REMEDA: Random Embedding EDA for optimising functions with intrinsic dimension

Sanyang, Momodou; Kaban, Ata

DOI:
10.1007/978-3-319-45823-6_80

REMEDA: Random Embedding EDA for optimising functions with intrinsic dimension

Momodou L. Sanyang[1,2] and Ata Kabán[1]

[1] School of Computer Science, University of Birmingham, Edgbaston, UK, B15 2TT., {M.L.Sanyang, A.Kaban}@cs.bham.ac.uk
[2] School of Information Technolgy and Communication, University of The Gambia, Brikama Campus, P.O. Box 3530, Serekunda, The Gambia. MLSanyang@utg.edu.gm

Abstract. It has been observed that in many real-world large scale problems only few variables have a major impact on the function value: While there are many inputs to the function, there are just few degrees of freedom. We refer to such functions as having a low intrinsic dimension. In this paper we devise an Estimation of Distribution Algorithm (EDA) for continuous optimisation that exploits intrinsic dimension without knowing the influential subspace of the input space, or its dimension, by employing the idea of random embedding. While the idea is applicable to any optimiser, EDA is known to be remarkably successful in low dimensional problems but prone to the curse of dimensionality in larger problems because its model building step requires large population sizes. Our method, Random Embedding in Estimation of Distribution Algorithm (REMEDA) remedies this weakness and is able to optimise very large dimensional problems as long as their intrinsic dimension is low.

Keywords: Estimation of Distribution Algorithm, Black-box Optimization, Intrinsic dimension.

1 Introduction

Optimisation over a high dimensional search space is challenging. However, it has been noted that in certain classes of functions most decision variables have a limited impact on the objective function. Examples include hyperparameter optimisation for neural and deep belief networks [1], automatic configuration of state-of-the algorithms for solving NP-hard problems [8], optimisation problems in robotics [14], and others [3]. In other words, these problems have low intrinsic dimensionality. In the numerical analysis literature [3] the influential parameter subspace has been termed as the 'active subspace', and methods have been developed to estimate this subspace. Fortunately, for optimisation, estimating the influential subspace is not required: In [14] it was shown that a sufficiently large *random* subspace contains an optimum with probability 1, and this was used to dramatically improve the efficiency of Bayesian optimisation by exploiting the low intrinsic dimensionality of problems.

In this paper we further develop the random embedding technique, and introduce it to evolutionary search, by employing it to scale up Estimation of Distribution Algorithms (EDA) for problems with low intrinsic dimension. Although the underlying theoretical considerations are applicable to any optimisation method, our focus on EDA is due to it being one of the most successful methods in low dimensional problems [11] and most unsuccessful or expensive in high dimensions [5, 9, 12].

Definition. A function $f : \mathcal{R}^D \to R$ has **intrinsic dimension** d_i, with $d_i < D$, if there exists a d_i dimensional subspace Υ such that $\forall x \in \mathcal{R}^D$, $f(x) = f(\text{Proj}_{\Upsilon}(x))$.
In the above, $\text{Proj}_{\Upsilon}(x)$ denotes the orthogonal projection, i.e. $\text{Proj}_{\Upsilon}(x) = \Phi\Phi^T x$, where $\Phi \in R^{D \times d_i}$ has columns holding a linear basis of Υ.

The following result in [14] shows that, for such functions, a global optimum exists in a randomly chosen linear subspace – hence a low dimensional search is sufficient.

Theorem 1 ([14]). *Assume we are given a function $f : \mathcal{R}^D \to \mathcal{R}$ with intrinsic dimension $d_i < d$ and a random matrix $R \in \mathcal{R}^{D \times d}$ with independent entries sampled from a standard Gaussian. Then, with probability 1, for any $x \in \mathcal{R}^D$, there exists a $y \in \mathcal{R}^d$ such that $f(x) = f(Ry)$.*

Given some box constraints on the original problem, the authors [14] develop an upper bound on the search box required for the low dimensional search. However, their proof only applies to the case when $d = d_i$, and in practice they recommend a smaller search box and use a slightly larger d. Recall, in practice we have no knowledge of the value of d_i. However, on synthetic problems the experimental results do appear to be better when d is slightly larger than d_i.

In the next section we derive a bound on the search box that holds true for $d > d_i$, and show that the required box size that guarantees to contain a global optimum is indeed smaller when d is larger. Secondly, we devise an EDA optimisation algorithm that implements these ideas employing a random Gaussian embedding.

2 REMEDA: Random embedding EDA

In this section we present our REMEDA algorithm and explain how it exploits the intrinsic dimensionality of problems. Instead of optimising in the high dimensional space, REMEDA will do a random embedding, using the random matrix $R \in \mathcal{R}^{D \times d}$, $d << D$ with i.i.d. entries drawn from a standard Gaussian, and then optimises the function $g(y) = f(Ry)$, $y \in \mathcal{R}^d$ in the lower dimensional space.

The psuedo-code of REMEDA is given in algorithm 1. It takes the population size N, box constraints for a D-dimensional problem, and the internal working dimension $d << D$. As in basic EDA, the REMEDA algorithm then proceeds by initially generating a population of individuals uniformly randomly. However, these individuals are generated in the d-dimensional space, within some suitable box constraints in this space that are determined from the given D-dimensional box. The details of how this is done will follow shortly. The algorithm then evaluates the fitness of these individuals with the use of a random embedding matrix $R \in \mathcal{R}^{D\times d}$ that transforms the d-dimensional points into the original D-dimensional space of decision variables. The matrix R has entries drawn i.i.d. from a standard Gaussian distribution, as in Theorem 1. Based on the fitness values obtained, the fittest individuals are selected using a selection method, such as truncation selection. The maximum likelihood estimates (MLE) of the mean $\mu \in \mathcal{R}^d$ and the covariance $\varSigma \in \mathcal{R}^{d\times d}$ of the promising solutions are computed from the set of selected fittest individuals, and these are used to generate the new generation by sampling from a multivariate Gaussian distribution. The new population is formed by replacing the old individuals by the new ones. We also use elitism, whereby the best individual of the previous generation is kept.

Algorithm 1 The Pseudocode of REMEDA with Population size N and intrinsic dimensionality of the problems, d_i

```
Inputs: $N$, $D$, $d$, Box
(1) Set the search box boundaries in the low-dimensional space $\mathcal{R}^d$ (cf. Theorem 2 & text)
(2) Set $P \leftarrow$ Generate $N$ points uniformly randomly within the box in $\mathcal{R}^d$ to give an initial population
(3) Set $R \leftarrow$ Generate a random embedding matrix, $R \in \mathcal{R}^{D\times d}$.
Do
    (4) Evaluate the fitness of $y_i$ as $f(Ry_i)$, $i = 1...N$
    (5) Select best individuals $P^{sel}$ from $P$ based on their fitness values
    (6) Calculate the mean $\mu$ and covariance $\varSigma$ of $P^{sel}$
    (7) Use the $\mu$ and $\varSigma$ to sample new population, $P^{new}$
    (8) $P \leftarrow P^{new}$
Until Termination criteria are met
Output: $P$
```

We have not yet specified how to determine the d-dimensional box constraints that correspond to the given D-dimensional ones. Given some box constraints in $\mathcal{R}^D$, the following theorem gives the required box constraints for the search in $\mathcal{R}^d$.

Theorem 2. *Let $f : \mathcal{R}^D \to \mathcal{R}$ be a function with intrinsic dimension $d_i < d < D$ that we want to optimise subject to the box constraint $\mathcal{X} \subset \mathcal{R}^D$, where $\mathcal{X}$ is centered around 0. Denote the intrinsic subspace by $\varUpsilon$, and let $\varPhi$ be a $D \times d_i$ matrix whose columns form an orthonormal basis for $\varUpsilon$. Denote by $x_t^* \in \varUpsilon \cap \mathcal{X}$ an optimiser of f inside $\varUpsilon$. Let R be a $D \times d$ random matrix with independent standard Gaussian entries. Then there exists an optimiser $y^* \in \mathcal{R}^d$ such that $f(Ry^*) = f(x_t^*)$ w.p. 1, and for any choice of $\epsilon \in (0,1)$, if $d > (\sqrt{d_i} + \sqrt{2\ln(1/\epsilon)})^2$, then $||y^*||_2 \leq \frac{||x_t^*||_2}{\sqrt{d}-\sqrt{d_i}-\sqrt{2\ln(\frac{1}{\epsilon})}}$ with probability at least $1-\epsilon$.*

Proof. The existence of y^* is guaranteed by Theorem 1, and global optimisers outside the subspace Υ are irrelevant since the function takes all its range of values in Υ. So our focus is to upper bound the length of y^*.

From the proof of Theorem 1 in [14] we know that $\exists y^* \in \mathcal{R}^d$ s.t.

$$\Phi\Phi^T R y^* = x_t^* \tag{1}$$

Hence,

$$||x_t^*|| = ||\Phi\Phi^T R y^*|| \geq s_{\min}(\Phi\Phi^T R)||y^*|| \tag{2}$$

where we use the Rayleigh quotient inequality, and $s_{\min}(\cdot)$ denotes the smallest singular value.

Note that $\Phi\Phi^T R$ is a $d_i \times d$ random matrix with i.i.d. Gaussian entries. When $d = d_i$ it is a square matrix, and a bound on its smallest singular value was applied in [14]. Instead, for the case $d > d_i$ we employ the bound of Davidson & Szarek that applies to rectangular Gaussian matrices [4]. We have for any $\epsilon \in (0,1)$ for which $\sqrt{d} - \sqrt{d_i} - \epsilon > 0$, that:

$$||y^*|| \leq \frac{||x_t^*||}{\sqrt{d} - \sqrt{d_i} - \epsilon} \tag{3}$$

with probability $1 - \exp(-\frac{\epsilon^2}{2})$. Now setting $\exp(-\epsilon^2/2) = \tau$ and solving for ϵ we get $\epsilon = \sqrt{2\ln(\frac{1}{\tau})}$. Plugging this back and renaming τ to ϵ completes the proof. □

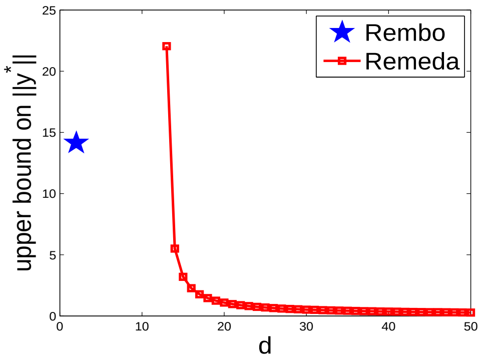


Fig. 1: Comparison of our theoretical bound (Remeda), with various values of $d > d_i$ versus the bound of [14] (Rembo), which holds when $d = d_i$.

In Figure 1 we plotted the bound on the search box from our Theorem 2 for various values of $d > d_i$ in comparison with the bound in [14] for $d = d_i$. We see that our result is tighter for nearly all values of d and it explains why a smaller search box is sufficient when $d > d_i$. The single point for Rembo in figure 1 is for $d = d_i$ where as the curve for Remeda is for values of $d > d_i$.

In practice, of course, we typically have no knowledge of the value of d_i, in which case we cannot use theoretical bounds directly to set our search box. However, we can fix the search box – for instance to $\sqrt{d}$-times the coordinates if the original box, as suggested in [14], and our Theorem 2 then suggests that increasing d can eventually make this fixed-size box sufficiently large to contain the optimiser y^*. This is what we used in the experiments reported.

3 Related Work

Model building in high dimensions is the subject of many recent research efforts, as high dimensionality limits the usefulness of optimisers in practice. Many approaches were proposed, here we will limit ourselves to a few of the most relevant ones.

Among these methods, the Eigendecomposition EDA ($ED - EDA$) [6] proposes to utilise a repaired version of the full covariance matrix estimate, with the aim to capture interactions among all decision variables and guide exploration of the search space. Other methods use limited dependencies. For example, Cooperative Co-evolution with Variable Interaction Learning ($CCVIL$) proposed by *Weicker et al.* in [15] is a deterministic method to uncover dependencies between decision variables, which has later been extended to the CCVIL framework by *Chen et al* in [13]. EDA with Model Complexity Control ($EDA - MCC$) [5] also employs a deterministic algorithm to split the decision variables into two disjoint subsets, of which one set contains decision variables with only minor interaction and the other set contains the strongly dependent variables that are further grouped randomly, and inter-group dependencies are neglected. Other methods include Covariance Matrix Adaptation *(CMA-ES)* [7], separable CMA-ES (*sep-CMA-ES*) [10] and Multilevel Cooperate Co-evolution ($MLCC$) [16].

There are also methods that apply dimensionality reduction techniques to reduce the dimension of the problems in order to avail EDA the opportunity to demonstrate its capabilities. An example of this type of techniques are random projections [9], [12] and [14]. However, none of these methods have been designed to take advantage of the intrinsic structure of the problems as our REMEDA approach does.

4 Experiments

4.1 Test functions and performance measures

We created test functions with intrinsic dimension 5 from existing benchmark functions, by embedding the d_i-dimensional versions of these problems into higher D-dimensions. That is, we add $D - 5$ additional dimensions which do not impact on the function value, and (optionally) rotate the search space around the origin in a random direction. Hence, the functions will take D-dimensional inputs, but only 5 linear combinations of these input variables determine the function value. The algorithm will have no knowledge of which these directions are, not even that there are 5, but it has knowledge that the number of important directions is much less that D. The functions we employed in this way here are the following: Shifted Ellipse, Shifted Schwefel's problem 1.2, Shifted Rotated High Conditional Elliptic function, and Shifted Rosenbrock function. We also took the Branin function from [14] which has intrinsic dimension 2. The functions are listed in table 1.

Table 1: Test functions of low intrinsic dimension of 2 or 5. o is the shift vector.

PN	Name	Expression
1	Sphere	$\sum_{j=1}^{d_i}(x_j - o_j)^2$
2	Ackley's	$20 - 20\exp(-0.2\sqrt{\frac{1}{d_i}\sum_{j=1}^{d_i}((x_j - o_j) * M)^2})$-
		$\exp(\frac{1}{d_i}\sum_{j=1}^{d_i}(\cos(2\pi(x_j - o_j) * M))$+e
3	Elliptic	$\sum_{j=1}^{d_i}(10^6)^{\frac{j-1}{d_i-1}} * (x_j - o_j) * M)$
4	Rosenbrock	$\sum_{j=1}^{d_i-1}(100(z_j^2 - z_{j+1})^2 + (z_j - 1)^2)$
		$z = x - o + 1$
5	Branin	$(-1.275\frac{x_1^2}{\pi^2} + 5\frac{x_1}{\pi} + x_2 - 6)^2$
		$+(10 - \frac{5}{4\pi})\cos(x_1) + 10$

We employ two common performance indicators: (i) The fitness gap achieved under a fixed budget is the difference between the best fitness achieved and the true optimum; (ii) The scalability is the budget of function evaluations needed to reach a pre-defined value to reach.

4.2 Results and Discussion

Experiments on a $d_i = 2$ problem In the first set of experiments we consider the $D = 25$ dimensional Branin function that has intrinsic dimension $d_i = 2$. Though, we should note that D can be as large in principle, as we like since the working strategy and the budget usage of REMEDA are independent of D. In this experiment, we vary the internal working dimension d, and the population size N, under a fixed budget of 500 function evaluations.

The results are shown in table 2, as obtained from 50 independent repetitions of each experiment. We can see from table 2 that $d = d_i = 2$ is not the best choice, as the size of the search box is not sufficient at $d = d_i$. Also observe that increasing d beyond 4 drops the performance – this is because searching in a larger dimensional space is not necessary and is less resource-effective. Furthermore, we see for all d tested, the higher the population sizes, the worse the performance. This is because a large population unnecessarily uses up the bud-

Table 2: Fitness gap achieved by REMEDA on the Branin function ($d_i = 2$ embedded in $D = 25$), with a total budget of 500 function evaluations.

Pop. size	d=2		d=4		d=6	
	Mean	std	Mean	std	Mean	std
300	1.4297	2.601	2.4908	3.1013	3.9007	3.0322
150	0.4128	0.6607	1.1701	1.313	2.1368	2.1046
80	0.826	2.9973	0.4193	0.4459	0.8303	0.9331
40	0.6375	2.9073	0.04	0.0969	0.1927	0.3865
30	0.6737	2.4939	0.0336	0.0853	0.1038	0.2615

get when the search only happens in a small dimensional subspace. With these insights in place, next we carry out a more comprehensive study.

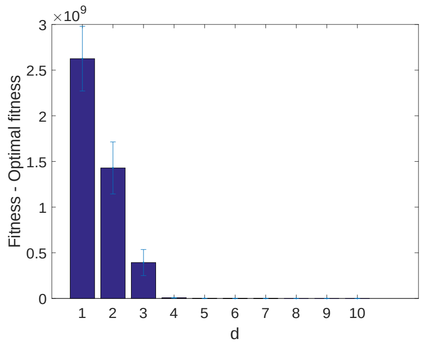
Results and comparisons on problems with $d_i = 5$ In this section, we compare our method with state of the art approaches in heuristic optimisation, on problems with intrinsic dimension $d_i = 5$. The ambient dimension was $D = 1000$ in these experiments, but as already mentioned this can be much higher without causing problems as long as d_i stays low.

We expect that REMEDA should gain advantage from its ability to exploit intrinsic dimensional property while other methods have not been designed to make use of such structure. On the other hand, REMEDA needs to find a good value for its internal dimension d without knowing d_i (as this information is normally not available in practice). This will use up part of the budget, but the hope is that it will pay off by a speedy progress in the search.

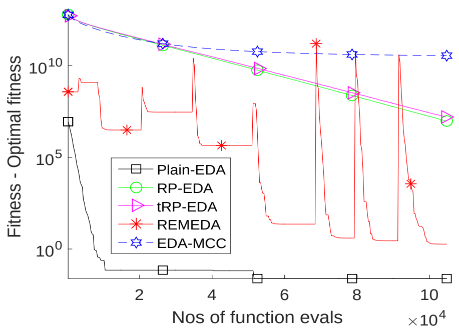
We start with $d = 1$, using convergence as a stopping criterion, and move up progressively to higher values of d until the fitness reached upon convergence is no longer improved by the increase of d. Within each value of d tried, we run REMEDA to convergence, until the relative change in fitness is below a threshold: $\frac{f(t)-f(t+1)}{f(t)} < 10^{-8}$, where t is the generation count and f is the fitness value. When this condition is satisfied, we move on to the next value of d, and re-initialise the population randomly (although other schemes could also be investigated). The total number of fitness evaluations used throughout this process is the total budget that we then provide to the competing algorithms.

The bar chart in the leftmost plot of Figure 2 shows an example of the fitness gaps achieved at convergence with consecutive values of d. The error bars show one standard error from 25 independent repetitions. In the rightmost plot we show the evolution of the best fitness. Superimposed, we also show the trajectories of competing state of the art methods: EDA-MCC [5], RP-EDA [9], and tRP-EDA [12]. All use the same budget and same population size. – For each d tried, we plot their concatenated trajectories in such a way that the next starts from the end of the current one.

From Figure 2 we can see that REMEDA attains a fitness value close to the optimum efficiently, while the other methods are not able to achieve the same within the same budget. We also superimposed an idealised version of plain EDA – that is a plain EDA that receives the d_i-dimensional version of the problem – and we see that REMEDA is nearly as good.



(a) Shifted Rosenbrock Function, Chosen $d = 10$



(b) Shifted Rosenbrock Function

Fig. 2: Finding d (left) and evolution of best fitness (right) for REMEDA, 3 competing methods, and a d_i-dimensional EDA on the idealised problem. Results are averaged over 25 independent runs. All methods use the same population size.

Table 3: Comparing REMEDA with other state of the art methods.

Fn	REMEDA		sep-CMA-ES		EDA-MCC		tRP-ENS-EDA		RP-ENS-EDA	
	Mean	std	Mean	std	Mean	std	Mean	std	Mean	std
F1	**0**	**0**	3.81E+04	2.16E+04	6.41E+04	8.21E+03	37.80	3.17	25.79	2.05
F2	**0**	**0**	1.00E+07	7.92E+05	1.99E+06	1.42E+06	1.40E+08	7.99E+06	1.38E+08	6.59E+06
F3	**0.18**	**0.91**	4.75E+07	3.47E+06	2.90E+09	2.29E+08	6.81E+08	5.60E+07	2.84E+08	2.62E+07
F6	**45.92**	**216.11**	5.44E+06	2.09E+06	3.56E+10	4.93E+09	1.60E+07	1.88E+06	9.64E+06	1.44E+06
F8	**14.19**	**7.89**	21.67	0.01	21.34	0.06	21.66	0.01	21.43	0.06

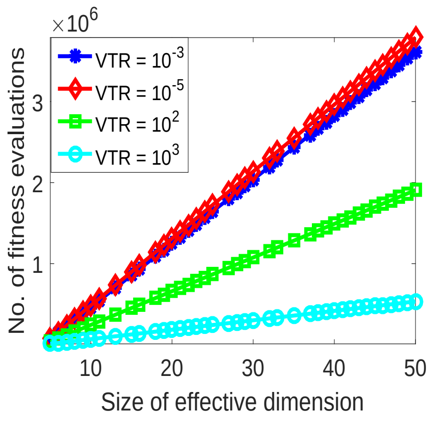
4.3 Scalability experiments

Function evaluation is costly in most practical problems. Here we study what is the required budget of function evaluations to reach a specified value of the fitness gap.

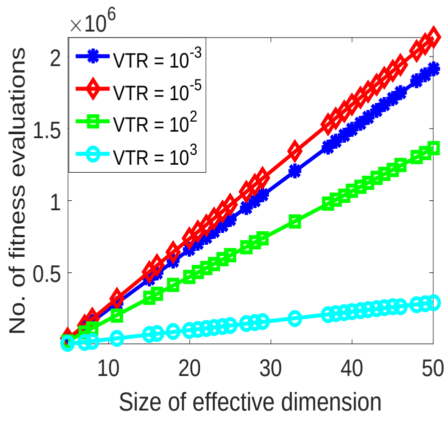
Before running these scalability experiments, we carried out some experiments to determine the required population size as a function of the intrinsic dimension of the problem, so that we can vary the latter and set the population size automatically. For this we use a bisection method as in [2]. To find the population size that results in the lowest number of evaluations required to reach a pre-determined fitness gap value to reach (VTR), we start from a very large population size that can solve the problem within a large predetermined budget and then search for a small population size that cannot solve the problem anymore. In between these limits we use binary search to find the optimal population size. We repeated this 25 times and took the average.

We fix the value to reach (VTR) to 10^{-5}, and vary the intrinsic dimensionality of the problem $d_i \in [2, 50]$. We count the number of fitness evaluations needed for our proposed REMEDA to reach the VTR. The same experiment was repeated for three other choices of VTR: 10^{-3}, 10^2 and 10^3 in order to make sure that the conclusions will not be specific to a particular choice of the VTR. In all these experiments the maximum fitness evaluations was fixed to 6×10^3, so the algorithm stops when the budget is exhausted.

Figure 3 shows the average number of function evaluations as computed from the successful runs out of 25 independent repetitions for each problem, with each intrinsic dimension tested. From the figure, we observe a linear fit on the scalability measurements.



(a) Shifted Sphere Function



(b) Shifted Rastrigin Function

Fig. 3: Number of function evaluations taken by successful runs of REMEDA to reach a pre-specified value to reach (VTR) as the problem intrinsic dimensionality is varied in $d_i \in [2, 50]$. The markers represent averages computed from 25 independent repetitions.

5 Conclusions and Future Work

We proposed random embedding in Estimation of Distribution Algorithm to scale up EDA by exploiting the intrinsic dimension of problems whereby the search takes place in a much lower dimensional space than that of the original problem. Our method is suited for large scale problems that take a large number of inputs but only depend on a few linear combinations of them. On such problems we have demonstrated that our method outperforms the best state of the art algorithms in evolutionary computation. Our technique and its theoretical basis are applicable in principle to any optimisation method, and in the light that problems with intrinsic dimension are quite prevalent in real-world applications, it seems a worthwhile avenue for future work to make use of it more widely.

References

1. J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Machine Learning Research*, 13:281305, 2012.
2. P. Bosman. On empirical memory design, faster selection of bayesian factorizations and parameter-free Gaussian EDAs. In *Proceedings of Genetic and Evolutionary Computation Conference (GECCO), pp. 389-396. ACM*, 2009.
3. P. Constantine. Active subspace methods in theory and practice: applications to kriging surfaces. *SIAM J. Sci. Comput. 36(4), pp. 1500-1524*, 2013.
4. K.R. Davidson and S.J. Szarek. Local operator theory, random matrices and banach spaces, in handbook of the geometry of banach spaces, vol 1. pp. 317-366., 2001.
5. W. Dong, T Chen, P. Tino, and X. Yao. Scaling up estimation of distribution algorithm for continuous optimisation. *IEEE Transaction of Evolutionary Computation. Vol 17, Issue 6.*, 2013.
6. W. Dong and X. Yao. Unified eigen analysis on multivariate gaussian based estimation of distribution algorithms. *Information Sciences*, 2008.
7. N Hansen. The cma evolution strategy: a comparing review. *Springer*, 2006.
8. F. Hutter. *Automated Configuration of Algorithms for Solving Hard Computational Problems.* PhD thesis, 2009.
9. A. Kabán, J. Bootkrajang, and R.J. Durrant. Towards large scale continuous eda: A random matrix theory perspective. *Evolutionary Computation, MIT Press*, 2015.
10. R. Raymond and N. Hansen. A simple modification in cma-es achieving linear time and space complexity. In *Parallel Problem Solving from Nature-PPSN . Springer Berlin Heidelberg, 296-305.*, 2008.
11. M. L. Sanyang and A. Kabán. Multivariate cauchy eda optimisation. In *IEEE IDEAL, LNCS 8669, pp. 449-456*, 2014.
12. M. L Sanyang and A. Kabán. Heavy tails with parameter adaptation in random projection based continuous EDA. In *IEEE Congress on Evolutionary Computation (CEC), pp. 2074-2081*, 2015.
13. Z. Yang W. Chen, T. Weise and K. Tang. Large-scale global optimization using cooperative co-evolution with variable interaction learning. In *Parallel Problem Solving from Nature, Bd. 11, pp. 300-309, 2010.*, 2010.
14. Z. Wang, M. Zoghi, F. Hutter, D. Matheson, and N. de Freitas. Bayesian optimization in a billion dimensions via random embeddings. In *IJCAI*, 2013.
15. 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 on. Vol. 3.*, 1999.
16. K. Tang Z. Yang and X. Yao. Multilevel cooperative co-evolution for large scale optimization. In *IEEE World Congress on Computational Intelligence 2008 (CEC 2008)*, 2008.