Heavy tails with Parameter Adaptation in Random Projection based continuous EDA

Momodou L. Sanyang*† and Ata Kabán*
*School of Computer Science
University of Birmingham, Edgbaston, UK, B15 2TT
{M.L.Sanyang, A.Kaban}@cs.bham.ac.uk
†School of Information Technology and Communication
University of The Gambia, Brikama Campus, P.O. Box 3530, Serekunda, The Gambia
mlsanyang@utg.edu.gm

***Abstract*—In this paper, we present a new variant of EDA for high dimensional continuous optimisation, which extends a recently proposed random projections (RP) ensemble based approach by employing heavy tailed random matrices. In particular, we use random matrices with i.i.d. t-distributed entries. The use of t-distributions may look surprising in the context of random projections, however we show that the resulting ensemble covariance is enlarged when the degree of freedom parameter is lowered. Based on this observation, we develop an adaptive scheme to adjust this parameter during evolution, and this results in a flexible means of balancing exploration and exploitation of the search process. A comprehensive set of experiments on high dimensional benchmark functions demonstrate the usefulness of our approach.**

I. Introduction

Optimisation is at the core of many scientific, engineering and management problems and has applications in many disciplines. Modern applications often require optimisation over large problems. Estimation of Distribution Algorithms (EDAs) are a relatively recent branch of stochastic optimization heuristics, which guide the search to the optimum by repeatedly building and sampling explicit probability models of the fittest individuals from the current population [15]. This allows the algorithm to learn the structure of the search space and to guide the search in further promising directions [25]. We can say they are an enhancement of regular EAs. The core of each EDA variant is the process of building its model. The probabilistic model is often the only device used for guiding the search to the global optimum [19]. This being the case, the model must represent the search space in great detail, but the required search cost and computational resources have to be appropriate as well. This becomes an issue when the search space is high dimensional, and therefore simplified models such as univariate models are frequently used instead of a full multivariate model. The simplest such approach is described in [16], called $UMDA_c$.

For non-separable problems, the design variables have inter-dependencies. The approach known as MIMIC, proposed by *De Bonet et. al* in [3] is set out to capture bivariate interactions between decision variables by sampling from the pairwise joint distribution between variables. Despite MIMIC is able to outperform univariate models, the majority of optimization problems will have larger groups of interacting design variables. Several proposals have been put forth to explicitly capture multivariate dependencies by building graphical dependency networks, for example Bayesian Networks [4]. But, from statistical, computational and memory points of view, learning probabilistic graphical models is highly expensive [2]. Thus, the scaling up of these model building processes to high dimensional problems is challenging. Efforts to alleviate 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 [22] were among such proposals.

Beyond the above simple methods, model building in high dimensions is the subject of many research efforts. Many approaches were proposed, here we will limit ourselves to a few most relevant ones. Eigendecomposition EDA ($ED-EDA$) [7] proposes to utilise a repaired version of the full covariance matrix estimate, thus giving it the ability to capture interaction among decision variables. Other methods use limited dependencies. For example, Cooperative Co-evolution with Variable Interaction Learning ($CCVIL$) proposed by *Weicker et al.* in [22] is a deterministic method to uncover dependencies between decision variables, which has later been extended to the CCVIL framework by *Chen et al* in [5]. In contrast to the other algorithms mentioned, $CCVIL$ tries to explicitly find variable interaction. EDA with Model Complexity Control ($EDA-MCC$) [6] also employs a deterministic algorithm to group variables. It 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. Otherwise, they are considered to be strongly dependent. Only the later dependencies are modelled.

Covariance Matrix Adaptation *(CMA-ES)* [9], is an evolutionary strategy technique, considered widely to be one of the leading optimization techniques. 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 scale problems (e.g. [10], [13]), but benchmarks on large scale problems are rather rare. A variant of *(CMA-ES)* called separable CMA-ES (*sep-CMA-ES*) [18] 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 [18]. The last category is the divide and conquer. Algorithms in this category are Multilevel Cooperate Co-evolution ($MLCC$), proposed in [24]. This is a framework that groups the decision variables of a problem randomly in order to put interacting variables in one subcomponent to form a group to tackle problems in high-dimensional optimization. The groups are optimized jointly but separately from other groups. Another divide and conquer technique is the recently proposed random projections (RP) EDA. This is an approach which is state of the art and was proposed in [12]. It 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 estimation and sampling job is done in the low dimensional space instead of 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 [12].

In this paper, we develop an approach which employs a heavy tailed distribution as a means to enhance a recently proposed random projections (RP) ensemble based approach [12] to increase exploration while maintaining exploitation and focus by using multivariate Gaussian EDA in which the entries of the RP matrices are drawn from an i.i.d. heavy tailed distribution instead of the commonly used Gaussian or sub-Gaussian entries.

II. Using heavy tails in Random Projection based continuous EDA

In this section we devise a new variant of the RP-based large scale multivariate Gaussian EDA of [12] by proposing to use random matrices with heavy tailed entries. To readers familiar with the area of random projections, this might come as a surprise since random projection theory requires sub-Gaussian matrices, but our reasons will become clear in the Analysis subsection shortly. In essence, as we shall see, the use of i.i.d. t-distributions, specifically its degree of freedom parameter, will allow us to enlarge the high dimensional ensemble-covariance matrix of the search distribution, which may facilitate exploration and escape early convergence, while still maintaining the focus of the search. Our analysis implies also that sub-Gaussian random matrices in this context would cause the ensemble covariance to shrink, thus making the algorithm more prone to pre-mature convergence.

1) Algorithm presentation: We build on random projection ensemble based EDA (RP-Ens-EDA) [12], and refer to our new variant as tRP-Ens-EDA. The pseudocode of tRP-Ens-EDA is shown in algorithm 1.

tRP-Ens-EDA proceeds by initially generating a population of individuals randomly everywhere and selects the $\tilde{N}$ fittest points based on their fitness values. This is the set P^{fit} in algorithm 1. 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 $k \ll d$, where d is the dimension of the search space, and k is also a parameter of the method. For both of

Algorithm 1 Pseudocode of the Multivariate Gaussian random projection EDA with R entries from t-distribution (tRP-Ens-EDA)

```
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 the fittest $\tilde{N}$ individuals $P^{fit}$ from $P$
    (5) Estimate $\mu := mean(P^{fit})$
    (6) Generate $M$ independent random matrices with entries iid from a t distribution with mean 0 and variance $\frac{1}{d}$.
    (7) For $i = 1, ..., M$.
        (a) Project the centred points into k-dimensions:
            $\mathbf{Y}^{R_i} := [R_i(x_n - \mu); n = 1, ..., \tilde{N}]$.
        (b) Estimate the $k$ x $k$ sample covariance $\Sigma^{R_i}$.
        (c) Sample $N$ new points $y_1^{R_i}, ..., y_N^{R_i} \sim_{i.i.d} N(0, \Sigma^{R_i})$.
    EndFor
    (8) Let the new population $P^{new} := \sqrt{\frac{dM}{k}}[\frac{1}{M}\Sigma_{i=1}^{M} R_i^T y_1^{R_i}, ..., \frac{1}{M}\Sigma_{i=1}^{M} R_i^T y_N^{R_i}] + \mu$.
    (9) $P \leftarrow P^{new}$
Until Termination criteria are met or MaxFE exceeded
Output $P$
```

these parameters we will use the default values as in [12]. 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. The entries of the RP matrices are drawn iid from a t distribution with mean 0 and variance $\frac{1}{d}$. This is done by sampling from $t(0, 1, \nu)$ standard t, and then multiply the samples by $\sqrt{\frac{\nu-2}{\nu d}}$. The reason we take the variance to be $\frac{1}{d}$ is to make sure we recover the original scale in Step (8) without having to modify the scaling factor: When d is high, R_i have 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 gets reduced by a factor of $\sqrt{M}$ after averaging [12]. Therefore, the scaling factor needed to ensure this recovery is $\sqrt{\frac{dM}{k}}$.

Step 7(a) projects the good samples 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 the new population P.

2) Analysis: We start by computing the ensemble-covariance of the new population in step (8) conditional on fixing the random projection matrices $R_i, i = 1 : M$. We will then condition on the fit individuals and look at the effect of $R_i, i = 1 : M$ by computing the expectation of this ensemble covariance with respect to $R_i, i = 1 : M$.

Proposition 1: Conditionally on all R_i, $i = 1...M$, the new generation produced at Step 8 of Algorithm 1 is i.i.d. Gaussian with mean μ 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 Σ is the sample covariance of the original selected individuals in P^{fit}

Proof: Recall from step 7(a) of Algorithm 1 that the set of projected points in the i-th subspace is:

$$\mathbf{Y}^{R_i} = \{R_i(x_1 - \mu), R_i(x_2 - \mu), ..., R_i(x_{\tilde{N}} - \mu)\}$$

Conditional on R_i, the sample covariance matrix of this set of points is:

$$\Sigma^{R_i} = \frac{1}{\tilde{N}} \sum_{n=1}^{\tilde{N}} R_i(x_n - \mu)(R_i(x_n - \mu))^T = R_i \Sigma R_i^T$$

So the samples in step 7(c) of algorithm 1 are $y_1^{R_i}, ..., y_{\tilde{N}}^{R_i} \sim N(0, \Sigma^{R_i})$.

To find the distribution of the individuals in P at step (8) of Algorithm 1, we look at the first individual:

$$P_1 := \sqrt{\frac{dM}{k}} [\frac{1}{M} \sum_{i=1}^{M} R_i^T y_1^{R_i}] + \mu. \tag{1}$$

Conditionally on $R_i, i = 1 : M$, this is a linear combination of independent Gaussian random variables , which is again a Gaussian[1] [17]. Hence, P_1 is Gaussian distributed, it has mean μ (since $y_1^{R_i}$ has zero mean), and we compute its covariance below.

In equation (1), denote $A := \sqrt{\frac{dM}{k}} \frac{1}{M} R_i^T$, then from step 7(c), we have that $y_1^{R_i} \sim N(0, R_i \Sigma R_i^T)$. So,

$$A y_1^{R_i} \sim N(0, A R_i \Sigma R_i^T A^T) \tag{2}$$

Replacing A in (2), we have

$$A y_1^{R_i} \sim N(0, \sqrt{\frac{dM}{k}} \frac{1}{M} R_i^T R_i \Sigma R_i^T \sqrt{\frac{dM}{k}} \frac{1}{M} R_i),$$

which simplifies to

$$N(0, \frac{d}{kM} R_i^T R_i \Sigma R_i^T R_i)$$

Repeating this reasoning for each P_j, we find that:

$$P_j \sim N(\mu, \frac{d}{kM} \sum_{i=1}^{M} R_i^T R_i \Sigma R_i^T R_i), \ \ j = 1, ..., N \tag{3}$$

[1]Assume $x \sim N(m_x, \Sigma_x)$ and $y \sim N(m_y, \Sigma_y)$, then

$$Ax + By + c \sim N(Am_x + Bm_y + c, A\Sigma_x A^T + B\Sigma_y B^T)$$

.

Hence, the form of the ensemble covariance in the d-dimensional search space is:

$$\Sigma_{rp} = \frac{d}{kM} \sum_{i=1}^{M} R_i^T R_i \Sigma R_i^T R_i \tag{4}$$

■

Now we want to see the effect of the random R_i's on Σ_{rp}. To this end, we condition on Σ, and look at the expectation $E_R[\Sigma_{rp}]$. For this we use the following result:

Lemma 1: (Kaban, 2014 [11]): 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 Σ be a $d \times d$ fixed positive semi-definite matrix with eigenvalues $\lambda_1, ..., \lambda_d$. Then, $E[R^T R \Sigma R^T R] = ...$

$$k \cdot E[R_{i,j}^2]^2 [(k+1)\Sigma + Tr(\Sigma) I_d + (\frac{E[R_{i,j}^4]}{E[R_{i,j}^2]^2} - 3) \sum_{d}^{i=1} \lambda_i A_i]$$

where A_i are $d \times d$ diagonal matrices with their jth diagonal elements being $\Sigma_{a=1}^{d} U_{ai}^2 U_{aj}^2$ and $U_{ai}$is the a-th entry of the ith eigenvector of Σ.

From (4), the expectation of Σ_{rp}, is

$$E[\frac{d}{k} R^T R \Sigma R^T R] = \frac{d}{k} E[R^T R \Sigma R^T R].$$

which we will compute. Since we have the entries of our t-distribution with mean 0 and variance $\frac{1}{d}$, then we will have $E[R_{ij}^2] = \frac{1}{d}$. Furthermore, we see the excess kurtosis of the entries of R featuring in this result. So we need to compute this for the t-distribution. This excess kurtosis will contain the degree of freedom parameter of the t distribution which we shall vary adaptively to control the exploration and exploitation and focus of our algorithm.

Definition: The excess kurtosis of a random variable x is defined as:

$$K = \frac{E[x^4]}{E[x^2]^2} - 3 \tag{5}$$

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

$$K = \frac{6}{\nu - 4}, \ \ \nu > 4 \tag{6}$$

Proof: It is relatively straight-forward to derive the form of the fourth moment, $E[x^4]$ and the second moment, $E[x^2]$.

Let $x \in R^n$ be a random variable, $k \in \{2, 4\}$ and $x \sim t(0, 1, \nu)$, then by definition

$$E[x^k] = \int_{-\infty}^{\infty} x^k f(x) dx$$

where $f(x)$ is the pdf of the t distribution and we only need to evaluate for $k \in \{2, 4\}$, since these are needed in the definition of excess kurtosis. The pdf of a t distribution is written as:

$$f(x) = c(1 + \frac{x^2}{\nu})^{\frac{-1}{2}(\nu+1)} \tag{7}$$

where

$$c = \frac{1}{\sqrt{\nu}B(\frac{\nu}{2},\frac{1}{2})} \quad (8)$$

and $B(\cdot,\cdot)$ is the beta function [1]. It can be observed that $f(x) = f(-x)$, since t is a symmetric distribution. We can re-write $E[x^k]$ as:

$$= \int_{-\infty}^{0} x^k f(x)dx + \int_{0}^{\infty} x^k f(x)dx$$

Now computing the fourth moment, $E[x^4]$ and applying a change of variable in the first integral by letting $t = -x$, then $dx = -dt$, we will have

$$E[x^4] = -\int_{\infty}^{0} (t)^4 f(-t)dt + \int_{0}^{\infty} x^4 f(x)dx$$

Interchanging the bounds of the integral, applying the symmetric distribution property and then throwing common factor will give us the following

$$E[x^4] = 2\int_{0}^{\infty} x^4 f(x)dx \quad (9)$$

Plugging in the form of f as defined in eq. 7, we have that eq. 9 equals

$$= 2c\int_{0}^{\infty} (x)^4 (1+\frac{x^2}{\nu})^{\frac{-1}{2}(\nu+1)}dx$$

We make the following change of variable: $t = \frac{x^2}{\nu}$, $x = (\nu t)^{\frac{1}{2}}$, and with some algebra, we have

$$= c\nu^{\frac{4+1}{2}} \int_{0}^{\infty} (t)^{\frac{4+1}{2}-1}(1+t)^{-(\frac{4+1}{2})-(\frac{\nu-4}{2})}dt$$

This integral represents a beta function [1]. Therefore we get

$$E[x^4] = c\nu^{\frac{4+1}{2}} B(\frac{4+1}{2}, \frac{\nu-4}{2})$$

Now plugging in the value of c back, we have from eq. 8

$$E[x^4] = \nu^2 \frac{\Gamma(\frac{5}{2})\Gamma(\frac{\nu-4}{2})}{\Gamma(\frac{\nu}{2})\Gamma(\frac{1}{2})}, \quad \frac{\nu-4}{2} > 0 \ \ since \ \ \Gamma(0) = \infty.$$

But $\Gamma(\frac{5}{2}) = \frac{3}{4}\sqrt{\pi}$ [1] and $\Gamma(\frac{1}{2}) = \sqrt{\pi}$ [1],

$$= \frac{\frac{3\nu^2}{4}\ \Gamma(\frac{\nu-4}{2})}{\Gamma(\frac{\nu}{2})}$$

By the definition of Gamma $\Gamma(x) = (x-1)\Gamma(x-1)$ [1], we have after substitution and simplification

$$E[x^4] = \frac{3\nu^2}{(\nu-2)(\nu-4)} \quad (10)$$

Analogously, computing the second moment, $E[x^2]$, we will arrive at

$$E[x^2] = \frac{\nu}{\nu-2}$$

Therefore, the excess kurtosis is

$$K = \frac{E[x^4]}{(E[x^2])^2} - 3 = \frac{\frac{3\nu^2}{(\nu-2)(\nu-4)}}{\frac{\nu^2}{(\nu-2)^2}} - 3$$

$$K = \frac{3(\nu-2)}{\nu-4} - 3 = \frac{6}{\nu-4}$$

■

Corollary: The excess kurtosis of a distribution with variance σ^2 remains unchanged.

Proof: . Let $c > 0$ be a constant. Then $c \cdot x$ has variance $c^2 var(x)$. Now show that the excess kurtosis of $c \cdot x$ is still

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

If $c > 0$, then the excess kurtosis of $c \cdot x$ is

$$\frac{E[(c\cdot x)^4]}{(E[(c\cdot x)^2]}^2 - 3$$

taking the constant out, we have

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

thus the excess kurtosis did not change. ■

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

$$\frac{d}{k}E[R^T R\Sigma R^T R] \preceq \frac{1}{d}[(k+1)\Sigma + Tr(\Sigma)(1+\frac{6}{\nu-4})I_d] \quad (11)$$

Now let us point out what we have gained. In previous works such as [12], Gaussian was used and Gaussian corresponds to ν being ∞, therefore the last term in eq. (11) tends to zero, so by our choice of degree of freedom that the t- distribution has, 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 ν is at least 5.

III. SETTING AND ADAPTATION OF THE DEGREE OF FREEDOM ν

Parameter setting methods are dichotomised into tuning and controlling [8]. Tuning means finding a good value via trial and error before running the algorithm and then fixing this value throughout the evolutionary process. On the other hand, parameter control starts with an initial value which is changed during the run, based on the feedback from the algorithm [8]. So the latter tries to adapt the control parameters automatically to adjust the algorithm to the problem while solving it during the search [23].

In our algorithm, the parameter we try to control is the degree of freedom of the t distributed entries of our random projection matrices ($R_i, i = 1 : M$). We drew our inspiration from [23] and carved out our own adaptive method with our own rules to vary the value of the degree of freedom automatically to fit our problem. The Pseudocode of our proposed adaptive method is shown in algorithms 2, 3a, and 3b.

Algorithm 2 takes degree of freedom *df* which is an array of different values of the degree of freedom to be tried as inputs. It runs these values concurrently during a generation

Algorithm 2 Adaptive degree of freedom (df) Algorithm

```
**Inputs**: $df$: array of df values tried, $L = lenght(df)$ .
(1) **for** $i = 1 : L$
(2)     $dftry := df[i]$;
(3)      run steps (6),(7),(8) and (3) of Algorithm 1;
(4)     f[i] := min fitness from step (3) of algorithm 1
(5) **endfor**
(6) $[fminfminInd] := min(f)$;
(9) $dfbest := df[fminInd]$;
(10) **if** $min(f) == max(f)$
(11)          $df[1] := round(df[1]/2)$;
(12)          **for** $i = 2 : L$
(13)          $df[i] == df[i] * 2$;
(14)          **endfor**
(16) **elseif**
(17)    UPDATEDF(2df) or UPDATEDF(5df);
(18) **endif**
**Output** $df$
```

Algorithm 3a UPDATEDF(2df)

```
(1) **procedure** UPDATEDF(2df)
(2) **if** $f[1] == min(f)$
(3)       $df1 := round(df1/2)$;
(4)       $df2 := round((df1 + df2)/2)$;
(5) **elseif** $f[2] == min(f)$
(6)       $df1 := round((df1 + df2)/2)$;
(7)       $df2 := df2 + round(df2/2)$;
(8) **endif**
(9) **endprocedure**
```

Algorithm 3b UPDATEDF(5df)

```
(1) **procedure** UPDATEDF(5df)
(2) **if** $f[1] == min(f)$
(3)       $df1 := round(df1/2)$;
(4)       $df2 := round((df1 + df2)/2)$;
(5)       $df3 := round((df2 + df3)/2)$;
(6)       $df4 := round((df3 + df4)/2)$;
(7)       $df5 := round((df4 + df5)/2)$;
(8) **elseif** $f[2] == min(f)$
(9)       $df1 := round((df1 + df2)/2)$;
(10)      $df2 := df2 + round(df2/2)$;
(11)      $df3 := round((df2 + df3)/2)$;
(12)      $df4 := round((df3 + df4)/2)$;
(13)      $df5 := round((df4 + df5)/2)$;
(14) **elseif** $f[3] == min(f)$
(15)      $df1 := round((df1 + df2)/2)$;
(16)      $df2 := round((df2 + df3)/2)$;
(17)      $df3 := df3 + round(df3/2)$;
(18)      $df4 := round((df3 + df4)/2)$;
(19)      $df5 := round((df4 + df5)/2)$;
(20) **elseif** $f[4] == min(f)$
(21)      $df1 := round((df1 + df2)/2)$;
(22)      $df2 := round((df2 + df3)/2)$;
(23)      $df3 := round((df3 + df4)/2)$;
(24)      $df4 := df4 + round(df4/2)$;
(25)      $df5 := round((df4 + df5)/2)$;
(26) **elseif** $f[5] == min(f)$
(27)      $df1 := round((df1 + df2)/2)$;
(28)      $df2 := round((df2 + df3)/2)$;
(29)      $df3 := round((df3 + df4)/2)$;
(30)      $df4 := round((df4 + df5)/2)$;
(31)      $df5 := df5 + round(df5/2)$;
(32) **endif**
(33) **endprocedure**
```

and keeps track of the best fitnesses found with each of these parameter values. These fitnesses of each of the df tried are compared to see which one is the best and then choose the degree of freedom that gives the best fitness to be taken forward. The contents of df are then updated accordingly, in a way that places the values that are to be tried next time around the best performing one. If all values tried performed the best then we spread out the content of df. The rules of how this is done are given in algorithms 3a and 3b, where we use 2 or 5 different values concurrently respectively. This process is then repeated in each generation.

The number of degree of freedom parameters that are tried concurrently at each generation need not be exactly two or five, and the rules can easily be designed for different numbers if desired. The more degree of freedoms used concurrently, the better the results, but this comes at the expense of more function evaluations.

IV. Analysis of the experimental results

A. Benchmark functions used

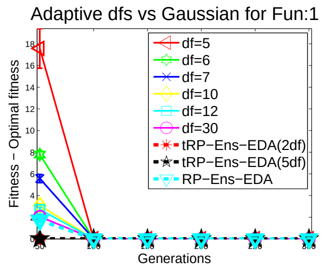
We use two sets of benchmark functions: the 1000-dimensional CEC'2010 test suite as described in [21], and the 50-dimensional CEC'2005 test suite [20]. All problems are minimizations. The majority of the test functions implemented from the CEC'2010 benchmarks contain modifications to make them non-separable, and hence harder for meta-heuristics optimisation.

TABLE I: 1000-dimensional test functions from the CEC'10 collection.

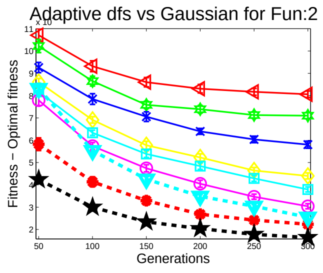
Problem	Name
F2	Shifted Rastrigin's function
F3	Shifted Ackley's function
F5	Single-group Shifted and m-rotated Rastrigin's function
F6	Single-group Shifted and m-rotated Ackley's function
F10	$\frac{D}{2m}$-group Shifted and m-rotated Rastrigin's function
F11	$\frac{D}{2m}$-group Shifted and m-rotated Ackley's function
F13	$\frac{D}{2m}$-group Shifted and m-dimensional Rosenbrock's function
F15	$\frac{D}{2m}$-group Shifted and m-rotated Rastrigin's function
F16	$\frac{D}{m}$-group Shifted and m-rotated Ackley's function
F18	$\frac{D}{m}$-group Shifted and m-dimensional Rosenbrock's function
F20	Fully nonseparable Rosenbrock

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 [21] suite we are most interested in the multimodal ones.

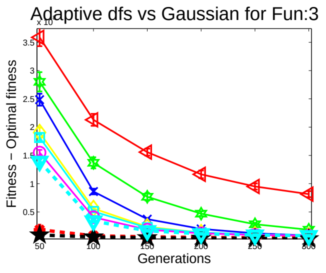
Table I lists the 1000-dimensional CEC'10 problems that we used, and Table II gives the 50-dimensional ones.



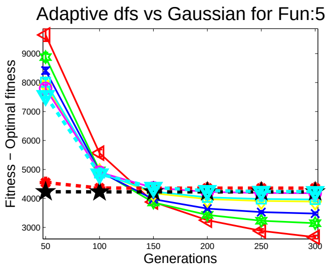
(a) Function 1



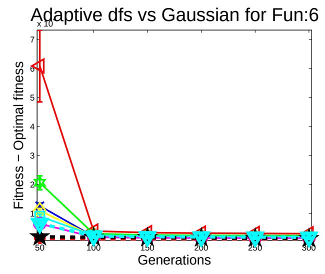
(b) Function 2



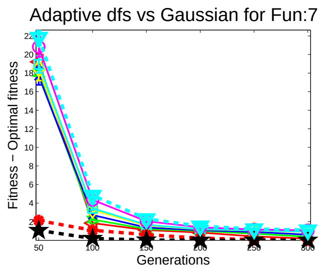
(c) Function 3



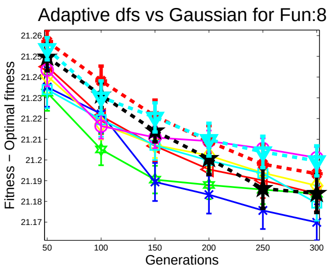
(d) Function 5



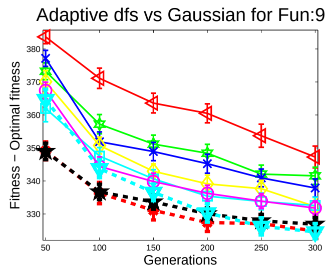
(e) Function 6



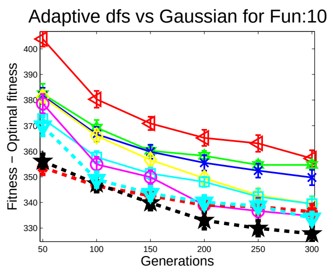
(f) Function 7



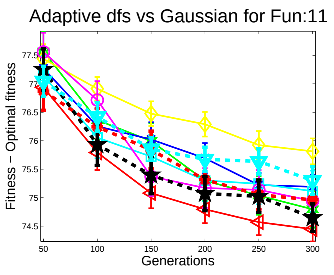
(g) Function 8



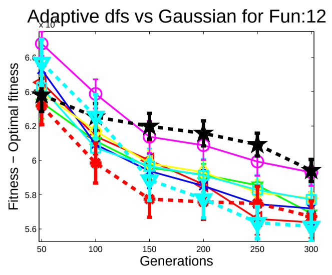
(h) Function 9



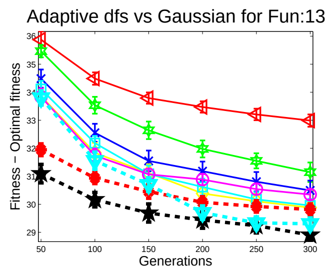
(i) Function 10



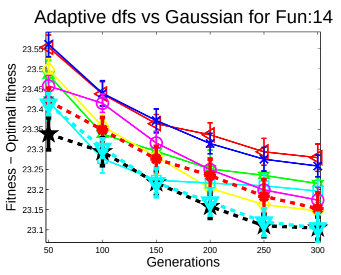
(j) Function 11



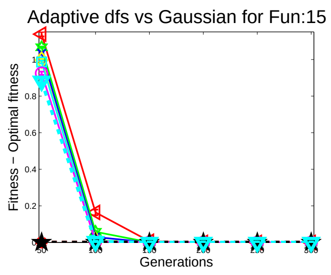
(k) Function 12



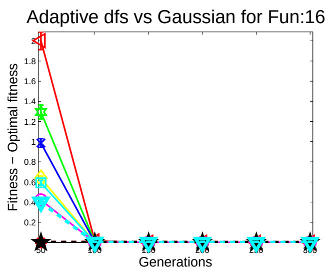
(l) Function 13



(m) Function 14



(n) Function 15



(o) Function 16

Fig. 1: Plots to compare different dfs, RP-Ens-EDA and two versions of our tRP-Ens-EDA on Functions 1-16. For better visibility, we display from generation 50 only and show legend of only the first plot. The error bars represent one standard error over 25 repeated runs.

B. *Experiments and Results*

We conducted two types of experiments. The first is with a fixed time frame of 300 generations in order to assess the potential of various values for the degree of freedom (df) as well as our adaptive procedures. The second type of experiment compares tuning with adaptation on a fixed budget of function evaluations, set to $5.4 \cdot 10^5$.

In all the experiments, 25 independent runs were performed except on the 1000 dimensional functions experiment, were due to time constraints, we were able to do only 10 independent runs of our method. The rest of the state of the art methods were given 25 independent runs.

1) RP-Ens-EDA vs. the proposed tRP-Ens-EDA in experiments with equal time frame: We use the 50 dimensional CEC 2005 benchmark functions. We ran the RP-Ens-EDA [12] and our proposed tRP-ENS-EDA with different degrees of freedom values for 300 generations each. The results are shown on figure 1.

As we can see from the results in figure 1, different degrees of freedom perform well on different problems. Therefore, we

TABLE IV: Comparison with state of the art under equal budget of $3 \cdot 10^6$ function evaluations.

	ED-EDA		CCVIL		MLCC		sep-CMA-ES		EDA-MCC		tRP-Ens-EDA	
	Mean	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**
F2	1.11E+04	1.12E+02	**4.00E-07**	**0.00E+00**	37.89	86.41	5.68E+03	4.89E+02	1.17E+04	6.38E+01	5.94E+02	1.76E+01
F3	7.22E-01	4.71E-02	4.00E-07	6.32E-07	8.45E-01	1.04E+00	2.15E+01	1.08E-01	2.92E+00	7.76E-02	**2.48E-13**	**4.57E-15**
F5	3.15E+08	1.33E+07	5.51E+08	1.55E+08	1.22E+08	8.63E+07	1.19E+08	2.92E+07	4.24E+08	1.67E+07	**4.99E+06**	**1.32E+06**
F6	3.56E+06	2.81E+05	4.14E+05	6.54E+05	1.06E+06	9.42E+05	6.39E+06	3.85E+06	1.53E+07	1.92E+05	3.20E+01	5.32E-01
F10	1.12E+04	1.00E+02	1.43E+03	6.34E+01	2.93E+03	6.72E+02	6.28E+03	2.51E+02	1.18E+04	6.35E+01	**5.96E+02**	**1.71E+01**
F11	1.49E+02	2.33E+01	**7.44E+00**	**2.41E+00**	1.64E+02	7.72E+00	2.12E+02	6.16E+00	1.91E+02	7.39E-01	3.10E+01	1.09E+01
F13	2.13E+06	2.25E+05	2.98E+11	8.57E+11	1.56E+03	5.52E+02	**2.94E+02**	**9.20E+01**	4.77E+10	2.99E+09	1.00E+06	5.74E+04
F15	1.13E+04	9.74E+01	2.78E+03	8.78E+01	7.11E+03	1.34E+03	6.76E+03	2.76E+02	1.18E+04	6.61E+01	**6.05E+02**	**2.74E+01**
F16	1.76E+02	2.35E+01	1.31E+01	2.92E+00	3.62E+02	7.80E+00	4.21E+02	1.59E+01	2.37E+02	2.43E+01	8.04E+01	1.04E+01
F18	1.02E+06	3.72E+05	6.42E+11	2.19E+12	3.36E+03	9.08E+02	**9.16E+02**	**1.94E+02**	1.49E+10	1.67E+09	4.41E+04	5.40E+03
F20	1.89E+05	9.81E+04	1.75E+11	8.77E+11	2.23E+03	3.20E+02	9.04E+02	3.91E+01	8.08E+08	7.81E+07	1.02E+03	2.83E+01

TABLE II: 50-dimensional test functions from the CEC'05 collection.

Problem	Name
P01	Shifted Sphere Function
P02	Shifted Schwefels Problem 1.2
P03	Shifted Rotated High Conditioned Elliptic Function
P05	Schwefel's Problem 2.6 with Global Optimum on Bounds
P06	Shifted Rosenbrock's Function
P07	Shifted Rotated Griewank's Function without Bounds
P08	Shifted Rotated Ackley's Function with Global Optimum on Bounds
P09	Shifted Rastrigin's Function
P10	Shifted Rotated Rastrigin's Function
P11	Shifted Rotated Weierstrass Function
P12	Schwefel's Problem 2.13
P13	Expanded Extended Griewank's plus Rosenbrock's Function
P14	Expanded Rotated Extended Scaffe's F6
P15	Shifted Griewank's Function
P16	Shifted Ackley's Function

TABLE III: Statistical Test for performance comparison between Tuning, 5df and 2df methods ran on equal budget.

Problems	**Tuning**	**2 concurrent df**	**5 concurrency df**
	(mean Optimal Gap)	**(mean Optimal Gap)**	**(mean Optimal Gap)**
P01	$0{\pm}0$	$0{\pm}0$	$0{\pm}0$
P02	4.07e+04±7.32e+03	1.53e+04±3.85e+03	1.62e+04±3.23e+03
P03	7.84e+06±1.74e+06	2.06e+06±3.82e+05	2.29e+06±4.87e+05
P04	**0±0**	2.17e+04±4.56e+03	2.05e+04±4.99e+03
P05	3.99e+03±401.26	4.27e+03±406.49	4.18e+03±268.13
P06	3.01e+03±3.14e+03	894.65±1.31e+03	1.02e+03±1.62e+03
P07	0.94±0.11	1.48e-05±1.44e-05	1.14e-05±8.43e-06
P08	21.19±0.04	**21.14±0.06**	21.18±0.04
P09	332.32±13.52	**313.40±11.45**	326.12±10.58
P10	341.78±15.66	**316.36±14.23**	326.60±11.13
P11	74.45±1.79	73.74±1.17	73.67±1.67
P12	5.61e+06±3.38e+05	**5.27e+06±3.03e+05**	5.55e+06±5.10e+05
P13	30.35±1.08	28.36±1.14	29.46±0.83
P14	23.08±0.16	**22.9686±0.15**	23.08±0.15
P15	7.79e-11±7.04e-011	0±0	0±0
P16	3.37e-13±1.05e-013	2.84E-14±0	2.84E-14±0

also ran our adaptive method to automatically select the value of the degree of freedom as the optimisation progresses to optimise each problem.

From the comparisons shown in figure 1, we can see that our method, tRP-Ens-EDA(5df) has demonstrated superiority over RP-Ens-EDA by outperforming it in 8 out of the 16 problems and it has almost the same performance with RP-Ens-EDA on 2 of the problems with RP-Ens-EDA outperforming our method in only two problems. Therefore, we can conclude that our method, tRP-Ens-EDA(5df) is superior to RP-Ens-EDA. However, this is not surprising since it used more budget of function evaluations per generations due to the number of concurrent trials it had to make. We also performed experiments with the version of our adaptive method that uses 2 values concurrently, and found, as expected, that it performs slightly inferior to the version that uses 5 concurrent values – although we need to bear in mind the tradeoff that using more concurrent values means more fitness evaluations per generation.

2) Tuning vs. adaptation in equal budget experiment: In this set of experiments, we compare tuning with adaptation under equal budget of total number of function evaluations. Tuning encompasses separate runs with the degree of freedom parameter being fixed to a value in the set $\{5, 6, 7, 10, 12, 30\}$. The number of generations was 300 with a population size of 300. This means the total number of function evaluations available for the tuning method was $5.4 \cdot 10^5$. Hence we gave the same amount of function evaluations to our two adaptive methods. This means that tRP-Ens-EDA(2df) ran for 900 generations with a population size of 300 and tRP-Ens-EDA(5df) ran for 360 generations with same population size. The aggregated results are summarised in figure 2. We see that our method with 2 concurrent values of degree of freedom is the out performed the other two. This is confirmed in a statistical test results in table III where bold font indicates statistical out-performance.

Therefore, with equal budget, it is better to use the method that uses less df, say 2df concurrently as it is more cost effective than the other methods. This is true because the tuning method only uses pre-defined dfs to choose from and does not try all other possible values, hence the inferiority. Our method that uses 5 df concurrently uses lots of function evaluation just to try out possible dfs and does not have much left to create new generations, while our version that uses 2 df concurrently uses less function evaluations than the other two methods to try out possible dfs and greater amount of function evaluations are used to create new generations, thus giving it the advantage of performing better than the rest of the methods.

3) Comparison with state of the art methods on 1000 dimensional problems: In this section, we show results of comparing our method $tRP-Ens-EDA(5df)$, with other state of the art performing methods on the 1000 dimensional CEC 2010 problems. The results of this comparison are summarised in table IV. We present the average and std of best fitness results after 3 million function evaluations throughout the EDA process. Bold font indicate statistical out-performance. Out of

11 problems, our method was able to outperform all the other methods in 4 of the problems while being very competitive in the other problems where significant out-performance were not registered. Other methods like $CCVIL$ and the famous $sep-CMA-ES$ where each able to win in 2 out of the 11 problems. Therefore, our method has once again proved to be very promising.

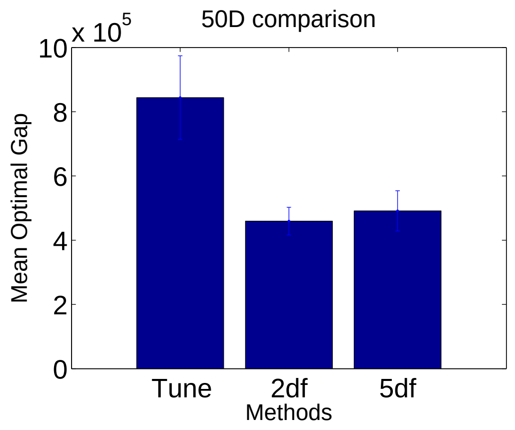


Fig. 2: Summary of the comparison experiment with equal budget set to $5.4 \cdot 10^5$.

V. Conclusion and future work

We devised a new approach for high dimensional continuous black-box optimization by building on RP ensemble based EDA. Our focus has been on the utility of the heavy tailed distributions as entries of our RP matrices. Our results suggest that taking RP matrices with entries drawn with i.i.d. t-distribution increases exploration and at the same time maintains exploitation and focus. We have demonstrated superiority of our method in comparison with a number of state of the art methods on 1000 dimensional problems. Our future work will focus on further developing our method, by adapting other parameters such as the dimension of the reduced space, k.

Acknowledgements

We thank Xiaofen Lu and Prof. Xin Yao for fruitful discussions on adaptive methods and Hanno Muehlbrandt for carrying out part of the experiments for table IV.

References

[1] M. Abramowitz and I. A. Stegun (Eds.). Handbook of mathematical functions with formulas, graphs and mathematical tables (applied mathematics series 55). Washington, DC: NBS., 1964.

[2] R. Armañanzas, I. Inza, R. Santana, Y. Saeys, J.L. Flores, J.A. Lozano, Y. Van de Peer, R. Blanco, V. Robles, C. Bielza, and P. Larrañaga. A review of estimation of distribution algorithms in bioinformatics. *BioData Mining*, 1(6), 2008.

[3] J.S. De Bonet, C.L. Isbell, and Jr. P. Viola. Mimic: Finding optima by estimating probability densities. In *Neural Information Processing Systems. Vol. 9, pp. 424-430*, 1997.

[4] P.A.N. Bosman and D. Thierens. An algorithmic framework for density estimation based evolutionary algorithms. Technical report, UU-CS., 1999.

[5] W. Chen, T. Weise, Z. Yang, and K. Tang. Large-scale global optimization using cooperative co-evolution with variable interaction learning. In *PPSN XI, Lecture Notes in Computer Science, Vol. 6239, pp. 300-309*, 2010.

[6] W. Dong, T. Chen, P. Tiňo, and X. Yao. Scaling up estimation of distribution algorithm for continuous optimisation. *IEEE Transaction of Evolutionary Computation. Vol 17, Issue 6, pp. 797 - 822.*, 2013.

[7] W. Dong and X. Yao. Unified eigen analysis on multivariate gaussian based estimation of distribution algorithms. *Information Sciences, Vol. 178, Issue 15. pp. 3000-3023.*, 2008.

[8] A.E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Transaction on Evolutionary Computationtion. Vol 3, Issue 2, pp. 124-141*, 1999.

[9] N. Hansen. *Towards a New Evolutionary Computation*. Springer Berlin Heidelberg, 2006.

[10] N. Hansen, A. Auger, R. Ros, S. Finck, and P. Pošík. Comparing results of 31 algorithms from the black-box optimization benchmarking bbob-2009. *ACM, NY, USA, pp. 1689-1696*, 2010.

[11] A. Kabán. New bounds for compressive linear least squares regression. In *The 17-th International Conference on Artificial Intelligence and Statistics (AISTATS 2014). Vol 33, pp. 448-456*, 2014.

[12] A. Kabán, J. Bootkrajang, and R.J. Durrant. Towards large scale continuous eda: a random matrix theory perspective. In Christian Blum and Enrique Alba, editors, *GECCO, pp. 383-390*. ACM, 2013.

[13] S. Kern, S.D. Müller, N. Hansen, D. Büche, J. Ocenasek, and P. Koumoutsakos. Learning probability distributions in continuous evolutionary algorithms- a comparative review. *Natural Computing. Vol. 3. Issue 1, pp 77-112.*, 2004.

[14] O. Kramer. Self-adaptive heuristics for evolutionary computation. *PhD thesis, Dept. CS, Univ. of Paderborn, Warburger,*, 2008.

[15] P. Larrañaga and J.A. Lozano. *Estimation of Distribution Algorithms: A new tool for Evolutionary Computation.* Norwell, MA: Kluwer Academic Publishers., 2001.

[16] T.K. Paul and H. Iba. Linear and combinatorial optimizations by estimation of distribution algorithms. In *Proceedings of the 9th MPS Symposium on Evolutionary Computation. IPSJ, Vol. 9, pp. 99–106*, 2002.

[17] K.B. Petersen and M.S. Pedersen. The matrix cookbook. In *Version: October 3, 2005, Page 53*. 2005.

[18] R. Ros and N. Hansen. A simple modification in cma-es achieving linear time and space complexity. In *Parallel Problem Solving from Nature-PPSN X . Springer Berlin Heidelberg. Vol. 5199, pp. 296-305.*, 2008.

[19] M.L. Sanyang, H. Muehlbrandt, and A. Kabán. Two approaches of using heavy tails in high dimensional eda. In *the Proceedings of the IEEE International Conference on Data Mining Workshop. pp 653-660*, 2014.

[20] P.N. Suganthan, N. Hansen, J.J. Liang, K. Deb, Y.P. Chen, A. Auger, and S. Tiwari. Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimisation. Technical report, CEC 2005 Special Session on Real-Parameter Optimisation., 2005.

[21] K. Tang, X. Li, P. N. Suganthan, Z. Yang, and T. Weise. Large scale global optimization 2010. CEC2010 special session. Technical report, Accessed on Sept. 10, 2014, 2010.

[22] K. Weicker and N. Weicker. On the improvement of co-evolutionary optimizers by learning variable inter-dependencies. In *Proceedings of the Congress on Evolutionary Computation. Vol. 3. pp. 1627–1632*, 1999.

[23] Y.Y. Wong, K.H. Lee, K.S. Leung, and C.W. Ho. A novel approach in parameter adaptation and diversity maintenance for genetic algorithms. *Soft Computing. Vol. 7. Issue 8. pp. 506-515*, 2003.

[24] Z. Yang, K. Tang, and X. Yao. Multilevel cooperative co-evolution for large scale optimization. In *IEEE World Congress on Computational Intelligence (CEC 2008). pp. 1663 - 1670*, 2008.

[25] B. Yuan and M. Gallagher. On the importance of diversity maintenance in estimation of distribution algorithms. In *H.G. Beyer and U.M OReilly, editors, Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 05), pp 719-729, New York, NY, USA.*, 2005.