Two approaches of using heavy tails in high dimensional EDA

Momodou L. Sanyang
School of Computer Science
University of Birmingham
Edgbaston, UK, B15 2TT
Email: M.L.Sanyang@cs.bham.ac.uk

Hanno Muehlbrandt
School of Computer Science
University of Birmingham
Edgbaston, UK, B15 2TT
Email: HXM239@student.bham.ac.uk

Ata Kaban
School of Computer Science
University of Birmingham
Edgbaston, UK, B15 2TT
Email: A.Kaban@cs.bham.ac.uk

Abstract—We consider the problem of high dimensional black-box optimisation via Estimation of Distribution Algorithms (EDA). The Gaussian distribution is commonly used as a search operator in most of the EDA methods. However there are indications in the literature that heavy tailed distributions may perform better due to their higher exploration capabilities. Univariate heavy tailed distributions were already proposed for high dimensional problems. In 2D problems it has been reported that a multivariate heavy tailed (such as Cauchy) search distribution is able to blend together the strengths of multivariate modelling with a high exploration power. In this paper, we study whether a similar scheme would work well in high dimensional search problems. To get around of the difficulty of multivariate model building in high dimensions we employ a recently proposed random projections (RP) ensemble based approach which we modify to get samples from a multivariate Cauchy using the scale-mixture representation of the Cauchy distribution. Our experiments show that the resulting RP-based multivariate Cauchy EDA consistently improves on the performance of the univariate Cauchy search distribution. However, intriguingly, the RP-based multivariate Gaussian EDA has the best performance among these methods. It appears that the highly explorative nature of the multivariate Cauchy sampling is exacerbated in high dimensional search spaces and the population based search loses its focus and effectiveness as a result. Finally, we present an idea to increase the chance of jumps and hence aid exploration of the search space [23].

While this is certainly true for low dimensional problems, building a reliable model in high dimensions is a difficult problem that, if treated naively, requires prohibitive search costs or else EDA type methods decline in performance very quickly as the search space dimensionality increases. Competitions held for large scale optimisations recently (e.g. [3], [2], [1]), had winning algorithms that followed approaches other than EDA: For example Local Search was combined with methods of Genetic Algorithms [15] or Trajectory Searches [24]. EDAs have been the subject of many recent research efforts to find ways to exploit their potential for high dimensional problems. One of the proposals has been to use a univariate heavy tail search distribution to increase the chance of jumps and hence aid exploration of the search space [23].

However, univariate search distributions are limited to separable problems, and it was found on 2D problems that a multivariate heavy tailed (such as Cauchy) search distribution is able to blend together the strengths of multivariate modelling of dependencies between the search variables and the high exploration power provided by a heavy tailed distribution. In this paper, we investigate whether a similar scheme would work well in high dimensional search problems. We also devise another approach employing heavy tailed distributions as a means to enhance a random projection based multivariate Gaussian EDA.

I. INTRODUCTION

Optimisation is at the core of many scientific and engineering problems and has applications in many disciplines e.g. where one has to optimise set of parameters. Mathematical optimisations only deals with very specific problem types while search heuristics, like evolutionary computation, work in a black box manner [22]. The latter don’t have the guarantees that mathematical optimisation have, but are less specialised and often work sufficiently well on a wider range of functions. This paper is concerned with a heuristic optimisation approach known as Estimation of Distribution Algorithms (EDAs), and we focus on continuous valued search spaces.

Estimation of Distribution Algorithms (EDAs) are a class of optimizations methods that, in contrast with classical Evolutionary Algorithms, guides the search to the optimum by building and sampling explicit probability models of selected candidate solutions [17]. Pelikan and Hausschild [10] provide a large overview of successful applications of different EDA techniques and their specific advantages that stem from learning the structure of the search space.

A typical EDA algorithm proceeds by initially generating a population of $N$ individuals. At each generation $t$, the current population $P$ is evaluated using the objective (or fitness) function. The most promising individuals with best fitness values are selected to build a probabilistic model and sample the next generation of candidate solutions from this model. A popular selection scheme used by many EDA variants is truncation selection. Truncation selection simply defines a percentage of individuals $\tau$ to be selected out of the entire population of size $N$ and selects the subset $P_{sel} = \text{round}(\tau N)$ individuals of the population. The parameter $\tau$ controls the selection pressure. Figure 1 illustrates the procedure of a simple EDA implementation and the pseudocode of a simple EDA can be found in Algorithm 1 which is adopted from [22].
Algorithm 1: The Pseudocode of a simple EDA with Population size $N$

1. Set $t \leftarrow 0$.
2. Set $P \leftarrow$ Generate $N$ points randomly to give an initial population.
   Do
   3. Evaluate fitness for all $N$ points in $P$
   4. Select some individuals $P^\text{sel}$ from $P$
   5. Calculate the sample statistics of $P^\text{sel}$
   6. Use the statistic to sample new population $P^\text{new}$
   7. $P \leftarrow P^\text{new}$
 Until Termination criteria are met

where $d$ is the dimensionality of the search space.

For problems that are not separable, the design variables have inter-dependencies. One of the first approaches to capture those inter-dependencies has been the MIMIC algorithm which was proposed by De Bonet et. al in [11]. This model has the potential of capturing bivariate interactions between decision variables by sampling from the pairwise joint distribution between variables according to Equation 2.

$$P_x(X) = P(x_{i_1}|x_{i_2}) \cdot P(x_{i_2}|x_{i_3}) \cdot \cdots \cdot P(x_{i_{d-1}}|x_{i_d})$$

This means that all decision variables are treated as if they were independent throughout the model building process. The probabilistic model has the potential of capturing bivariate interactions between decision variables by sampling from the pairwise joint distribution between variables according to Equation 2.

$$P(x) = \prod_{i=1}^{d} P(x_i)$$

where $d$ is the dimensionality of the search space.

Despite MIMIC’s ability to outperform univariate models, the majority of optimization problems will have larger groups of interacting design variables. Several proposals have advanced to explicitly capture multivariate dependencies by building graphical dependency networks, for example Bayesian Networks [25]. But, from statistical, computational and memory points of view, learning probabilistic graphical models is highly expensive [4]. Thus, the scaling up of these model building processes to high dimensional problems is challenging. Efforts to mitigate such problems have resulted in several algorithms being proposed recently. For example, an approach that is correlation based such as in [6] and a variable interaction learning process such as in [27] were among such proposals.

Neglecting all or the majority of correlations is often the method of choice in high dimensional problems in order to make model building feasible. To compensate for this, heavy tailed search distributions have been proposed and demonstrated to work better in high dimensions [23]. The rationale is that sampling from a heavy tailed distribution can spread out the population via longer jumps and thus the capability to explore the search space is increased. However, it has also been found in 2D problems that a multivariate heavy-tailed distribution works better than both the univariate one and the multivariate Gaussian [22], hence in this work we are interested if this would also be the case in high dimensions.

B. Related methods and state of the art

Although many approaches have been proposed, we will limit ourselves to a few most relevant ones. They may be categorised into approaches that uses marginalisation, divide and conquer, limited dependencies, clustering, and evolutionary strategy techniques.

As already mentioned, UMDAc does not consider variable interaction. It can be said that UMDAc samples new individuals from a Gaussian with diagonal covariance matrix. It takes the variance of each decision variable into account, but no covariance among different variables. To get around this limitation, eigen decomposition EDA (ED – EDA) [28] samples individuals from a full covariance matrix and, thus gives it the ability to capture interaction among decision variables.

Multilevel Cooperative Co-evolution (MLCC), proposed in [30] is a framework that groups the decision variables of a problem randomly. It employs a technique called random grouping in order to group interacting variables in one subcomponent to tackle problems in high-dimensional optimization.

1) Model estimation in EDAs: The basic skeleton of most EDA approaches is very similar to the one described in Algorithm 1. But the cream of each EDA variant is the process of building its model. The probabilistic model has a fundamental role and influence on the performance of the algorithm, since it is most of the time the only tool used for guiding the search to the global optimum [16]. Thus, the model must represent the search space in a great detail, but the required search cost (i.e. number of fitness evaluations used) and computational resources have to be appropriate as well. This becomes a problem when the search space is high dimensional, and therefore simplified models such as univariate models are frequently used instead of a full multivariate model. This means that all decision variables are treated as if they were independent throughout the model building process, and for each decision variable a separate model is estimated. An early approach is the UMDAc, which is described in [18]. The model building in this univariate approach is done using Equation 1 [18].

$$P(x) = \prod_{i=1}^{d} P(x_i)$$

In the top right and bottom left graphs, the population has converged close to the global optimum. In the top right graph, the population has converged close to the global optimum.
The groups are optimized jointly but separately from other groups.

EDA with Model Complexity Control (EDA − MCC) [6] employs a deterministic algorithm to group variables. It first of all splits all decision variables into two independent subsets, one set contains decision variables with only minor interaction with other variables and the other contains strongly dependent variables. If the absolute value of the correlation of the variables with every other variable is below a certain threshold, they are considered to be weakly dependent. But, if the values are at least one, they are considered to be strongly dependent.

Cooperative Co-evolution with Variable Interaction Learning (CCVIL) proposed by Weicker et al. in [27] is a deterministic method to uncover dependencies between decision variables, which has later been extended to the CCVIL framework by Chen et al in [26]. In contrast to the other algorithms mentioned, CCVIL tries to explicitly find variable interaction.

Covariance Matrix Adaptation- Evolutionary Strategy (CMA-ES) [9] is widely considered as one of the leading optimization techniques [5]. Even though it is titled as an Evolutionary Strategy, its main concepts are very similar to the ones of a regular EDA [14]. CMA-ES and its variants have been very successfully used on many low-dimensional problems (e.g. [7], [8]), but benchmarks on high dimensional problems are rather rare. A variant of (CMA-ES) called separable CMA-ES (sep-CMA-ES) [21] does not sample from a full covariance matrix, but rather from a diagonal covariance. This procedure replaces the requirement to perform eigendecomposition for sampling and therefore reduces the complexity per generation to linear in the problem dimension [21].

A recent approach with state of the art performance [13] introduces an ensemble of random projections (RP) to low dimensions. This way the full covariance is compressed but no correlations are explicitly discarded. The compression is done on the set of fittest search points. The algorithm does its estimation and sampling job in the low dimensional space instead of doing them in the high dimensional space, which makes it efficient. The new population is created and returned to live in the full search space by combining populations from several low dimensional subspaces [13].

III. A LARGE SCALE MULTIVARIATE CAUCHY-EDA

In this section we devise a modified version of the RP-based large scale EDA of [13] which will replace its multivariate Gaussian search distribution with a multivariate Cauchy.

It is a well known fact that a multivariate distribution as a search operator has the advantage of modelling the correlation structure of the search variables. Univariate distributions lack this advantage. However, the Gaussian distribution as a search operator often convergences prematurely when the population is far from the optimum [17], [22]. It has been suggested recently that replacing the univariate Gaussian with a univariate Cauchy distribution in EDA has positive impact in alleviating this problem because it is able to make larger jumps in the search space due to the Cauchy distribution’s heavy tails [29]. A recent paper [22] used multivariate Cauchy to blend the advantages of multivariate modelling with the Cauchy samplings ability of escaping early convergence to efficiently explore the search space. The main drawback of [22] is that all the experiments where benchmarked on 2 dimensional problems. The approach in this section extends [22] to higher dimensional problems leveraging the recent techniques of random projections for optimization [13]. Our goal here is to test whether the conclusions drawn from 2D experiments in [22] still hold in high dimensional problems when the search distribution is multivariate Cauchy.

1) Algorithm presentation: The method we build on is a Multivariate Gaussian random projection EDA, which shall be denoted as MGrpEDA. Our Multivariate Cauchy random projection EDA will be referred to as MCrpEDA. The pseudocode of MGrpEDA is shown in algorithm 2 as was proposed in [13].

Algorithm 2 Pseudocode of the Multivariate Gaussian random projection EDA (MGrpEDA)

Inputs: $k, M, N, MaxFE$
(1) Set $t \leftarrow 0$.
(2) Set $P \leftarrow$ Generate $N$ points randomly to give an initial population.

Do
(3) Evaluate fitness for all $N$ points in $P$
(4) Select some individuals $P^{fit}$ from $P$
(5) Estimate $\mu := mean(P^{fit})$
(6) Generate $M$ independent random matrices $R_i$ with entries drawn iid from a 0-mean Gaussian with variance $\frac{1}{d}$.
(7) For $i = 1, ..., M$.
(a) Project the centred points into k-dimensions:
   $Y_i^R := [R_i(x_n - \mu); n = 1, ..., N]$.
(b) Estimate the $k \times k$ sample covariance $\Sigma_i^R$.
(c) Sample $N$ new points $y_1^{R_i}, ..., y_N^{R_i} \sim_{i.i.d} N(0, \Sigma_i^R)$.
EndFor
(8) Let the new population $P^{new} := \sqrt{\frac{dM}{k}} [\sum_{i=1}^{M} R_i^T y_1^R, ..., \sum_{i=1}^{M} R_i^T y_N^R] + \mu$.
(9) $P \leftarrow P^{new}$

Until Termination criteria are met or MaxFE exceeded

Output $P$

MGrpEDA proceeds by initially generating a population of individuals randomly everywhere and selects the $N$ fittest points based on their fitness values. This is represented as $P^{fit}$ in algorithm 2. The number of subspaces is denoted by $M$ which is a parameter. These subspaces are created in order to project the fittest individuals down to these subspaces with dimensionality, $k \ll d$. The latter is also a parameter of the method. The other input parameters are the population size $N$ and the maximum fitness evaluations allowed MaxFE. Once the $P^{fit}$ is determined, its mean is estimated in step (5) to be used in centering the points. Since we are going to have $M$ subspaces, $M$ independent random projection matrices are generated in step (6) so as to project the fittest individuals down to $k$ dimensions in these $M$ subspaces. We use the default option of RP matrices with i.i.d. Gaussian entries. Step 7(a) projects the fittest individuals down to the subspaces of dimension $k$, then estimates the $k \times k$ covariance matrices for each of the subspaces and samples $N$ new points in each
subspace using the multivariate Gaussian distribution. Step (8) averages the individuals obtained from the different subspaces to produce new population \( P \). This is the ‘combine’ stage of the algorithm which outputs the new population, and it can be shown that, conditioned on the projection matrices, this new population is distributed as a multivariate Gaussian with covariance that is a regularised version of the sample covariance. More precisely:

**Proposition 1:** Conditionally on all \( R_i, \ i = 1,...,M \), the new generation produced at Step 8 of Algorithm 2 is i.i.d. Gaussian with mean \( \mu \) and the following \( d \times d \) covariance matrix:

\[
\Sigma_{rp} = \frac{d}{kM} \sum_{i=1}^{M} R_i^T R_i \Sigma R_i^T R_i
\]

where \( \Sigma \) is the sample covariance of the original selected individuals in \( P^{fit} \).

We build on this observation to modify the algorithm to sample from multivariate Cauchy instead.

**The modified Step 8 of Alg. 2 that samples from multivariate Cauchy:** The Multivariate Cauchy density function is defined as:

\[
f(x; \mu, \Sigma_{rp}, d) = \frac{1}{\pi^d} \frac{1}{\left| \Sigma_{rp} \right|^\frac{1}{2} \left[ 1 + (x - \mu)^T \Sigma_{rp}^{-1} (x - \mu) \right]^\frac{1}{2}}
\]

where \( x \) is the random variable, \( d \) the dimension of the search space, \( \mu \) the location parameter and \( \Sigma_{rp} \) is a matrix valued parameter for which we use the covariance from Eq.(3).

We will use the fact that this probability density function can also be written as a Gaussian scale mixture, that is a convolution of Gaussian and a Gamma [19]. So our search distribution will be:

\[
\int_{u \geq 0} N(x; \mu, \Sigma_{rp}) \text{Ga}(u; \frac{1}{2}, \frac{1}{2}) du
\]

where \( u \) may be regarded as an unobserved variable.

Now, since we saw in equation (3) that conditionally on \( R_i \)'s Step (8) of Algorithm 2 already provides i.i.d. samples from a Gaussian, therefore to transform these into analogous conditionally independent Cauchy samples for our Cauchy-EDA sampling step, we just take the multivariate Gaussian sample that this step creates, and before adding the \( \mu \) to it, we draw a sample of \( N \) points \( u_1, ..., u_N \) i.i.d. from the univariate Gamma distribution and divide those Gaussian sample points with the square root of the Gamma distributed sample points. In other words, we use the sample mean and the RP-based regularised sample covariance as parameters for sampling from a multivariate Cauchy. This is in analogy with what has been done e.g. in [22] in 2D.

**IV. A Multivariate Gaussian-EDA with t distribution entries in R**

In this section we devise a different variant of the RP-based large scale multivariate Gaussian EDA of [13]. Now we are going to stay with the multivariate Gaussian search distribution but employ the heavy tailed t-distribution with low degrees of freedom to form the entries of the RP matrices \( R_i, \ i = 1,...,M \) in step(6) of Algorithm 2. Unlike before, we sample these entries \( i.i.d \) from a t-distribution with mean 0 and variance \( 1/d \). The t distribution converges to the Gaussian when the degrees of freedom goes to infinity, otherwise it is a heavy tailed distribution. Using this in RP matrices might come as a surprise to readers in the area of random projections since random projection theory requires sub-Gaussian entries in the random projection matrix for distance preservation purposes, but for reasons which will be explained shortly, our use of t distributed entries will allow us to further control the \( d \times d \) covariance \( \Sigma_{rp} \) in algorithm 2 and heavy tailed \( R_i \)'s will make it expand while sub-Gaussians would cause it to shrink.

The modified Step 6 of Alg. 2 that generates samples from t distribution.

We generate \( M \) independent random matrices with entries \( i.i.d \) from a t distribution with mean 0 and variance \( \frac{1}{2} \). By taking the variance to be \( \frac{1}{2} \) we make sure we recover the original scale in Step (8) without having to modify the scaling factor.\(^2\)

Now we want to see the effect of the random \( R_i \)'s on \( \Sigma_{rp} \). To this end, we condition on \( \Sigma \), and look at the expectation \( E_{\Sigma} [\Sigma_{rp}] \). We will use the following result, which computes expectations of this form:

**Lemma 1:** (Kaban, 2014): Let \( R \) be a \( k \times d \) random matrix, \( k < d \), with entries drawn i.i.d. from a symmetric distribution with 0-mean and finite first four moments. Let \( \Sigma \) be a \( d \times d \) fixed positive semi-definite matrix with eigenvalues \( \lambda_1, ..., \lambda_d \). Then, \( E[\Sigma R \Sigma R^T] = ... k \cdot E[R_{i,j}^2] \left( (k+1) + \sum_j \lambda_j \right) + \sum_j \lambda_j A_j \), where \( A_j \) are \( d \times d \) diagonal matrices with their \( j \)-th diagonal elements being \( \sum_{a=1}^{d} U_{a,i}^2 U_{a,j}^2 \) and \( U_{a,i} \) is the \( a \)-th entry of the \( i \)-th eigenvector of \( \Sigma \).

The expectation of \( \Sigma_{rp} \) is \( \frac{d}{k} E[\Sigma R \Sigma R^T] = \frac{d}{k} E[R_i^2 R_s^T R_i^T R_s] \). Since we have the entries of our t-distribution with mean 0 and variance \( \frac{1}{2} \), then we will have \( E[R_i^2] = \frac{1}{2} \). Furthermore, we see the excess kurtosis of the entries of \( R \) features in this result.

**Definition:** The excess kurtosis of a random variable \( x \) is defined as:

\[
K = \frac{E[x^4]}{E[x^2]^2} - 3
\]

**Proposition 2:** The excess kurtosis of a standardised \( t(0, 1, \nu) \) distribution with degree of freedom \( \nu \), is:

\[
K = \frac{6}{\nu - 4}, \quad \nu > 4
\]

\(^2\)This is done by sampling from the standardised \( t \), i.e. \( t(0, 1, \nu) \), and multiplying the samples by \( \sqrt{\frac{2}{\nu - 2}} \).

\(^3\)When \( d \) is high, \( R_i \) have a nearly orthonormal rows if the entries have variance \( 1/d \). So, pre-multiplying with \( R_i \) is like orthogonally projecting the points from the \( d \) dimensional space to a \( k \) dimensional subspace, which shortens the lengths of vectors by a factor of \( \sqrt{\frac{k}{d}} \) and the standard deviation get reduced as well by \( \sqrt{M} \) factor [12] after a simple averaging. Therefore, the scaling factor needed to ensure this recovery is \( \sqrt{\frac{M}{k}} \).
and it is easy to show that changing the variance leaves the excess kurtosis unchanged.

So, replacing this into Proposition 2, and noting that we can simplify the last term using that \( \sum_{i=1}^{d} \lambda_i A_i \leq \text{Tr}(\Sigma) \cdot I_d \), we obtain the result:

\[
\frac{d}{k} E[R^T R \Sigma R^T R] \geq \frac{1}{d} ((k+1) + \text{Tr}(\Sigma)(1 + \frac{6}{\nu - 4}) I_d) \tag{8}
\]

Let us point out what we have gained. In previous works such as [13], Gaussian RP matrices were used, and the Gaussian corresponds to \( \nu = \infty \) – therefore the last term in equation 8 becomes zero. Hence, by our choice of a lower degree of freedom for the t- distribution, we will be adding more regularity to the covariance which also makes it larger and gives it more chance to explore the search space better. Existence of the matrix expectation we just computed requires that \( \nu \) is at least 5. Therefore, 5, 6 and 7 may be good choices for \( \nu \) to investigate.

V. Experimental Studies

The goal of our first set of experiments is to find out whether the heavy tailed Cauchy search distribution is preferable in high dimensional EDA optimisation. So far there are only partial answers to this question. The second part is devoted to testing our idea of heavy tailed entries in the random matrices, and finally we put the results in context by comparing with existing state of the art.

A. Benchmark functions used

We use two sets of benchmark functions: the 1000-dimensional CEC’2010 test suite as described in [2], and the 50-dimensional CEC’05 test suite [20]. All problems are minimizations. The majority of the test functions implemented from the CEC’2010 benchmarks contain modifications to make them non-separable, and hence harder for meta-heuristics optimisation.

In the CEC’2005 problems, 5 are unimodal and 11 multimodal. All the global optima are within the given box constrains. However, problem 7 was without a search range and with the global optimum outside of the specified initialization range. From the problems in the [2] suite we are most interested in the multimodal ones.

In all the experiments, unless otherwise noted, 25 independent runs with a maximum budget of \( 3 \cdot 10^6 \) function evaluations are used with the 1000 dimensional problems and \( 6 \cdot 10^5 \) function evaluations with the 50 dimensional problems. Table I lists the 1000-dimensional CEC’10 problems that we used, and Table II gives the 50-dimensional ones.

B. Results

1) Cauchy vs. Gaussian in Univariate Marginal Distribution Algorithms: Before we delve into experiments that involve our multivariate Cauchy EDA, we will first check in the univariate model setting if the Cauchy search distribution indeed tends to work better than the Gaussian in high dimensions. To this end we will compare UMDA-Cauchy with the Gaussian UMDA. We created two variants of UMDA-Cauchy: a vanilla version, and a mixed version. The algorithm latter method is biased to sample new individuals using the Cauchy distribution at early stages of a run and will favour the Gaussian distribution at the later stages of a run. A function \( T(t) \) is defined to be decreasing with time from 1 to 0. \( T(t) \) is defined as \( T(t) = \frac{\text{currentFES}}{\text{maxFES}} \), making it slowly approach one towards the end of the run. If a uniform random number \( U(0, 1) \) is greater than \( T(t) \), i.e \( U(0, 1) > T(t) \), we apply Cauchy mutation and otherwise Gaussian mutation. The reasoning is the following: At the beginning of the run, the population should explore the search space, thus we make use of the larger step size of Cauchy mutation. Towards the end, it should exploit the most promising section of the search space and, thus, a smaller step size (Gaussian) is more suitable. UMDA-Cauchy also reduces the scaling parameter of the distribution linearly with time. We experimented with different population sizes as in [16], and the results reported are from experiments on 1000-dimensional test functions to a maximum number of fitness evaluations of 3 millions, and the population size of 1000 individuals with 75% elitism. These numbers was based on empirical results. Table III summarises our results from 25 independent repetitions. Bold font indicates a statistically significant outperformance as determined using T-Tests at the 95% confidence level. The comparisons were done between the standard UMDA-Cauchy variant using a Gaussian distribution and the winning implementation.
of the Cauchy variants.

We see that Cauchy’s performances on problems with a small domain, e.g. [−5, 5]^{1000} are low. This is because Cauchy explores too much and focus very small. The Gaussian distribution does pretty well in such domains, making it suitable for covering a small search space. The Gaussian variant performs significantly less on problems with a large search spaces such as [−100; 100]^{1000}. The larger step size induced by the Cauchy distribution is able to cover a larger area better. We also see that UMDAc with mixed distribution wins on several functions.

In conclusion, we can say that using Cauchy distribution as an alternative search distribution improves on the performance of UMDAc on most benchmark problems. We should also observe though that from the practical point of view the differences are not very large.

2) Cauchy vs. Gaussian in the Multivariate Model Setting:

As already mentioned, estimating a multivariate model in high dimensions is difficult. We will use the 50-dimensional test functions first and compare the Multivariate Cauchy random distribution to Gaussian random selection EDA (MCrpEDA) presented in Section III with the originally proposed Multivariate Gaussian random projection EDA (MGrpEDA) [13], that is, \( k = 3 \) and we have set \( M = 4d/k \). We used the same parameter settings as in [13], but varied the population size as it could impact on the outcome. We tested population sizes of 20, 50, 100, 500, and 1000. For each population size tested, the percentage of individuals retained at the truncation selection stage is \( \tau = 30\% \) of \( N \) – this ratio is frequently used in the literature. We did 25 independent runs for each problem on a fixed budget of 600,000 function evaluations in each case. As before, the performance criterion was the difference of the best fitness found and the known optimal fitness at the global optimum.

For space constraints we show only the results from 50 and 1000 population sizes, summarised in Tables IV-V with bold font indicating statistically significant outperformance at the 95% confidence level.

We observed, rather surprisingly, that in small population size conditions, up to the experiments with population size of 100, the Gaussian gains the upper hand. The performance is about equal when the population size is as large as 500 or 1000 although the two methods win on different functions. This is very different from what has been observed in 2D in [22] where the multivariate Cauchy EDA was better than the Gaussian in small population conditions. We then inspected the fitness trajectories to gain insight into the reasons that the multivariate Cauchy search distribution no longer presents the advantages it did in 2D. We observed that the Cauchy trajectories display a much slower convergence. So it seems that in high dimensions, the search space being so huge, the multivariate Cauchy samples explore too much and focus too little. If we let the algorithm run for many more generations, we often observe the performance of Cauchy-RP-EDA eventually getting better than that of its Gaussian counterpart that converged much quicker. However the search costs until this happens increase with the dimension and the multivariate Cauchy therefore loses its attractiveness. Figure 2 shows examples of trajectories for the P03, P06 and P16 functions respectively when the population size is 500, 50 and 300 with different maximum generations. MGrpEDA converges faster than MCrpEDA at the beginning, but by the end of the budget MCrpEDA manages to catch up with MGrpEDA and later outperformed it. See figure 2 (Top left), Taking a closer look towards the end of the budget in figure 2 (Top right), MCrpEDA is still progressing while the MGrpEDA has already achieved its potential. Now when we allow more function evaluations, MCrpEDA has a better chance to win, see figure 2 (Top right) where MCrpEDA outperformed MGrpEDA when the number of generation was increased to 1000. MCrpEDA caught up with MGrpEDA after the 450th generations, and wins afterwards. MGrpEDA was winning from the beginning until at the 450th generation when it is already at its optimum, when MCrpEDA overtook it. Due to page limit constrain, we cannot show all the plots where this behaviour happened, but the scenario occurred 30 times out of all the 95 runs. But there are also examples where it is clear that MGrpEDA performs better. See figure 2 for P16 of 100 dimensions (bottom left) and P06 of 1000 dimensions (bottom right).

We have ran further experiments on the 1000-dimensional functions, repeating the previous comparison, and the results confirmed the same conclusion. So far we have the univariate-Cauchy was better than univariate Gaussian but the multivariate comparison concluded the opposite. In view of this it is now extremely interesting to see how does the univariate and the multivariate Cauchy compare to each other. We have decided to fix the population size to 300 since we have seen this parameters has had rather little influence in the conclusions in the previous section. We set \( k = 3 \) as before. Figure 3 shows the average of the best fitness comparatively
for the univariate-Cauchy, the multivariate-RP-Cauchy the multivariate-RP-Gaussian, the CMA-ES and we also included our Multi-RP-Gaussian with t-entries as described in Section IV. We chose the degrees of freedom to be 7, which gives us a heavy tailed behaviour.

Focussing on the first three of these methods for now, we can see two important findings: (1) The multivariate-RP-Cauchy EDA outperforms the univariate-Cauchy-EDA in nearly all functions. (2) The multivariate-RP-Cauchy EDA is outperformed by the multivariate-RP-Gaussian EDA in most of the functions.

The first of these findings is reassuring. It shows that despite in the univariate model setting the Cauchy search distribution performed better than the Gaussian, the multivariate Cauchy wins over among these. However we also see this does not imply that MCrpEDA is a good choice, since it is outperformed by MGpEDA. MGpEDA also outperformed the famous CMA-ES in most of the benchmark functions, and our modified version with t-entries displays a further improved performance.

3) Comparison with state of the art methods: Finally, we provide some context of our results by comparing with several state of the art approaches that we reviewed in an earlier section. The experiments were run on the 1000-dimensional functions, and are summarised in Tables VII and VI. We present the average and std of best fitness results\(^2\) at two different points throughout the EDA process: FESI: Results after 0.6 million function evaluations. FESII: Results after 1.2 million function evaluations Table VI indicates that at early stages the Gaussian-Multi-RP wins since it managed to outperform the rest of the methods with statistical significance in two of the functions. Gaussian-Multi-RP-t(df=7) and CCVIL also won in one of the functions.

For results after 1.2 million function evaluation(FESII), it is clear that Gaussian-Multi-RP-t(df=7) performed significantly better in more functions than the rest were able to do. It outperformed all the others in two of the functions with statistical significance, and we can also see by inspection that Gaussian-Multi-RP-t(df=7) is competitive in the rest of the functions where no statistically significant performances were not registered. Once three million function evaluations are consumed (omitted for space constraints) we observed that CCVIL became the best performer among all the methods considered on F2, F11 and F16, while CMA-ES was the best on F13, F18 and F20. Gaussian-Multi-RP-t(df=7) was the best on only one which is F3. While there were no significant difference among the methods in the rest of the functions, Gaussian-Multi-RP competed well with Gaussian-Multi-RP-t(df=7) on quite a good number of them, by inspection.

Hence, based on these preliminary results and analysis we can conclude that Gaussian-Multi-RP, and especially our variant with t-entries excel at the relatively early stages of the optimisation process – which is a great practical advantage.

\(^2\)Computed from 25 independent runs for most methods, 5 runs for Gaussian-Multi-RP and Cauchy-Multi-RP EDAs, 3 runs for Gaussian-Multi-RP-t(df=7) – as these are still work in progress.
Given a large budget then CCVIL wins over, nevertheless our mentioned methods still remain competitive. Since these are early results for Gaussian-Multi-RP with t-entries, and these results looks indeed promising, our future work is going to investigate this in more detail.

**REFERENCES**


