0}^{a_j-1} b\,c\,p_{ij}^{\omega}(b,c) - \mathscr{E}(\omega_i)\,\mathscr{E}(\omega_j),$$

$$i \neq j, \quad \text{where} \quad \mathscr{E}(\omega_i) = \left(\sum_{b=0}^{a_i-1} b\,p_i^{\omega}(b)\right)$$

We numerically adjust ρ_{ij} until the resulting calculated gene correlation between ω_i and ω_j exactly equals r_{ij}. As pointed out in [24], this problem is monotonic and there is guaranteed to be a single unique solution for ρ_{ij} lying within $[-1,1]$. Straightforward and efficient one-dimensional bisection root-finding algorithms can be used to quickly solve for each ρ_{ij}. In our implementation, we used Brent's root-finding bisection method, which on average converged within only six iterations. Alternative descriptions of this approach can be found in [8] and [24]. This process is repeated to estimate ρ_{ij} for each pair of variables until eventually Σ is fully calculated.

3.2.5 Repairing the Correlation Matrix

If the resulting Σ is a valid correlation matrix, i.e. is positive semi-definite, then configuration of the DG model is complete. Positive semi-definite means all eigenvalues are non-negative $\lambda_i \ge 0$ in the eigenvector decomposition: $\Sigma = QDQ^T$ where $D = \mathrm{diag}(\lambda_i)$. Σ may not always be positive semi-definite due to incompatibilities, under this method, between different gene correlation targets in R. However, efficient *correlation matrix repair* algorithms exist that can then replace Σ with its nearest valid correlation matrix, calculated in terms of the Frobenius matrix norm.

A paper by Nicholas Higham [15] introduced the first algorithm for finding, for any arbitrary real matrix, its nearest valid correlation matrix. This method was based on Djikstra's "alternating projections method" and had linear convergence. However, later Newton-method based algorithms have been developed with fast quadratic convergence [29]. We used a publicly available[2] C-code version of this Newton-based algorithm. Our implementation of DICE is available from: `https://github.com/FergalLane/DICE`.

A simpler method, though not examined here, which we have found empirically almost as effective, is *covariance matrix repair*. The nearest valid covariance matrix to Σ is given by $\Sigma^+ = Q\,\mathrm{diag}(\max(\lambda_i,0))\,Q^T$. Σ^+ can then easily be converted back into a correlation matrix by renormalization: $\Sigma_{ij} = \frac{\Sigma_{ij}^+}{\sqrt{\Sigma_{ii}^+\Sigma_{jj}^+}}$.

Both of these repair approaches have cost $O(d^3)$ (due to the matrix operations involved). As can be seen from Algorithm 1, configuring the DG model each generation has an overall computational cost of $O(\max(a^2,d^2n,d^3))$. Using the DG model, generating each new individual point has cost $O(d^2)$ and new population, $O(d^2n)$. Unless allele arities are large, the effective cost complexity of the DG approach per generation is $O(\max(d,n)\,d^2)$.

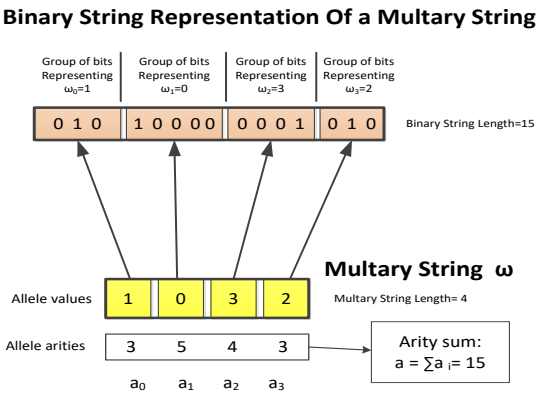
4 Extending the DG Model

4.1 Categorical Variables

DICE uses a correlation-based modelling approach. However, variable correlations may not always be an entirely relevant measure for some types of multary search space. For example, in the graph colouring problem, which we will encounter later, while numeric values can indeed be assigned to colours in a representation, a colour *average* in this case is not really meaningful. Colour is not a quantitative value for this problem; there is no inherent ordering between the possible colours. A variable holding such a qualitative value

[2] Downloadable from: `http://www.math.nus.edu.sg/~matsundf/`

Fig. 3: An illustration of how a multary string is mapped to a binary representation



is usually termed a *categorical* or *nominal* variable. Correlation between the assigned colour values can still have a degree of usefulness. However, the inadequate treatment of categorical multary variables is a potential weakness in our original DICE approach. Therefore, in this section, we extend the original DG model to deal with this categorical case.

4.2 Binary Representations for Multary Strings

This extended model uses a binary representation for multary strings. Suppose we have a multary string $\omega = \{\omega_0, \omega_1, \ldots, \omega_{d-1}\}$ with individual variables ω_i having arities a_i. In our new approach, this multary string will be represented as a longer bit string b whose total length is $a = \sum_{k=1}^{d} a_k$ (the *arity sum*). Each multary variable ω_k is represented by a contiguous group of a_k bits in b. Within this group, one and only one bit is allowed to be set. The position of this set bit corresponds to the allele value in ω_k. The first bit being set corresponds to an allele value of 0, the second being set corresponds to a value of 1 etc. Fig. 3 gives a concrete example of this mapping process.

In this representation, every possible allele value is allocated an individual bit position and is treated separately; correlations are calculated between individual allele values (collective averages and correlations between entire variables are no longer calculated).

4.3 Generating Multary Strings With Categorical Variables

For a population of multary search space points expressed using this a-dimensional bit string representation, a DG model can be built in the standard previously-described way (requiring the construction of an $a \times a$ correlation matrix and a threshold values for $\mathscr{T}$). This can then be used to generate random bit strings with statistical characteristics matching the population. For example, the probability of any particular bit k, which represents allele value c at variable ω_i in the population, being set will equal the univariate marginal density $p_i^{\omega}(c)$ for that allele value.

However, a complication arises if we want to use this standard DICE approach to then randomly generate new individuals. Within any group of a_i bits representing a multary variable ω_i, on average one of those bits will be set. However, sometimes more than one bit in the group will be set; at other times none at all will be set. To randomly simulate multary strings, we need to be able to generate random binary vectors with the above statistical characteristics, but also constrained in such a way that always precisely one bit is set within each variable group (not just on average).

4.4 A Related Statistical Sampling Problem

Luckily, related problems have been studied in the field of stratified sampling in statistics [30]. Borrowing a technique from this area, we can indeed generate such bit strings with reasonable accuracy. This involves dichotomising the multivariate normal vector x generated by DICE in a new way.

The statistical topic, which closely relates to our needs, is the question of how best to randomly sample, without replacement, n items out of a population of size N, so that the selection probability for each item k is proportional to some associated non-negative weight W_k (we can also assume that these weights have been normalized so that $\sum_k W_k = n$). Several techniques developed for doing this involve associating with each item k an independent random number $U_k \sim U(0,1)$ and then using the W_k as thresholds. If we select only items for which $W_k \geq U_k$, i.e. for which $\zeta_k = \frac{W_k}{U_k} \geq 1$, then the resulting sample will have, on average, n items exactly sampled proportional to their weights W_k. Unfortunately, the resulting sample will only have a size of n in expectation.

Several authors have proposed techniques to remedy this issue. Ohlsson proposed *Sequential Poisson Sampling* (SPS) [26], which simply selects the n items with the largest ζ_k values. He showed that this technique produces sampling probabilities close to the desired ones.

This example almost exactly mirrors our requirement to set/select exactly one ($n = 1$) bit out of a group of $N = a_i$ bits that represent a multary variable ω_i. In DICE, each bit k is set if its associated normal variable x_k exceeds the corresponding threshold value t^k in $\mathscr{T}$. On average, exactly one bit per group is set. The principal difference with the statistical example is that we are using normal rather than uniform random variates. However, using the standard normal CDF function $\Phi(x)$, we can equivalently reformulate this in terms of the uniform distribution, putting $U_k = \Phi(x_k)$ and $W_k = \Phi(t^k)$.

4.5 A New Thresholding Procedure

To always exactly set one bit in a group, we can use the SPS technique for thresholding: we first calculate $\zeta_k = \frac{W_k}{U_k}$ for each bit in the variable group and only set the bit with the largest ζ_k value. This alternative thresholding technique will always produce a binary vector in a format that we can directly convert back into a multary string.

Rosen later created a similar sampling technique with even better properties: Pareto sampling [30]. In our work, we used his slightly different formula: $\zeta_k = \frac{W_k(1-U_k)}{U_k(1-W_k)}$, setting only the single bit in each group with the largest such value.

This new thresholding technique is easy to implement and is likely to produce univariate and bivariate marginal density properties close to the desired ones. The resulting overall DG model is also bigger and more detailed (learning an $a \times a$ rather than a $d \times d$ correlation matrix from the data). Exploiting more problem information could lead to improved performances. However, there are added costs (the cost per generation is now $O(\max(a^3, d^2 n))$ and, if there is not sufficient data to learn the bigger model, performance could potentially be negatively impacted.

5 Experimental Setup

Our overall goal was to test and compare the performance of DICE (using both original and extended dichotomised Gaussian models) with other existing discrete bivariate EDAs. Previously in [19], we only tested DICE on problem domains with binary search spaces. This time our test suite contains mostly problem domains with multary search spaces. To ensure an absolutely fair model comparison, we used the same basic EDA algorithm with identical settings with each EDA model.

5.1 EDA Algorithm Settings

All the EDAs used a population of 200 individuals. All algorithms were run for 100 generations. At each iteration, 200 new individuals were generated. We did this by going through all 200 existing individuals and, with probability p_m, either mutating it (by picking one variable at random and modifying its allele value) or, otherwise, replacing it entirely with an individual randomly generated using the current EDA model. Such a strategy was found to improve the performance of all EDAs we were testing. We used a setting of $p_m = 0.5$ for all EDAs (empirically this particular setting was determined to most boost general EDA performances).

The 100 fittest individuals in the new population were then selected and used in updating the probability model. All the probabilistic EDA models were constructed, at each iteration t, using a set H_t^* of univariate and/or bivariate marginal densities (estimated from current and previous selected populations). We used an exponentially-decaying weighted average of previously encountered population density histograms. A model decay parameter $\tau \in [0,1]$ was used to determine the factor at which the previous model was discounted at each iteration. So the new EDA model H_t^*would be the weighted combination: $H_t^* = (1-\tau)H_t + \tau H_{t-1}^*$ of the just previously discounted model $H_{t-1}^*$and the most recent selected population histogram H_t. Extensive empirical testing determined that $\tau = 0.75$ was the best general setting for the EDAs on the problem test suite we examined here. An identical $\tau = 0.75$ setting was used for all runs. Batches of 100 runs were used to produce all the experimental results given below (except for MAX-SAT and graph colouring where 200 runs were used). We tested the EDAs on a range of dimensionalities: $d = 10, 20, \ldots, 100$ for all problem domains.

We compared our original DICE EDA, and the new extended version, *C*ategorical *V*ariable-DICE (CV-DICE), against a simple *Univariate EDA* (UEDA) model and the three principal discrete bivariate EDA models available in the literature. These were the MST-based BMDA model (using the Pearson chi-squared statistic), the MST-based MIMIC model (scoring interactions using mutual information) and an inexpensive random tree (EDAM) model where a mixture of ten randomly chosen dependency tree structures was used (the same model used in [31]).

5.2 Problem Test Suite

Several well-known combinatorial optimization problem domains, all defined on either binary or multary search spaces, were used; these are described in more detail in Table 1. All of these problem domains generated new fitness functions for every run by randomly sampling a new set of weights. We deliberately chose such problem domains because they readily scale to higher dimensions, and we wanted to test the performance of these models on search spaces of varying dimensionalities.

Some, but not all, of the problem domains previously used in [19] had straightforward natural extensions to multary search spaces. Multary search spaces in this study used an arity of 4 for all variables (except for graph colouring, which used 3 colours and an arity-3 representation). Multary versions of QUBO and CUBO (Cubic Unconstrained Binary Optimization) were used that simply replaced binary variables with integer ones (and the formulas rescaled appropriately).

Phase transition behaviour is well understood for both graph colouring [18] and 3-Uniform SAT. To try to ensure problem difficulty, we used an unweighted version of MAX-SAT with the number of clauses chosen using the formula

from [6]: $c = 4.258\,d + 58.26\,d^{\frac{1}{3}}$ to place it at the phase transition point for SAT (similarly for 3-graph colouring).

In [23], some non-binary search space versions of NK Landscapes were developed. One of these, *Nominal NK Landscapes*, was a simple multary extension of the original where the fitness component functions F_i were also allowed to accept multary allele values. This NK Landscape variant has a categorical variable structure. We wished to also include a multary version of NK Landscapes that had variables with non-categorical (ordinal or quantitative) properties. Therefore, we constructed a novel but simple multary NK Landscape based on tessellations (or random cuts). Associated with each variable i_k in every F_i function (see Table 1) is a randomly generated cut value $c_i \sim \mathrm{U}(0, a_i - 1)$. Allele values below this cut are treated by the F_i functions as 0, while allele values greater than this cut point are treated as 1 (ensuring that allele values further apart are more likely to be "cut" and assigned different random values).

We also included the Quadratic Knapsack Problem, which has a significant associated literature. We use the classic form of the problem as originated by Gallo [10].

6 Results and Analysis

Fig. 4 details the EDA performances on both types of NK Landscapes. Like other graphs in this section, mean best run fitness for the EDA models for individual problem domains is plotted. Clearly, DICE (using the original DG model) and CV-DICE (using the new extended model) are the best performers in all cases. CV-DICE has better performance in more cases. However, for higher dimensions on the Nominal NK Landscapes with K=3,4, DICE has better performances. The reasons for this will need further investigation; it is possible our populations contain an insufficient number of individuals for the more detailed CV-DICE model to be fully learned, particularly at higher dimensions.

Fig. 5 gives EDA performances on the Multary QUBO and CUBO problem domains. DICE and CV-DICE have the overall best performances on QUBO. DICE spectacularly outperforms all other EDAs on multary QUBO. CV-DICE is still clearly the runner-up, but lags far behind DICE. This is perhaps unsurprising given that the variable structure is very much quantitative, rather than categorical, for this problem domain.The performance gaps between DICE and CV-DICE and back to the others are all significant at least a 99% level of significance using the two-tailed Student's t-test (for all further comparisons a 99% level of significance using this test can be assumed unless specified otherwise).

For nine out of the eleven problem domains in this test suite, DICE and CV-DICE have the best performances. Multary CUBO is one of two cases where this did not hold. In [19], DICE could successfully cope with moderate numbers of higher order interactions. Here, perhaps the much denser $O(d^3)$ number of order-three variable interactions simply overwhelms the $O(d^2)$-parameter DG model.

For the two problem domains, 3-Uniform MAX-SAT and 3-Graph Colouring, that made use of phase transition theory, the performance margins between all EDAs were numerically very small and graphically hard to see. The optimizers quickly were able to satisfy almost all colour or clause constraints. However, at the phase transition point, while there may be many local optima coming relatively close to satisfying all clauses/colour constraints, there will be very few or no points satisfying all of them. The real difficulty with such problem domains is in finding solutions which satisfy the final small number of remaining constraints.

For these two problem domains, we ran batches of 200 rather than 100 simulations in order to better statistically differentiate between such small performance gaps. To further simplify comparison of results and reduce standard errors, we give combined mean best run fitnesses, averaged across all dimensionalities, for these problem domains. These combined results are given in table form in Table 2 rather than as graphs. Similar performance figures for the Quadratic Knapsack problem domain are also given (performance gaps were also small in this case).

CV-DICE performs best on the Quadratic Knapsack and DICE is in second place. DICE and CV-DICE perform best on MAX-SAT with their performance gap not being statistically significant; there should not really be a significant performance gap as both DG models should perform almost identically on binary search spaces. The performance gap back to BMDA and MIMIC is also not quite statistically significant.

For 3-Graph Colouring, MIMIC and BMDA beat CV-DICE and DICE into third and fourth places. However, at the 3-Graph Colouring phase transition, each vertex is connected on average with only 4.687 other vertices. For such sparse limited-dependency graphs, simpler tree-based models may be adequate (the added expressivness of the DG model may confer no real advantage in this case).

Overall, for nine out of eleven problem domains, both DICE and CV-DICE have superior performances to all other EDAs.

7 Conclusions

In this paper, we generalise the first fully bivariate model for discrete EDAs to cover both binary and multary search spaces. A key innovation was a treatment of categorical variables which are not directly amenable to correlational models. A key feature of the DICE family is its computational complexity: in its original form, despite exploiting all $O(d^2)$ bivariate dependencies, DICE is no more computationally

Table 1: Problem Domain Test Suite Details

Problem Domain	**Category**	**Ref**	**Fitness Function Formula/Details**
Multary QUBO	Multary	[3]	$f(\omega)=\sum_{i,j=1}^{d} w_{ij}(2s_i-1)(2s_j-1)$, $w_{ij}\sim\mathcal{N}(0,\frac{1}{d^2})$, $s_i=\frac{\omega_i}{a_i-1}$
Multary CUBO	Multary	[12]	$f(\omega)=\sum_{i,j,k=1}^{d} w_{ijk}(s_i)(s_j)(s_k)$, $w_{ij}\sim\mathcal{N}(0,\frac{1}{d^3})$, $s_i=\frac{\omega_i}{a_i-1}$
Nominal NK-Landscapes	Categorical Multary	[23]	$K=2,3,4$; $f(\omega)=\frac{1}{\sqrt{d}}\sum_{i=1}^{d} F_i(\omega_i;\omega_{i_1},\ldots,\omega_{i_K})$ where the i_j are chosen randomly without replacement; each F_i returns a unique random normally distributed value for every unique $K+1$-allele combination encountered.
Tessellation NK-Landscapes	Multary		Same as above except a random *cut value* $c_{i_k}\sim \mathrm{U}(0,a_{i_k}-1)$ is associated with each variable in every F_i and $f(\omega)=\frac{1}{\sqrt{d}}\sum_{i=1}^{d} F_i(\mathrm{I}_{\omega_i\geq c_i};\mathrm{I}_{\omega_{i_1}\geq c_{i_k}},\ldots,\mathrm{I}_{\omega_{i_K}\geq c_{i_K}})$ so allele values on the same side of a cut are treated identically, and those on opposites sides, differently.
3-Uniform MAX-SAT	Binary	[6]	The number of clauses c is set using the phase transition formula in [6].
3-Graph Colouring	Categorical Multary	[18]	Fitness is the number of edges in the graph whose vertex colours differ; the graph connectivity is set as close as possible to 4.687 to be at the phase transition.
Quadratic Knapsack Problem	Binary	[10]	$f(\omega)=\frac{1}{d}\sum_{i,j=1}^{d} p_{ij}\omega_i\omega_j$ subject to $\sum_{j=1}^{d} w_j\omega_j\leq c$, each p_{ij} is non-zero with probability $\Lambda=0.33$, $p_{ij}\sim \mathrm{U}(1,50)$, $w_i\sim \mathrm{U}(1,100)$ and $c\sim \mathrm{U}[50,\sum_{j=1}^{d} w_i]$, $\omega_i\in\{0,1\}$

Fig. 4: Mean Best Run Fitnesses for the EDA models on the Nominal NK and Tessellation NK Landscapes

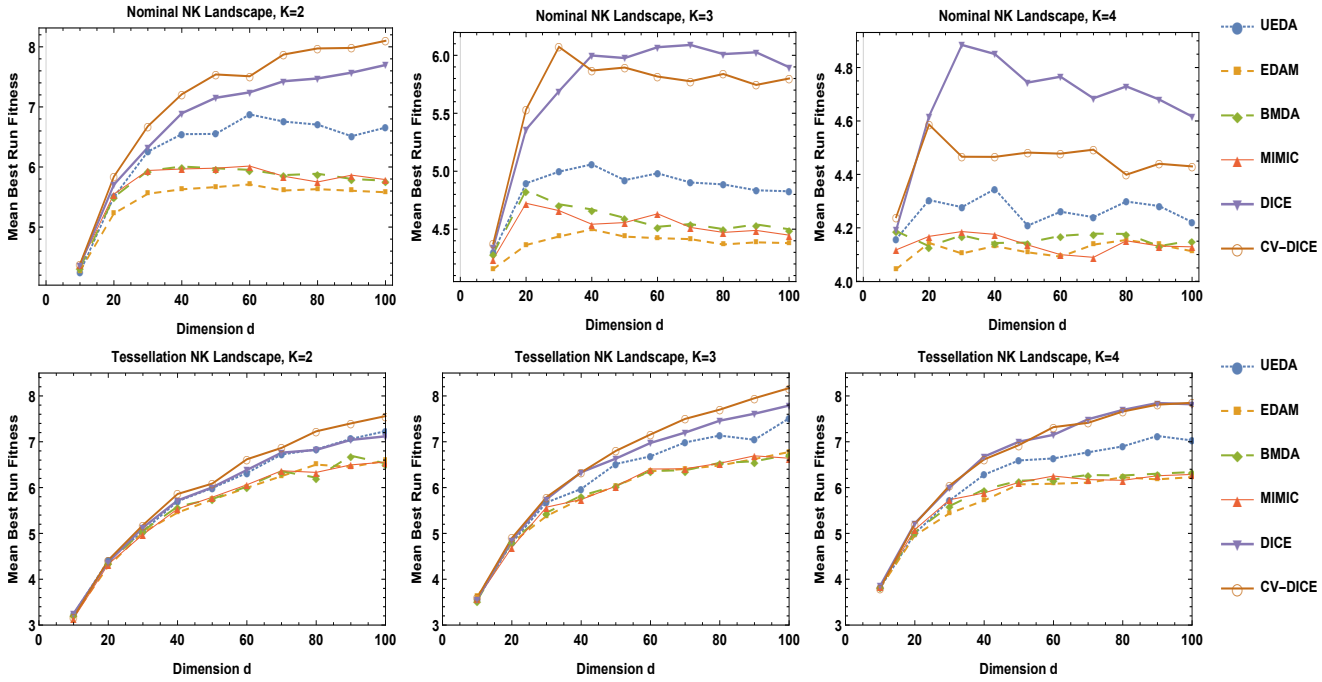


Fig. 5: Mean Best Run Fitnesses for the EDA models on the Multary QUBO and CUBO problem domains

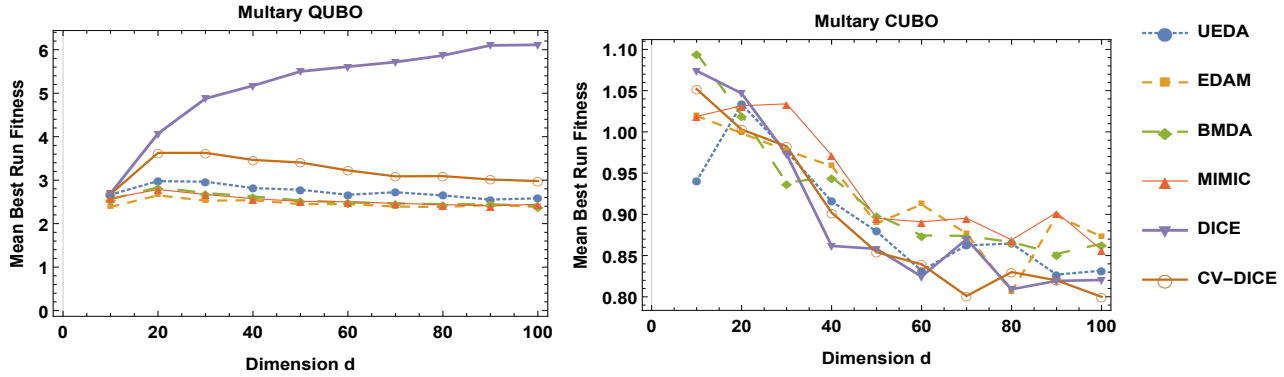


Table 2: Combined Mean Best Run Fitnesses for the EDA models averaged across all Problem Sizes for the 3-Graph Colouring, 3-Uniform Knapsack and Quadratic Knapsack problems domains

	Problem Domains		
EDA Model	3-Uniform MAX-SAT	3-Graph Colouring	Quadratic Knapsack
UEDA	236.04 ± 0.03	53.09 ± 0.02	230.10 ± 0.03
EDAM	235.77 ± 0.03	52.73 ± 0.02	227.85 ± 0.03
BMDA	236.18 ± 0.03	53.61 ± 0.02	228.28 ± 0.03
MIMIC	236.18 ± 0.03	53.69 ± 0.02	228.27 ± 0.03
DICE	236.24 ± 0.03	53.34 ± 0.02	230.38 ± 0.03
CV-DICE	236.20 ± 0.03	53.50 ± 0.02	230.81 ± 0.03

complex than its approximate counterparts that only explore $O(d)$ dependencies. In a majority of cases in our experiments, the extended DICE outperformed the original DICE model. Overall, however, the two DICE versions outperformed all the other state-of-the-art discrete bivariate EDAs on nine out of eleven problems in the test suite.

7.1 Future Work

The DG model has the potential to adapt other algorithms from continuous domains, particularly those based on Gaussian distributions, to discrete spaces. CMA-ES [13], a popular and powerful EDA-like optimizer for continuous search spaces, would be a promising candidate for such treatment.

The DG model would be a good candidate for knowledge incorporation. Correlation matrix repair can also be performed using various forms of *weighted* Frobenius norms (rather than the standard norm we use in this paper). Increased weights can be used to give increased priority and protection to variable interactions learned to be more significant.

Acknowledgements This work was supported with the financial support of the Science Foundation Ireland grant 13/RC/2094.

References

1. Baluja, S., Caruana, R.: Removing the genetics from the standard genetic algorithm. In: 12th Int. Conf. on Machine Learning. pp. 38–46 (1995)
2. Baluja, S., Davies, S.: Using optimal dependency-trees for combinational optimization. In: 14th Int. Conf. on Machine Learning. pp. 30–38 (1997)
3. Boros, E., Hammer, P., Tavares, G.: Local Search Heuristics for Quadratic Unconstrained Binary Optimization (QUBO). Journal of Heuristics 13(2), 99–132 (Apr 2007)
4. Caprara, A., et al.: Generation of antipodal random vectors with prescribed non-stationary 2-nd order statistics. IEEE Transactions on Signal Processing 62(6), 1603–1612 (2014)
5. Chow, C., Liu, C.: Approximating discrete probability distributions with dependence trees. IEEE Transactions on Information Theory 14(3), 462–467 (1968)
6. Crawford, J., Auton, L.: Experimental results on the crossover point in random 3-SAT. Artificial intelligence 81(1), 31–57 (1996)
7. De Bonet, J., Isbell, C., et al.: MIMIC: Finding optima by estimating probability densities. Advances in Neural Information Processing Systems pp. 424–430 (1997)
8. Emrich, L., Piedmonte, M.: A method for generating high-dimensional multivariate binary variates. The American Statistician 45(4), 302–304 (1991)
9. Etxeberria, R., Larranaga, P.: Global optimization using Bayesian networks. In: Second Symposium on Artificial Intelligence (CIMAF-99). pp. 332–339. Habana, Cuba (1999)
10. Gallo, G., et al.: Quadratic knapsack problems. In: Combinatorial Optimization, pp. 132–149. Springer (1980)
11. Gange, S.: Generating multivariate categorical variates using the iterative proportional fitting algorithm. The American Statistician 49(2), 134–138 (1995)
12. Glover, F., Hao, J.K., Kochenberger, G.: Polynomial unconstrained binary optimisation – part 2. International Journal of Metaheuristics 1(4), 317–354 (2011)
13. Hansen, N., Kern, S.: Evaluating the CMA evolution strategy on multimodal test functions. In: PPSN VIII. pp. 282–291 (2004)
14. Harik, G., Lobo, F., et al.: The Compact Genetic Algorithm. IEEE Transactions On Evolutionary Computation 3(4), 287–297 (1999)
15. Higham, N.: Computing the nearest correlation matrix : a problem from finance. IMA Journal of Numerical Analysis 22(3), 329–343 (2002)
16. Hyrš, M., Schwarz, J.: Multivariate Gaussian copula in estimation of distribution algorithm with model migration. In: Foundations of Computational Intelligence (FOCI). pp. 114–119. IEEE (2014)
17. Jin, R., Wang, S., et al.: Generating spatial correlated binary data through a copulas method. Science Research 3(4), 206–212 (2015)
18. Krząkała, F.: How many colors to color a random graph? Cavity, complexity, stability and all that. Progress of Theoretical Physics Supplement 157, 357–360 (2005)
19. Lane, F., Azad, R., Ryan, C.: DICE: A new family of bivariate estimation of distribution algorithms based on dichotomised multivariate Gaussian distributions. In: European Conference on the Applications of Evolutionary Computation. pp. 670–685. Springer (2017)
20. Larrañaga, P., Etxeberria, R., et al.: Combinatorial optimization by learning and simulation of Bayesian networks. In: 16th Conf. on Uncertainty in Artificial Intelligence. pp. 343–352 (2000)
21. Lee, A.: Generating random binary deviates having fixed marginal distributions and specified degrees of association. The American Statistician 47(3), 209–215 (1993)
22. Li, B., Wang, X., Zhong, R., Zhuang, Z.: Continuous optimization based-on boosting Gaussian mixture model. In: 18th Int. Conf. on Pattern Recognition. vol. 1, pp. 1192–1195. IEEE (2006)
23. Li, R., Emmerich, M., et al.: Mixed-integer NK landscapes. In: Parallel Problem Solving from Nature (PPSN IX). pp. 42–51 (2006)
24. Macke, J., Berens, P., et al.: Generating spike trains with specified correlation coefficients. Neural Computation 21(2), 397–423 (2009)
25. Mühlenbein, H.: The equation for response to selection and its use for prediction. Evolutionary Computation 5(3), 303–346 (1997)
26. Ohlsson, E.: Sequential Poisson Sampling. Journal of Official Statistics 14(2), 149 (1998)
27. Pelikan, M., Goldberg, D., Cantú-Paz, E.: BOA: The Bayesian optimization algorithm. In: GECCO 1999. pp. 525–532 (1999)
28. Pelikan, M., Mühlenbein, H.: The bivariate marginal distribution algorithm. In: Advances in Soft Computing, pp. 521–535 (1999)
29. Qi, H., Sun, D.: A quadratically convergent Newton method for computing the nearest correlation matrix. SIAM journal on matrix analysis and applications 28(2), 360–385 (2006)
30. Rosén, B.: On sampling with probability proportional to size. Journal of Statistical Planning and Inference 62(2), 159–191 (1997)

31. Zhang, Q., Sun, J., Tsang, E., Ford, J.: Estimation of Distribution Algorithm based on Mixture. Tech. rep.