DICE: A New Family of Bivariate Estimation of Distribution Algorithms based on Dichotomised Multivariate Gaussian Distributions

Fergal Lane[1], R. Muhammad Atif Azad[2] and Conor Ryan[1]

[1] CSIS Department, University of Limerick, Ireland
{Fergal.Lane,Conor.Ryan}@ul.ie
[2] School of Computing and Digital Technology,
Birmingham City University, UK
atif.azad@bcu.ac.uk

Abstract. A new family of *Estimation of Distribution Algorithms* (EDAs) for discrete search spaces is presented. The proposed algorithms, which we label DICE (*D*iscrete *C*orrelated *E*stimation of distribution algorithms) are based, like previous bivariate EDAs such as MIMIC and BMDA, on bivariate marginal distribution models. However, bivariate models previously used in similar discrete EDAs were only able to exploit an $O(d)$ subset of all the $O(d^2)$ bivariate variable dependencies between d variables. We introduce, and utilize in DICE, a model based on *dichotomised multivariate Gaussian distributions*. These models are able to capture and make use of all $O(d^2)$ bivariate variable interactions in binary and multary search spaces. This paper tests the performances of these new EDA models and algorithms on a suite of challenging combinatorial optimization problems, and compares their performances to previously used discrete-space bivariate EDA models. EDAs utilizing these new *dichotomised Gaussian* (DG) models exhibit significantly superior optimization performances, with the performance gap becoming more marked with increasing dimensionality.

Keywords: Dichotomised Gaussian models, EDAs, Combinatorial Optimization

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. The intent is that such models identify and capture pertinent data dependencies and other structures within fitter more promising candidate solutions. At each iteration, the model is used to generate a population of new candidate solutions. These are evaluated and the fitter solutions selected. These, then, are used to update the probabilistic model for the next iteration. Pseudocode for a canonical EDA is shown in Algorithm 1.

Algorithm 1 Pseudocode for a canonical EDA

```
set $t \longleftarrow 0$ (uniformly randomly generate an initial population $P_0$ composed of $n$ individuals)
while termination condition not met do
    Select a collection $P_t^*$ of $m$ candidate solutions from the current population $P_t$
    create an updated probabilistic model $M_t$ using $P_t^*$
    generate a new population by sampling from the probabilistic model $M_t$
    set $t \longleftarrow t + 1$
end while
```

Many different types of EDAs have been proposed for optimization in both continuous and discrete problem domains. Some of the earliest EDAs used relatively simple univariate models. Examples would include *Population-Based Incremental Learning* (PBIL) [3], the *Compact Genetic Algorithm* (cGA) [19] and the *Univariate Marginal Distribution Algorithm* (UMDA) [33]. Obviously, such simple models were going to be inadequate when used in the optimization of more complex problem domains. There was a natural progression in the use of models able to capture more complex problem dependencies and structures.

EDA utilizing models that could capture and exploit bivariate marginal distributions appeared fairly early on in the development of this field. For discrete spaces, the principal examples would be *Mutual Information Maximizing Input Clustering* (MIMIC) [10], *Combining Optimizers with Mutual Information Trees* (COMIT) [4], and the *Bivariate Marginal Distribution Algorithm* (BMDA) [35]. 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. An example of a bivariate EDA for continuous spaces would be the *Estimation of Multivariate Normal Algorithm* (EMNA) [29], which operates using an underlying multivariate Gaussian distribution model.

However, successively more expressive models, capable of capturing ever more complicated problem features, have been investigated. The *Extended Compact Genetic Algorithm* (ECGA) [20] used a marginal product model where the search space variables were partitioned into several variable groupings (using the *minimum description length* criterion) with the overall model being a product of multivariate marginal distributions. The use of graphical models and Bayesian networks has been the most popular approach. Some discrete search space examples would be the *Bayesian Optimization Algorithm* (BOA) [34] and the *Esti-*

mation of Bayesian Networks Algorithm (EBNA) [12], and, in continuous search spaces, the *Estimation of Gaussian Network Algorithm* (EGNA) [28].

In this paper, we propose an EDA approach for discrete search spaces based on dichotomised multivariate Gaussian distributions. These models can construct and generate candidate solutions relatively efficiently (with a cost complexity of $O(d^3)$). They also have the attractive property of capturing and using all of the possible $O(d^2)$ bivariate interactions between the d variables of a problem. As far as these authors are aware, all bivariate marginal distribution models previously used in discrete space EDAs have been restricted to using just $O(d)$ of these bivariate interactions.

The structure of the paper is as follows. Section 2 looks at related past work with bivariate EDAs. Section 3 begins with a short survey of the literature related to the simulation of correlated multivariate Bernoulli variables. This field is the source of the *dichotomised Gaussian* (DG) technique we apply here. The remainder of section 3 describes this method in detail.

Section 4 details our suite of combinatorial optimization problem domains and the configuration of the EDA algorithms we test and compare on them. Section 5 presents results and analysis for these experiments. Finally, in section 6, we give some conclusions and lay out some ideas for future work.

2 Bivariate EDAs

2.1 Minimum Spanning Tree Techniques

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)[3] [18] , which utilize such models have, therefore, long been widely available and used.

For discrete spaces, the only bivariate EDAs available up-to-now use restricted bivariate interaction models. 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 [8]. 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. The cost of this MST algorithm for arbitrary matrices is $O(d^2)$, which also gives the overall cost for the procedure. This tree along with the $d-1$ pairwise bivariate distributions involved is then used 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). They also have certain optimality properties. Out of the all such restricted trees or chains of $O(d)$ connections, they are the unique model that

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

minimizes Kullback–Leibler divergence from the original probability distribution. However, even if this approach identifies and makes use of the $O(d)$ most significant pairwise interactions, it still is the case that the vast majority are discarded. This increasingly impacts 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 [41], 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. Despite randomly constructing the dependency trees, they claim their algorithm, nonetheless, performed similarly to MIMIC on tests.

2.2 Copula EDAs

An interesting relatively-recent development in EDAs is the use of copula techniques (see [17] for a survey). Copulas are a statistical tool that allow a multivariate dependency to be decomposed into a univariate marginal distribution function and a *copula*, which describes the dependence structure between the variables. Both aspects can then be modelled separately. This can allow particular EDA models to be applied to a wider set of problem domains. For example, copulas might allow a Gaussian distribution model to be used even when the problem univariate marginal distribution itself is not Gaussian. One application of copulas techniques to bivariate EDAs was the development of a more general copula-based version of the MIMIC algorithm in [40].

A small number of authors have previously used multivariate Gaussian copulas models in EDAs. An example of this, which has some relevance to our work is [24]. There are some similarities between the general approach of the algorithm given in section III of that paper and our method here. Their algorithm is potentially capable of learning and exploiting all the bivariate marginal dependencies in the continuous problem domains examined in their paper. However, it would be difficult and very expensive to apply the rejection sampling procedure they describe to discrete spaces. That would involve the computation of rectangular integrals of arbitrary multivariate Gaussians. As pointed out in Alan Genz's book [14], beyond a very small number of dimensions, exact computation of these is difficult and expensive. Therefore, this method, while useful in continuous spaces, is not practical as a technique in discrete search spaces.

2.3 Computational Cost

A seeming advantage of the MST or chain techniques that have been used in all discrete bivariate EDAs up to now, is their relatively low $O(d^2)$ cost (the main cost bottleneck is in the construction of the MST) in building such a model. However, this does not factor in the $O(d^2n)$ cost in estimating all $O(d^2)$ bivariate interactions in a population of n candidate solutions. It is hard to see how this $O(d^2n)$ cost can be computationally escaped, particularly for algorithms that seek to use all such interactions. MIMIC, BMDA and previous algorithms (except for EDAM) necessarily need to perform such operations (they still need to measure and score all such interactions in order to find and use the $O(d)$ most significant ones) . Therefore, there is certainly scope to increase the cost complexity of the EDA model used to a similar order of $O(d^2n)$ and yet still maintain the same overall time cost complexity of the EDA.

The cost complexity of our new *dichotomised Gaussian* (DG) bivariate EDA model, which will be described in detail later, is $O(d^3)$. 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 minimum be at least 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. This extra computational scope was one of the primary motivations that spurred us to seek more accurate discrete bivariate models.

2.4 Kernel Methods

A second motivation has been our interest in the use of kernel methods [39] in the principled design of *Evolutionary Algorithms* (EA) and EA search operators [26,27]. At the core of every kernel model is a *kernel function* that is chosen to match the inherent statistical characteristics of the problem domain at hand. The core strategy of kernel methods is the so-called *kernel trick* [1]. This allows primarily linear algorithms, which principally operate using inner products, to be extended to implicitly and cheaply operate in richer and higher-dimensional kernel feature spaces $\mathcal{V}$ where the original problem is easier to linearly separate and/or model. For example, a standard linear classifier that could not effectively separate data in the original space might successfully separate these in some higher dimensional feature space via a kernel; this is the basis of *Support Vector Machine* (SVM) techniques [9] in classification and *Gaussian Random Functions* (GRFs) [39] in machine learning (also known as *Gaussian Processes*).

Our longer range goal is the use of kernel methods and the kernel trick in EDA design. The very earliest EDAs primarily used simple linear univariate models. The kernel trick is a way of non-linearly extending linear algorithms that work primarily via inner products. We feel a similar strategy might successfully be used to construct a new family of non-linear EDAs, which are capable of being easily tailored to the problem at hand via learnable kernel functions.

The family of polynomial kernel functions [16] and, in particular, the quadratic kernel function [7], which is quite popular in natural language processing, represent probably the very simplest kernel function special case (only just beyond basic linearity). It is possible to recast and reformulate existing continuous bivariate EDAs like CMA-ES and EMNA as kernelized versions of simpler linear algorithms (using as a framework such quadratic kernels and their associated feature spaces). We wished to do the same for the discrete bivariate EDA case, which sparked our interest in these DG models. Having tools like the DG model to deal with the basic quadratic kernel case seems essential if we are to hope to later construct even more general and powerful kernel EDAs.

3 Dichotomised Multivariate Gaussian Distribution Models

3.1 Simulation of Correlated Multivariate Bernoulli Variables

The literature concerning the generation of correlated binary vectors has a long history. A multivariate Bernoulli variable is, in principle, fully specified by $2^d - 1$ parameters (in effect, the individual probabilities for every possible bitstring 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 $O(d^2)$ 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 use the maximum entropy distribution for the situation where 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 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 to sample binary vectors from an Ising model even when one is found (one has to resort to expensive methods like the "perfect sampling" Markov chain Monte Carlo simulation method [36]). Therefore, many other methods have been proposed for simulating such correlated binary vectors.

Examples would include [6] that proposed a method using look-up tables of size $O(d^3)$, [30] that introduced two methods – one based on setting up a linear programming problem and another based on Archimedean Copulas, [13] that introduced an "iterative proportional fitting algorithm", and [25] that represents a more recent copula approach to this problem.

3.2 Dichotomised Gaussian Simulation of Correlated Binary and Multary Vectors

The particular technique, the *dichotomised Gaussian* (DG) method, that we have elected to use has been described and utilized in several past papers, the first description possibly being in [11]. A more recent exposition of the method can be found in [31,32]. 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.

The Basic Method Suppose we are dealing with the general case of 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^{th} dependent variable (gene). 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 $\left\{\hat{f}_i(c)\right\}_i$ 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, R is a sample correlation matrix estimated from a sample population of selected search space points, and the $\hat{f}_i(c)$ are empirical univariate marginal densities estimated from normalized allele frequency counts in the population.

At the core of the DG method is a d-dimensional multivariate normal distribution $\mathcal{N}(0, \Sigma)$ where $(\Sigma)_{i,j} = \rho_{ij}$ is its $d \times d$ correlation matrix (not to be confused with R). The DG method also makes use of a set of threshold values (with $a_i - 1$ threshold values needed for each dimension i):

$$\mathcal{T} = \left\{t_i^b,\, i \in \{1, \ldots, d\}, b \in \{0, \ldots, (a_i - 2)\}\right\}$$

For each variable i, these $a_i - 1$ threshold values partition $\Re$ into a_i disjoint intervals:

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

To randomly generate a multary search space point $\omega \in \Omega$, we first generate a continuous d-dimensional random vector x from the multivariate normal distribution. We, then, use these thresholds to convert (or dichotomise) this vector into a multary search space point. At each position i, x_i must belong to one of the a_i disjoint thresholded intervals: $K_i^0,\, K_i^1, \ldots, K_i^{a_i-1}$, so if $x_i \in K_i^c$ then, at the i^{th} position in the resulting multary vector ω, we set $\omega_i = c$.

Replicating the Marginal Univariate Densities We can set these thresholds to exactly replicate the target univariate marginal densities $\left\{\hat{f}_i(c)\right\}_i$. Let $\hat{F}_i(c) = \sum_{d=0}^{c} \hat{f}_i(d)$ be the target univariate marginal *cumulative distribution function* (CDF) for variable i. If we then calculate the thresholds in $\mathcal{T}$ according to $t_i^b = \Phi^{-1}(\hat{F}_i(b))$ where $\Phi^{-1}(x)$ is the standard inverse normal CDF function, then it can be easily verified that the univariate marginal densities of the resulting randomly generated multary vectors ω will indeed equal $\left\{\hat{f}_i(c)\right\}_i$.

Replicating Gene Correlations The next step in the DG method is to adjust each multivariate normal correlation value ρ_{ij} in Σ so that the resulting correlation $corr\,(\omega_i, \omega_j)$ between variables i and j in the simulated random multary vector ω equals the desired target gene correlation value r_{ij}.

We can use the efficient-to-evaluate standard bivariate normal CDF function $\Psi_2(x, y; \rho)$ to calculate bivariate marginal CDFs for the output vector ω as: $F_{ij}\,(b, c) = \mathrm{Prob}\,\{\omega_i \leq b \,\cap\, \omega_j \leq c\} = \Psi_2\left(t_i^b, t_j^c; \rho_{ij}\right), i \neq j$. For convenience, if we also define $F_{ij}(b, c)$ to be 0 whenever $b < 0$ or $c < 0$, then it is easy to calculate the bivariate marginal densities $\mathrm{f_{ij}(b, c)} = \mathrm{Prob}\,\{\omega_i = b \,\cap\, \omega_j = c\}$ for ω as:

$$\mathrm{f_{ij}(b,c)} = F_{ij}\,(b,c) - F_{ij}\,(b{-}1,c) - F_{ij}\,(b,c{-}1) + F_{ij}\,(b{-}1,c{-}1)$$

From these density values, the correlations $corr\,(\omega_i, \omega_j)$ between the variables of the generated search space points can be directly and easily calculated.

For each pair of variables, we need to solve for the unique ρ_{ij} that will produce a $\mathrm{corr}(\omega_i, \omega_j)$ value that matches the desired gene correlation r_{ij} in R. As pointed out in [31], these problems are monotonic and there is always a single unique solution for ρ_{ij}, guaranteed to lie within $[-1, 1]$. Straightforward and efficient one-dimensional bisection root-finding algorithms can be used to 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 and more detailed descriptions of this approach can be found in [11] and [31].

Repairing the Correlation Matrix The resulting Σ matrix may not always be positive semi-definite (in other words, may not be a valid correlation matrix). If it is not, however, then efficient algorithms exist to repair Σ by finding and replacing it with the nearest valid correlation matrix. A paper by Nicholas Higham [22] introduced the first algorithm for finding, for any arbitrary correlation 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 [37]. We used a publicly available[4] C-code version of this Newton-based algorithm in our implementation.

Generating Multary Search Space Points To actually generate search space points ω, standard Gaussian simulation algorithms are used to sample continuous vectors from the multivariate normal distribution $\mathcal{N}(0, \Sigma)$ we constructed. The set of threshold values $\mathcal{T}$ is then used to dichotomise these random normal vectors into multary vectors, which will have the exact target univariate marginal densities and the desired (or at least very close to) gene correlation behaviour. This DG simulation process has an overall computational cost of $O(d^3)$. Calculating each individual ρ_{ij} and each individual threshold has a low fixed cost unrelated to the dimensionality d. The most expensive step is correlation matrix repair (due to the matrix operations involved) with cost $O(d^3)$. Pseudocode for this process is given in Algorithm 2.

3.3 Knowledge Incorporation

In our experiments, we used standard correlation matrix repair methods that found the correlation matrix that was nearest in terms of the Frobenius norm: $\|A\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{i,j}|^2}$. However, versions of these correlation repair algorithms are available that use various types of *weighted* Frobenius norms. For

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

Algorithm 2 Pseudocode for the dichotomised Gaussian simulation of correlated binary and multary vectors

```
Estimate the empirical univariate marginal densities $\{\hat{f}_i(c)\}_i$ and the sample correlation matrix $R$ from the selected population of search space points.
Calculate the set of threshold values $\mathcal{T}$ that will replicate the target univariate marginal densities.
Individually calculate, using a fast root-finding technique, the $\rho_{ij}$ values in $\Sigma$ that replicate the desired gene correlations.
If necessary, use a nearest correlation algorithm to repair the correlation matrix $\Sigma$.
Generate multary search space points by sampling random normal vectors from $\mathcal{N}(0, \Sigma)$, and dichotomising these into allele values using the thresholds in $\mathcal{T}$.
```

example, one such type of weighted Frobenius norm, dubbed the H-norm, is described in [22]: $\|A\|_H = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} w_{i,j} \left|a_{i,j}\right|^2}$ where $w_{i,j} \geq 0$. An efficient Newton-based nearest correlation matrix algorithm that uses this H-norm is presented in [38].

The use of such weights would allow us to naturally and easily incorporate into the DG model acquired/prior knowledge about the relative strengths or significances of individual bivariate interactions. We could assign larger weights to interactions we believe will have a greater impact on fitness. While not tested here, we plan to investigate versions of DICE that can incorporate prior knowledge of this type in the near future.

4 Experimental Setup

Our goal was to test and compare the performance this new dichotomised Gaussian EDA model with other existing discrete bivariate EDA models. To ensure an absolutely fair model comparison, we used the same basic EDA algorithm with identical settings with each EDA model.

We chose a test suite of seven challenging combinatorial optimization problem domains, deliberately selected so their dimensions could be easily varied.

4.1 EDA Algorithm Settings

All the EDAs used a population of 200 individuals. All algorithms were run for 100 generations. At each iteration, the probability model was used to generate 200 new individuals. The 100 fittest of these 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 estimated univariate and/or bivariate marginal densities (estimated from current and previous selected populations). We used an exponentially-decaying weighted average to combine present and past 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. Hence, the current model would be based on a weighted combination:

$H_t^* = H_t + \tau H_{t-1} + \tau^2 H_{t-2} + \tau^3 H_{t-3} + \ldots$ of present and past univariate and/or bivariate marginal population histograms.

Extensive empirical testing determined that $\tau = 0.7$ was the best general setting for the EDAs we examined. An identical $\tau = 0.7$ setting was used for all runs. Batches of 100 runs were used to produce all the experimental results given below.

We compared our DG model against 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 [41]).

4.2 Problem Domain Set

Five well-known combinatorial optimization problem domains defined on bit-string search spaces were used; these are described in more detail in Table 1. Three NK-Landscape instances with K=2, 3 and 4 resulted in a total test suite size of seven problem domains. We have also included results on the simple linear "Counting Ones" problem for comparison, but these are not included in the test suite averages. 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.

Problem Domain	**References**	**Fitness Function Formula/Details**
Counting Ones (OneMax)		$f(x) = \sum_{i=1}^{d} w_i (2x_i - 1)$, $w_i \sim \mathcal{N}(0, \frac{1}{d})$.
QUBO (Quadratic Unconstrained Binary Optimization)	[5]	$f(x) = \sum_{i,j=1}^{d} w_{ij} (2x_i - 1)(2x_j - 1)$, $w_{ij} \sim \mathcal{N}(0, \frac{1}{d^2})$.
CUBO (Cubic Unconstrained Binary Optimization)	[15]	$f(x) = \sum_{i,j,k=1}^{d} w_{ijk} (2x_i - 1)(2x_j - 1)(2x_k - 1)$, $w_{ij} \sim \mathcal{N}(0, \frac{1}{d^3})$
NK-Landscapes	[2]	"Random neighbourhood" model (without replacement); $K = 2, 3, 4$
K-Uniform MAX-SAT (with $20\,n$ random clauses)	[23]	$K = 3$ (variables per clause)
Weighted MAX-CUT	[21]	$f(x) = \sum_{i,j=1}^{d} w_{ij} (x_i \oplus x_j)$, $w_{ij} \sim \mathcal{U}(-\sqrt{\frac{3}{d^2}}, \sqrt{\frac{3}{d^2}})$.

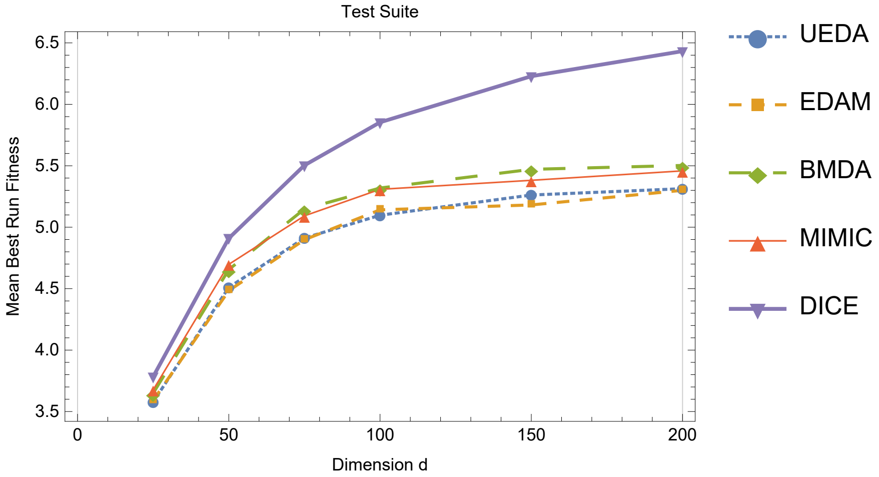
Table 1: Problem Domain Set Details

5 Results and Analysis

Fig. 1 gives mean best run fitness performances for the EDA models averaged over the test suite for problem dimensions d=25, 50, 75, 100, 150 and 200. Clearly, DICE, utilizing the DG model, has overall the best performances. Its mean test suite performance always is superior to its next nearest competitor. For $d = 25$, the narrow 2.4% performance gap to second-place MIMIC is significant using a 95% confidence interval in a two-tailed student-t test. The performance gap seems to gradually increase as the dimensionality increases. The performance gaps over its next competitor at higher dimensionalities are significant at a 99% level using the same significance test.

The accuracy gap between the DG model and the other models is only likely to become greater as dimensionality increases, as a greater and greater proportion of the bivariate interactions in the other models go unexploited. This may explain the seeming gradually increasing performance gap (the DG model is barely ahead of the nearest runner-up at $d = 25$ with a 2.4% gap, is 4.5% ahead at $d = 50$, is 7.0% ahead at $d = 75$, is 17.2% ahead at $d = 100$, and is 16.9% ahead at $d = 200$).

Fig. 1: Test Suite Mean Best Run Fitnesses for the EDA models



Figures 2 and 3 provide individual plots of relative EDA model performances for all eight problem domains tested.

All EDA models seemed able to adequately cope with the simple OneMax "Counting Ones" problems. Similar performances were demonstrated by all algorithms.

Not a great deal separates the performances of the BMDA and MIMIC models (they essentially differ only in how they score bivariate interactions). Unsurprisingly, they are generally superior to the simpler univariate (UEDA) model.

Fig. 2: Mean Best Run Fitnesses for the EDA models on the OneMax, QUBO, CUBO and MAX-CUT problem domains

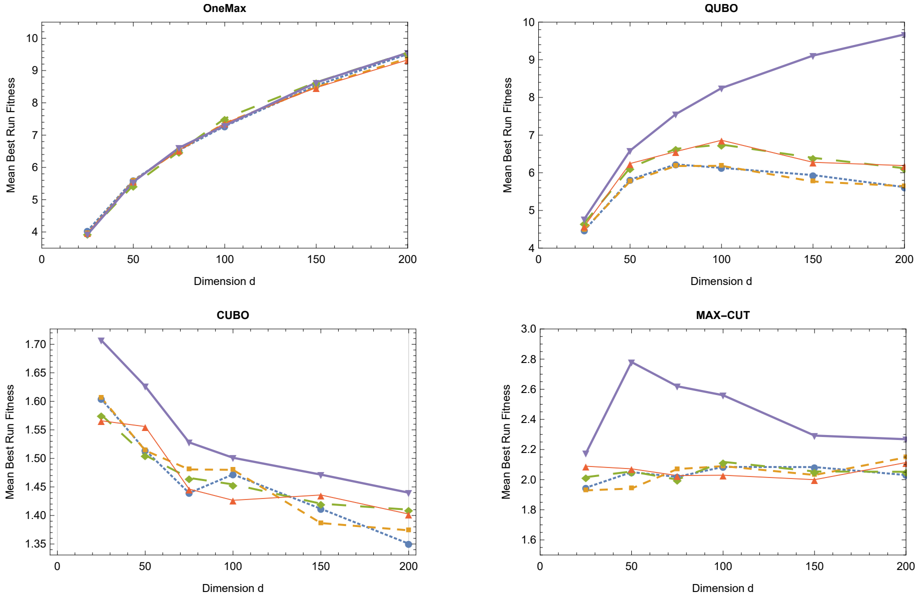
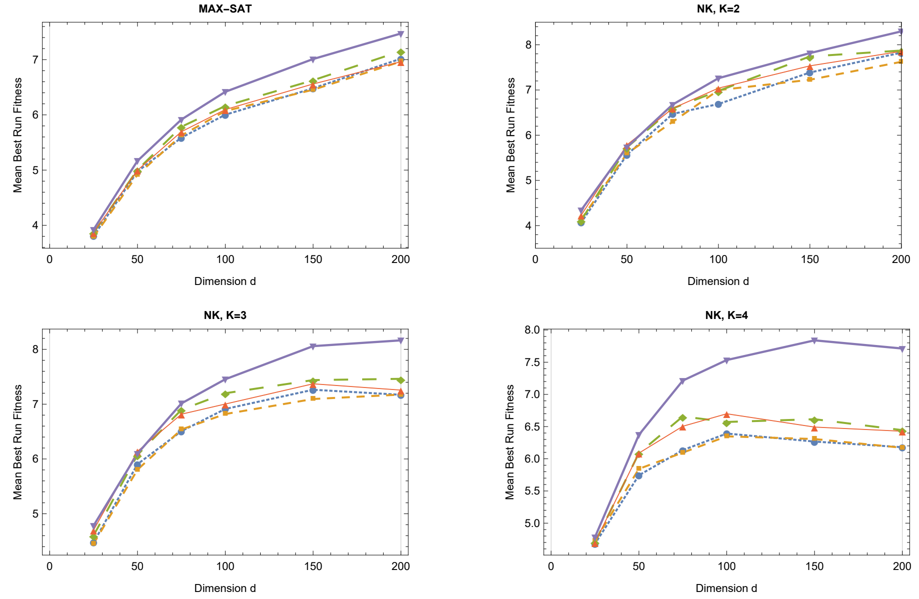


Fig. 3: Mean Best Run Fitnesses for the EDA models on the MAX-SAT and NK-Landscape (with K=2, 3 and 4) problem domains



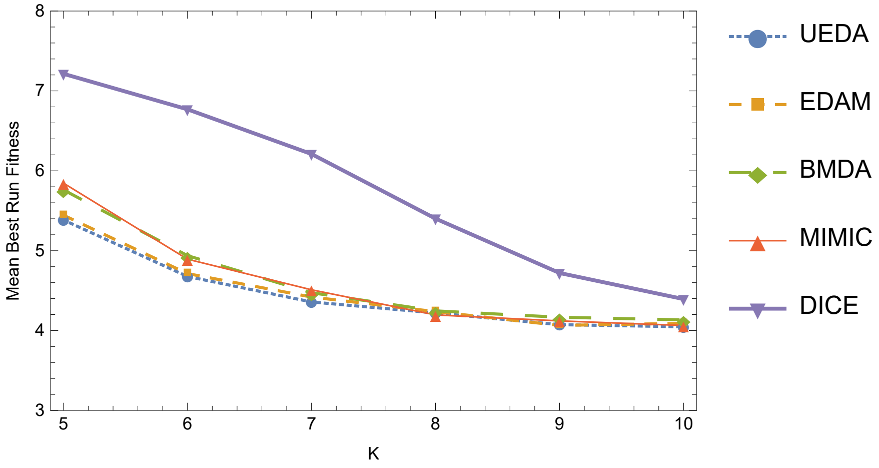
The simple random tree (EDAM) approach also performs disappointingly compared to the other bivariate models. Seemingly, the extra effort that the BMDA and MIMIC models expend in identifying the most significant $O(d)$ of the bivariate interactions does pay off in better performance.

The DG model performs best on QUBO. QUBO's quadratic structure is similar in nature to DG model's own underlying model, consisting solely of bivariate interactions.

This purely quadratic structure, however, does not hold true for some of the other problem domains. While the DG model's superior performance is not as marked on these, nonetheless it still seems to be able to exploit the predominantly higher order interactions present in problems such as the NK-Landscapes with higher values of K. This promisingly seems to indicate that combinations of lower order bivariate interactions may be usefully approximating and guiding the search towards fitter higher order interactions. To investigate this behaviour further, we tested the behaviour of all of the EDA models on a range of NK-Landscape problems with $d = 100$ as K ranged from 5 up to 10. The results can be seen in Fig. 4.

DICE maintains a very healthy performance margin over all the other models as K increases. Intriguing, as the interaction order increases towards ten, the relative performance differences between all models start to narrow. This may be indicating that all of the models are becoming less capable of dealing with interactions involving increasingly large numbers of variables (a point worthy of future investigation).

Fig. 4: Mean Best Run Fitnesses for the EDA models on NK-Landscapes with varying K values



6 Conclusions

In this paper, we introduced a new (almost) fully bivariate model for discrete EDAs, based on dichotomised Gaussian models. At lower dimensions, DICE was competitive with otherwise identical EDAs that used other more established bivariate models (as found in MIMIC and BMDA). At intermediate dimensions, superior performance began to be exhibited by our model. With increasing dimensionality, this performance gap became even more pronounced. In this initial investigation, these models have exhibited much promise.

6.1 Future Work

A much more comprehensive investigation of this EDA modelling technique will be necessary on a much wider variety of problem domains.

It is very likely that memetic algorithm approaches could be profitably combined with DICE. Local search techniques are particularly useful for combinatorial optimization problems. The earlier COMIT EDA successfully combined a local hillclimber with its MST-based limited bivariate model.

We envisage that DG models could be usefully applied elsewhere in evolutionary computation. For example, it should be possible to construct crossover operators able to capture and respect arbitrary pairwise dependencies between variables (DG models could be used to generate crossover masks). This would provide increased opportunities to better tailor EA search operators to the characteristics of the problem domain at hand.

We are also investigating combining the DG model with CMA-ES, which is a popular and powerful EDA-like optimizer for continuous search spaces. Our DG model should be able to allow CMA-ES to be efficiently extended to discrete search spaces.

Acknowledgements

This work was supported, in part, by Science Foundation Ireland grant 10/CE/I1855 to Lero - the Irish Software Engineering Research Centre (`www.lero.ie`).

References

1. Aizerman, A., Braverman, E., Rozoner, L.: Theoretical foundations of the potential function method in pattern recognition learning. Automation and remote control 25, 821–837 (1964)
2. Altenberg, L.: NK fitness landscapes. Handbook of evolutionary computation 7, 5–B2 (1997)
3. Baluja, S., Caruana, R.: Removing the genetics from the standard genetic algorithm. In: Machine Learning: Proceedings of the Twelfth International Conference. pp. 38–46 (1995)

4. Baluja, S., Davies, S.: Using optimal dependency-trees for combinational optimization. In: Proceedings of the Fourteenth International Conference on Machine Learning. pp. 30–38. Morgan Kaufmann Publishers (1997)
5. Boros, E., Hammer, P., Tavares, G.: Local Search Heuristics for Quadratic Unconstrained Binary Optimization (QUBO). Journal of Heuristics 13(2), 99–132 (Apr 2007)
6. Caprara, A., Furini, F., Lodi, A., Mangia, M., Rovatti, R., Setti, G.: Generation of antipodal random vectors with prescribed non-stationary 2-nd order statistics. IEEE Transactions on Signal Processing 62(6), 1603–1612 (2014)
7. Chang, Y.W., Hsieh, C.J., Chang, K.W., Ringgaard, M., Lin, C.J.: Training and testing low-degree polynomial data mappings via linear SVM. Journal of Machine Learning Research 11(Apr), 1471–1490 (2010)
8. Chow, C., Liu, C.: Approximating discrete probability distributions with dependence trees. IEEE transactions on Information Theory 14(3), 462–467 (1968)
9. Cristianini, N., Shawe-Taylor, J.: An introduction to support vector machines and other kernel-based learning methods. Cambridge University Press (2000)
10. De Bonet, J., Isbell, C., Viola, P., et al.: MIMIC: Finding optima by estimating probability densities. Advances in Neural Information Processing Systems pp. 424–430 (1997)
11. Emrich, L., Piedmonte, M.: A method for generating high-dimensional multivariate binary variates. The American Statistician 45(4), 302–304 (1991)
12. Etxeberria, R., Larranaga, P.: Global optimization using Bayesian networks. In: Second Symposium on Artificial Intelligence (CIMAF-99). pp. 332–339. Habana, Cuba (1999)
13. Gange, S.: Generating multivariate categorical variates using the iterative proportional fitting algorithm. The American Statistician 49(2), 134–138 (1995)
14. Genz, A., Bretz, F.: Computation of multivariate normal and t probabilities, vol. 195. Springer Science & Business Media (2009)
15. Glover, F., Hao, J.K., Kochenberger, G.: Polynomial unconstrained binary optimisation – part 2. International Journal of Metaheuristics 1(4), 317–354 (2011)
16. Goldberg, Y., Elhadad, M.: SplitSVM: fast, space-efficient, non-heuristic, polynomial kernel computation for NLP applications. In: Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers. pp. 237–240. Association for Computational Linguistics (2008)
17. González-Fernández, Y., Soto, M.: A survey of estimation of distribution algorithms based on copulas. Tech. rep.
18. Hansen, N., Kern, S.: Evaluating the CMA evolution strategy on multimodal test functions. In: International Conference on Parallel Problem Solving from Nature. pp. 282–291. Springer (2004)
19. Harik, G., Lobo, F., Goldberg, D.: The Compact Genetic Algorithm. IEEE Transactions On Evolutionary Computation 3(4), 287–297 (1999)
20. Harik, G., Lobo, F., Sastry, K.: Linkage Learning via Probabilistic Modeling in the Extended Compact Genetic Algorithm(ECGA). In: Scalable Optimization via Probabilistic Modeling, pp. 39–61. Springer (2006)
21. Heras, F., Larrosa, J., Oliveras, A.: MiniMaxSAT: An efficient weighted Max-SAT solver. J. Artif. Intell. Res.(JAIR) 31, 1–32 (2008)
22. Higham, N.: Computing the nearest correlation matrix : a problem from finance. IMA journal of Numerical Analysis 22(3), 329–343 (2002)
23. Hoos, H., Stützle, T.: Stochastic Local Search: Foundations & Applications. Elsevier (2004)

24. Hyrš, M., Schwarz, J.: Multivariate Gaussian copula in estimation of distribution algorithm with model migration. In: Foundations of Computational Intelligence (FOCI), 2014 IEEE Symposium on. pp. 114–119. IEEE (2014)
25. Jin, R., Wang, S., Yan, F., Zhu, J.: Generating spatial correlated binary data through a copulas method. Science Research 3(4), 206–212 (2015)
26. Lane, F., Azad, R., Ryan, C.: Principled evolutionary algorithm design and the kernel trick. In: Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion. pp. 149–150. ACM (2016)
27. Lane, F., Azad, R., Ryan, C.: Principled evolutionary algorithm search operator design and the kernel trick. In: 2016 IEEE Symposium on Model Based Evolutionary Algorithms (IEEE MBEA'16), part of the IEEE Symposium Series on Computational Intelligence 2016 (2016)
28. Larrañaga, P., Etxeberria, R., Lozano, J., Peña, J.: Combinatorial optimization by learning and simulation of Bayesian networks. In: Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence. pp. 343–352. Morgan Kaufmann Publishers Inc. (2000)
29. Larranaga, P., Lozano, J., Bengoetxea, E.: Estimation of distribution algorithms based on multivariate normal and Gaussian networks. Tech. rep., EHU-KZAA-IK-1 (2001)
30. Lee, A.: Generating random binary deviates having fixed marginal distributions and specified degrees of association. The American Statistician 47(3), 209–215 (1993)
31. Macke, J., Berens, P., Ecker, A., Tolias, A., Bethge, M.: Generating spike trains with specified correlation coefficients. Neural Computation 21(2), 397–423 (2009)
32. Macke, J., Murray, I., Latham, P.: How biased are maximum entropy models? In: Advances in Neural Information Processing Systems. pp. 2034–2042 (2011)
33. Mühlenbein, H.: The equation for response to selection and its use for prediction. Evolutionary Computation 5(3), 303–346 (1997)
34. Pelikan, M., Goldberg, D., Cantú-Paz, E.: BOA: The Bayesian optimization algorithm. In: Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 1. pp. 525–532. Morgan Kaufmann Publishers (1999)
35. Pelikan, M., Mühlenbein, H.: The bivariate marginal distribution algorithm. In: Advances in Soft Computing, pp. 521–535. Springer (1999)
36. Propp, J., Wilson, D.: Exact sampling with coupled Markov chains and applications to statistical mechanics. Random structures and Algorithms 9(1-2), 223–252 (1996)
37. 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)
38. Qi, H., Sun, D.: An augmented Lagrangian dual approach for the H-weighted nearest correlation matrix problem. IMA Journal of Numerical Analysis 31(2), 491–511 (2011)
39. Rasmussen, C., Williams, C.: Gaussian processes for machine learning. 2006. The MIT Press, Cambridge, MA, USA (2006)
40. Salinas-Gutiérrez, R., Hernández-Aguirre, A., Villa-Diharce, E.: Using copulas in estimation of distribution algorithms. In: Mexican International Conference on Artificial Intelligence. pp. 658–668. Springer (2009)
41. Zhang, Q., Sun, J., Tsang, E., Ford, J.: Estimation of Distribution Algorithm based on Mixture. Tech. rep.