**SPRING SEMESTER 2021/22**

**COMP2024 Coursework**

|  |  |
| --- | --- |
| Project title | Technical Report of Stochastic Optimizers |
| Date | ??/04/2022 |
| Group | 6 |
| Leader | Tan Zhun Xian (20313854) |
| Other Group Members | * Heng Yuen Qi * Azzam Eusuf Chaai Bin Amier * Wong Khai Hin * Sabrina Felicia Kusumawati |
| Supervisor | * Tissa Chandesa * Siang Yew Chong |

**We declare that we have read and understood the University’s Academic Integrity and Misconduct statements and policies.**

**Literature Review**

**Metaheuristics**

A metaheuristic is a problem-independent, high-level algorithmic framework that provides a set of recommendations or techniques for developing heuristic optimization algorithms [1]. Genetic/evolutionary algorithms, tabu search, simulated annealing, and ant colony optimization are all examples of metaheuristics, but there are many more. A metaheuristic is a problem-specific implementation of a heuristic optimization algorithm that follows the rules provided in a metaheuristic framework. Glover (1986) invented the phrase, which combines the Greek prefix meta- (metá, high-level beyond) with heuristic (from the Greek heuriskein or euriskein, to search).

There are three general categories for metaheuristics: **evolutionary**, **physics-based** and **swarm intelligence** algorithms.

**Evolutionary**

Nearly three decades of research and applications have clearly demonstrated that modelling the search process of natural evolution can yield very robust, direct computer algorithms, although these models are crude simplifications of biological reality. The resulting *evolutionary* *algorithms* [2]are based on the collective learning process within a population of individuals, each of which represents a search point in the space of potential solutions to a given problem. The population is arbitrarily initialized, and it evolves toward better and better regions of the search space by means of randomized processes of *selection* (which is deterministic in some algorithms), *mutation,* and *recombination* (which is completely omitted in some algorithmic realizations). The environment delivers quality information *(fitness value)* about the search points, and the selection process favours those individuals of higher fitness to reproduce more often than those of lower fitness. The recombination mechanism allows the mixing of parental information while passing it to their descendants, and mutation introduces innovation into the population. \*

**Physics-based**

In recent years, several optimization methods especially metaheuristic optimization methods have been developed by scientists. People have utilized power of nature to solve problems. Therefore, those metaheuristic methods have imitated physical and biological processes of nature. In 2007, Big Bang Big Crunch [3] optimization algorithm based on evolution of universe and in 2009, Gravitational Search Algorithm [4] based on gravity law have been proposed and have been applied to solve complex problems. \*

**Swarm intelligence**

Swarm intelligence (SI) [5] is a subset of artificial intelligence (or bio-inspired computation in general) (AI). It was first coined by Gerardo Beni and Jing Wang in 1989 in the context of building cellular robotic systems, and it has been recognised as an emerging field. The growing popularity of such SI-based algorithms can be attributed to a number of factors, the most important of which being the flexibility and variety they provide. The algorithms' primary features are their ability to self-learn and adapt to environmental variations, which has sparked a lot of attention and led to the identification of various application areas. Swarm intelligence has gained appeal in recent years as NP-hard issues have become more prevalent, making finding a global optimum in a real-time setting nearly impossible.

**Optimizers**

**CMAES**

The covariance matrix adaptation evolution strategy (CMA-ES) is one of the most powerful evolutionary algorithms for real-valued optimization [6] with many successful applications [7]. The main advantages of the CMA-ES lie in its invariance properties, which are achieved by carefully designed variation and selection operators, and in its efficient (self-) adaptation of the mutation distribution. The CMA-ES is invariant against order-preserving transformations of the fitness function value and in particular against rotation and translation of the search space—apart from the initialization. If either the strategy parameters are initialized accordingly or the time needed to adapt the strategy parameters is neglected, any affine transformation of the search space does not affect the performance of the CMA-ES. \*

**Differential Evolution**

Evolutionary algorithms (EAs) are inspired by the natural evolution of species. It involves mutation and crossover. A lot of optimisation problems have been solved by EAs in many areas. However, we need to carefully choose the appropriate parameters and encoding schemes in order to optimise the problem that we have. Otherwise, computational costs will be expensive [8].

Differential evolution (DE) is one of the EAs. It is a population-based optimisation algorithm that can be used to solve many practical problems and the results are quite satisfactory. However, it is not so efficient to solve non-separable functions. This is in light of the fact that crossover helps in separable function and on the other hand, it destroys possible combination of good offspring in non-separable function. \*

**Genetic Algorithm**

Binary Genetic Algorithm or also known as Genetic Algorithm (GA) was introduced by John Henry Holland and his associates in 1975 in a book published by the MIT press [9]. This algorithm is a computational model inspired by evolution. In relation to its name, the way component vectors are configured within this algorithm follows the genetic structure of a chromosome [9]. Its main idea takes from natures natural selection/survival of the fittest and works by simulating evolution, starting with an initial set of solutions or hypotheses and generating future “generations” of solutions [10]. \*

**Particle Swarm Optimization**

Particle swarm optimization (PSO) [11] is a simple evolutionary algorithm that searches for an optimal solution in the solution space. It differs from other optimization techniques in that it simply requires the objective function and is unaffected by the gradient or any differential form of the objective. It also has a small number of hyperparameters. PSO is a metaheuristic as it makes few or no assumptions about the problem being optimized and can search very large spaces of candidate solutions.

Particle swarm optimization (PSO) [11] optimizes a problem by iteratively attempting to enhance a candidate solution in terms of a particular quality measure. It solves a problem by generating a population of possible solutions, which are referred to as particles, and moving them around in the search space using a simple mathematical formula based on their position and velocity. The movement of each particle is controlled by its local best-known position, but it is also directed toward the best-known positions in the search space, which are updated when better positions are discovered by other particles. The swarm is predicted to migrate toward the best options as a result of this. \*

**Optimizer Configurations**

**BIPOP-CMA-ES**

After a first single run with default population size, we use two interlaced multi-start regimes, each equipped with a function evaluation budget accounting for the so far conducted function evaluations, after a single run with default population size. A complete run of either one or the other strategy is initiated, depending on which budget figure is lower. Under the first regime, the first and last restarts are carried out.

In the first regime, we resume with rising population size, increasing the population size by a factor of two before each restart. There are a total of nine restarts which comes out to a maximum factor of 512. In the second regime, multi-start regime with small population size is applied. The second multi-start regime begins if and only if its recent budget is less than that of the first multi-start regime with growing populations.

**DEAE**

The DE algorithm will be made rotationally invariant by this Adaptive Encoding (AE) framework [12]. This type of algorithm is needed as modern benchmark normally uses rotation matrices. This matrix is used to increase the difficulty to solve the problem as real-world problems are normally very difficult to solve. The notion of rotation invariant is that we can get the same output after we rotate the input [13].

In general, AE framework consists of three steps, that are encode, decode and update. Encoding means transforming the population into a space which gives benefits to the modification operators. A key to take note is that the offsprings are evaluated in the original space. So, it needs to be decoded first. After that, the transformation matrix is adjusted in the update step [14]. \*

**Genetic Algorithm**

The GA optimizer was initialised with floor (sqrt (5000) \* 4) = 282 population size. The individuals in the population have DIM number of chromosomes with 32 bit each. The selection operator was set to tournament selection with a 2-point crossover operator. Mutation probability was set to even.

**PSO\_Bounds**

The algorithm used is a simple PSO algorithm utilizing the global best model. The only design choice made was to select the absorbing boundaries to handle any particles leaving the search space, where the position is set to the boundary and the velocity is reset to zeros [15]. The swarm has 40 particles with the parameters set as c1 = c2 = 2. \*

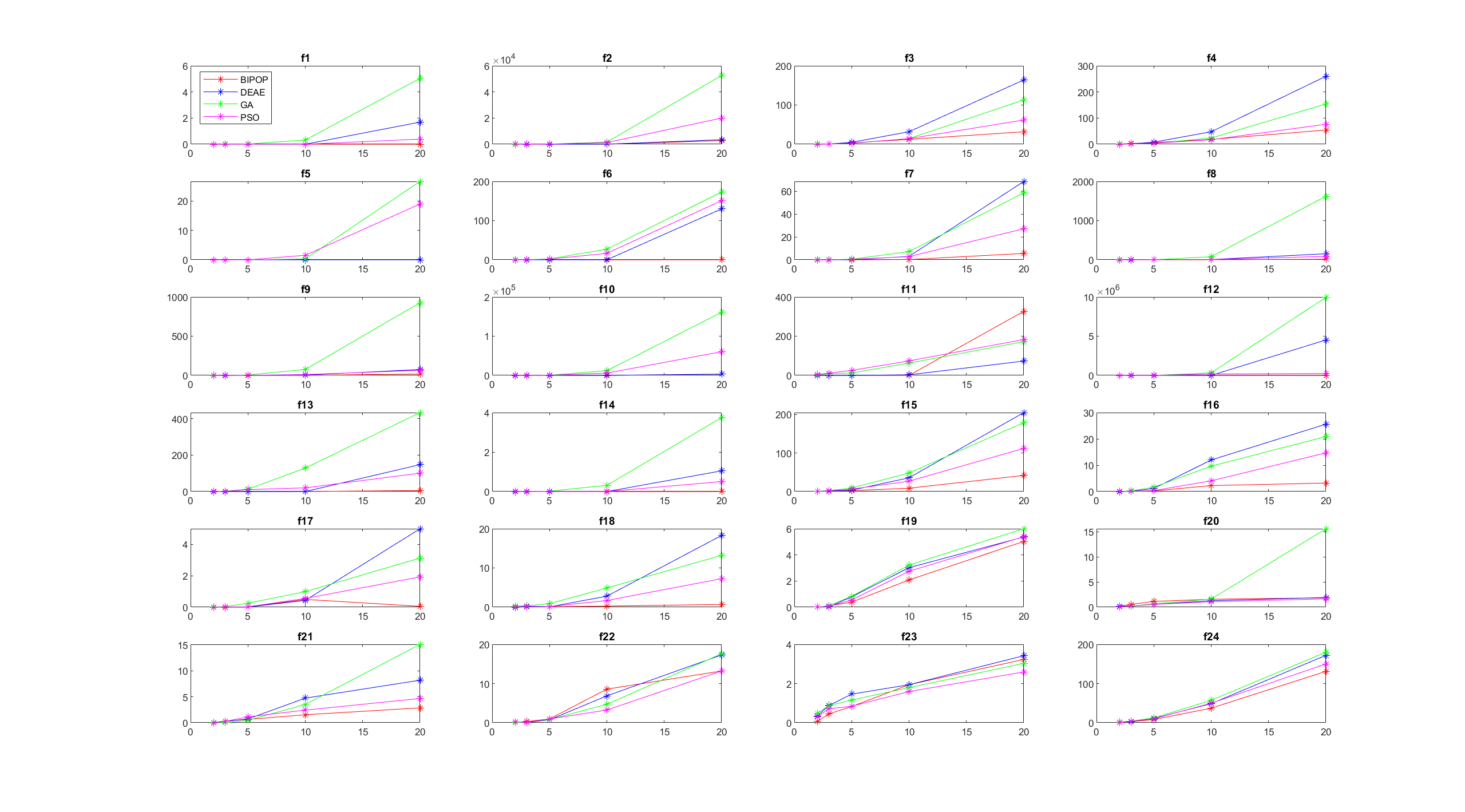
**Results**

Results below show the results of the 4 optimizers tested with 5000 maximum function evaluation 15 times with initial seed 20313854 at 5 dimensions [2D,3D,5D,10D,20D] plotted on a graph. Average and STD are calculated and shown below.

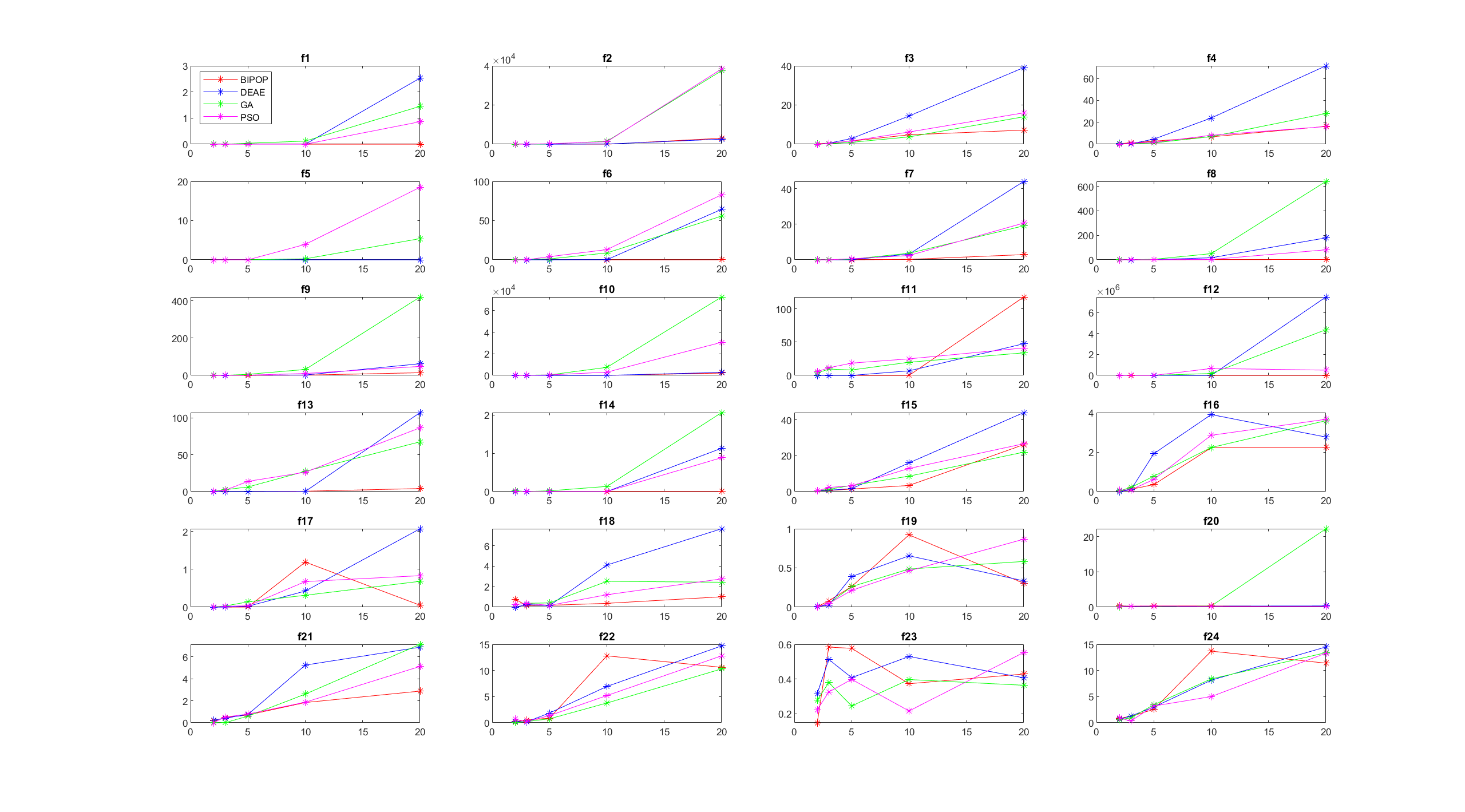
Full results are linked here.

Boxplots are also provided at these links: SortByDimensions | SortByOptimizers

**Average**

****

**STD**

****

**Observation of Results**

At lower dimensions, all optimizers perform similarly. However, once we reach higher dimensions, we find that the results quickly diverge. Generally, the BIPOP performs the best out of the 4 optimizers, having the lowest average score more often than not. However, an exception does occur for f11 as BIPOP was found to perform the worst at 20-D. We find that GA has the most occurrences of highest average scores at 20-D followed by DEAE which has the highest average score half as often as GA.

The BIPOP optimizer was found to have the lowest average scores across the board. After losing out to DEAE at 2-D, BIPOP manages to surpass the other optimizers beginning from 3-D before getting a supermajority of the lowest average scores starting from 10-D. BIPOP performs quite well at low, moderate and high condition functions especially at lower dimensions, achieving a substantial amount of 0 average scores. BIPOP also performs quite consistently at these conditions, achieving low standard deviations. Of course, when the average score is 0 then standard deviation is also 0. BIPOP also generally performs better than the other optimizers at multi-modal functions, achieving low average scores and standard deviations.

The DEAE optimizer manages to beat out BIPOP at 2-D, getting the lowest average scores. However, this does not last as the share of lowest average scores decreases at the number of dimensions increases as the scores are slowly dominated by BIPOP. Remarkably, at lower dimensions, DEAE was found to achieve more 0 average scores than BIPOP, dominating the low, moderate and high condition functions at lower dimensions. However, as mentioned above, this does not last as we reach higher dimensions. Unfortunately, DEAE was shown to get its fair share of highest average score as shown above especially on higher dimensions like 20-D.

The GA optimizer was only shown to perform well in separable functions and multi-modal functions with weak global structure at lower dimensions, dominating f3-f5 at 3-D and 5-D while performing well at f20-f22 at lower dimensions. After 10-D, GA has not been able to get even a single lowest average score. GA was also shown to get the highest average scores especially at 20-D.

The PSO optimizer was shown to perform extremely well at separable functions at 2-D, even outperforming the likes of BIPOP and DEAE. However, as the number of dimensions grows, separable functions was then dominated by GA and then by BIPOP. However, PSO was shown to dominate in multi-modal functions with weak global structure especially in higher dimensions. This is quite a surprise as usually BIPOP dominates at higher dimensions.

Overall, we can see that as the number of dimensions gets larger, the average results for all functions also increases. The overall average shows that BIPOP dominates the results overwhelmingly, with DEAE and PSO fighting over the remainder. GA was completely wiped out.

Looking at fsmap scores, we see that BIPOP beats out DEAE with only a slim margin. PSO received only half the score as the top 2 optimizers while GA received an embarrassingly low score. Although DEAE was seen to perform much worse at higher dimensions than PSO, its score was much higher than PSO.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **BIPOP** | **DEAE** | **GA** | **PSO** |
| **2-D** | 4160 | 4345 | 969 | 2919 |
| **3-D** | 3305 | 3572 | 469 | 1573 |
| **5-D** | 2553 | 2567 | -44 | 433 |
| **10-D** | 729 | 361 | -511 | -95 |
| **20-D** | 60 | -484 | -802 | -564 |
| **Total** | 10807 | 10361 | 81 | 4266 |

**Analysis of Results**

The reason for the similar results of optimizers at lower dimension is the low complexity of the functions which allow the optimal result to be quickly and easily reached. As the number of dimensions increases, the average score also increases as the complexity of the function increases with increasing dimensions.

In separable functions f1 - f4, BIPOP performs the best. This is due to its ability to simplify the search process to D times one-dimensional search procedures, allowing them to achieve high accuracy with fewer function evaluations by utilising the separable property. When a solution with a lower objective function value is found in CMA-ES, the children are chosen as the next parent, Xk+1 as well as the best two offspring and the parent Xk. The optimizer performs well in terms of avoiding local minima and small-scale exploration.

Low and moderate condition functions f6 - f9 performs the best with the BIPOP optimizer, which means they can avoid huge plateaus and handle near-zero or zero gradients well. BIPOP also works best in functions f10 - f14 with high conditioning, demonstrating that it can be precise with micro-movements towards the optimum when on a steep ridge or peak.

BIPOP is also the most accurate and most consistent at Multimodal functions f15 – f19. This optimizer has good exploitation but limited exploration since it gets caught in local optima. However, if it lands in a favourable valley, it can minimise to a very low result.

Finally, in multimodal functions with weak global structure f20 – f24, BIPOP performs quite well. It can find strong scores even when there appears to be no pattern or structure to exploit in the search area. The population size of BIPOP plays a significant role in achieving the best precision in multimodal functions f20-f24. For each function evaluation, BIPOP generates a given population size of offspring, identifies the closest offspring to the target, then generates another set of offspring based on that closest offspring. Even in a weak global structure, where improvements between locations are difficult to discern via point-by-point exploration, this can ensure that the likelihood of finding a solution closest to the target can be identified and will rise depending on the population size set by parameter tuning.

For the other optimizers, DEAE, GA and PSO are able to perform relatively well at lower dimensions but their results get worst as the number of dimensions increases. This is due to the increase in function complexity as the number of dimensions increases. This might be also due to the fact that the optimizers fell into a local optimum with a large convergence basin and are unable to escape if exploitation is prioritised over exploration. If exploration is prioritised over exploitation, then the optimizer may jump from one local optima to the next without minimizing the score.

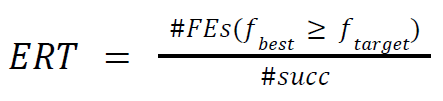
Looking at the average plots and the fsmap results, we find the even though DEAE was shown to get its fair share of highest average scores, it manages to vastly outscore PSO and GA in the fsmap results. This is due to the fact that fsmap uses the function result = floor (-log10 (answer)) which varies logarithmically. So, even if DEAE performs poorly at higher dimensions, it will not affect its results as much as performing poorly at lower dimensions such as in the case of GA.

**Post-Processed Data**

We used the python script package provided to construct tables and figures reflecting the results of the benchmarking experiment, and we compared the results of each optimizer. The complete post processing findings for each optimizer derived from BB0B-2010 experiment data, as well as comparisons, can be found at the following links:

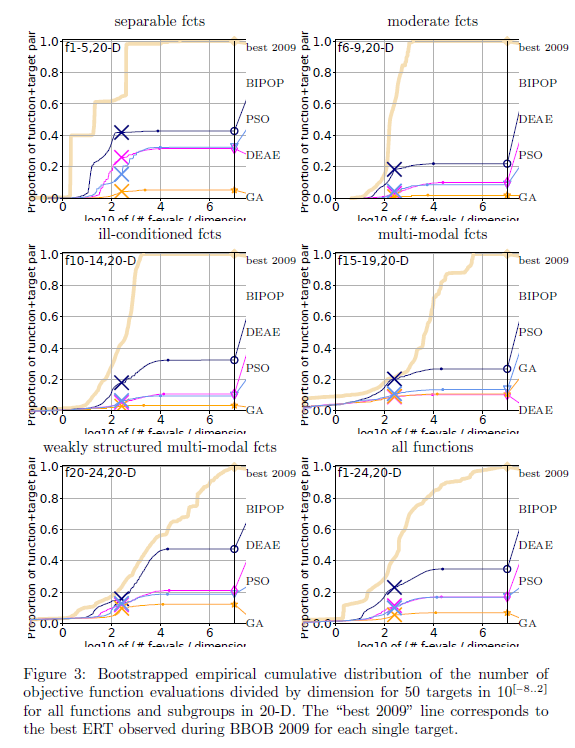
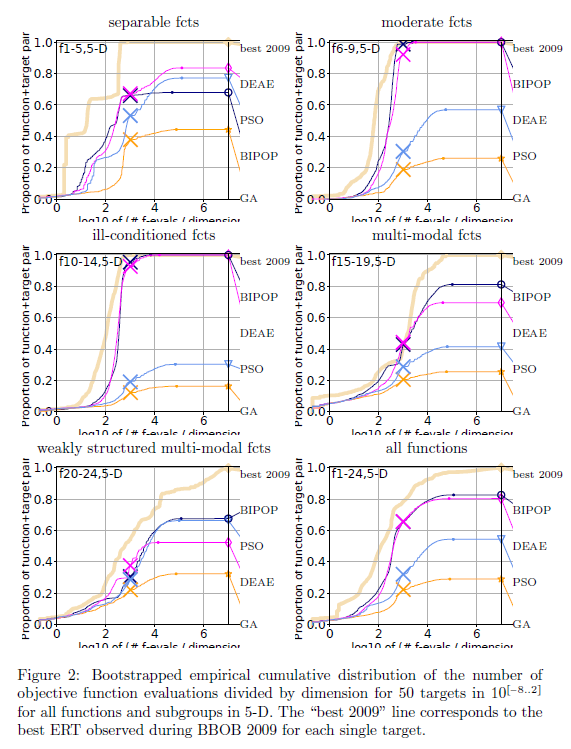
1. BIPOP
2. DEAE
3. GA
4. PSO
5. Comparison between all 4 optimizers

The overall performance of each optimizer is calculated using the Expected running time (ERT) and denotes the expected number of function evaluations to reach a target function value for the first time. ERT is defined as



where #𝐹𝐸𝑠 (𝑓𝑏𝑒𝑠𝑡 ≥ 𝑓𝑡𝑎𝑟𝑔𝑒𝑡) is the total number of function evaluations in which the best function value is not smaller than the function target value over all trials and #𝑠𝑢𝑐𝑐 is the number of successful trials. The post-processed data of each document has been linked as well as the comparison between the 4 optimizers.

In table 1, we show the bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension for 50 targets in 10[-8..2] for all functions and subgroups in 5-D and 20-D. The “best 2009” line corresponds to the best ERT observed during BBOB 2009 for each single target.



The empirical cumulative distribution (ECDF) above shows that for 5-dimensions, aside from losing to DEAE and PSO at separable functions, BIPOP clearly performs better than the other optimizers by consistently getting the best results. However, DEAE was able to consistently get second place, performing almost identically to BIPOP. The only exception to this is the weakly structured multimodal functions where PSO was able to beat out DEAE to become to get the second-best results. The gap between second and third place is quite significant in some cases like moderate functions and ill conditioned functions followed by a slightly smaller gap between third and last place. GA was consistently the worst optimizer of the 4.

For the 20-dimensions ECDF, BIPOP was able to get the best results out of all the optimizers in all types of functions. This time, BIPOP was able to beat out the other optimizers decisively, creating a noticeable gap between first and second place. However, the second and third place optimizers perform nearly identically. With the exception of multimodal functions, the second and third place optimizers alternate between DEAE and PSO. Finally, GA was also shown to almost exclusively occupy last place.

From the post-processed data, the phenomenal performance by BIPOP can be attributed to the many modifications of the CMA-ES algorithm, which are considered to be among the best in the field of black-box optimization. For PSO, its poor showing could be blamed on the sensitivity of the optimizer to hyperparameter tuning which has proved to be challenging. With a better choice of parameters, PSO could perhaps have performed substantially better in the experiments as the hyperparameters can seriously affect the final results. However, in the case of BIPOP, no tedious parameter tuning is required which makes it much convenient to use than PSO which is a massive point in its favour.

**Conclusion**

This report documents the experimentation of selected stochastic optimisers for the single-objective continuous optimisation problems, namely the BIPOP, DEAE, GA and PSO optimisers. In order to evaluate their performance, the BBOB-2010 benchmark test functions were utilised and tested against all selected optimisers. In this report, we organised the results of the optimisers after running the benchmark for dimensions 2,3,5,10 and 20 then compared and analysed the difference in their results. Based on the performance reported, it is clear that the BIPOP optimiser has the best performance among them, with GA being the worst performing optimiser. BIPOP has been shown to consistently solve most functions (f1-f19), obtaining the best results while PSO performed excellently in multi-modal functions with weak global optimum (f20-f24). DEAE was a close contender with BIPOP for best optimizer at lower dimensions but lost out to it at higher dimensions. Based on the obtained results, we concluded that the best optimiser out of the 4, is the BIPOP.

**References**

[1] K. Sörensen and F. Glover. Metaheuristics. In S.I. Gass and M. Fu, editors, Encyclopedia of Operations Research and Management Science, New York, To appear. Springer.

[2] Back, T., Hoffmeister, F., & Schwefel, H.-P. (1992). Applications of evolutionary algorithms. Report of the Systems Analysis Research Group SYS-2/92, University of Dortmund, Department of Computer Science.

[3] Alatas, B., (2011). Uniform Big Bang–Chaotic Big Crunch Optimization, Communication Nonlinear Science and Numerical Simulation, 16(9), 3696-3703.

[4] Rashedi, E., Nezamabadi, H., Saryazdi, S., (2009). Gravitational Search Algorithm, Information Sciences, 179, 2232-2248.

[5] Kar AK (2016) Bio-inspired computing—a review of algorithms and scope of applications. Expert Syst Appl 59:20–32

[6] Hansen, N. and A. Ostermeier (2001). Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation 9(2), 159–195.

[7] Hansen, N. (2005). References to CMA-ES applications. <http://www.bionik.tu-berlin.de/user/niko/cmaapplications.pdf>.

[8] V. L. Huang, S. Z. Zhao, R. Mallipeddi, P. N. Suganthan, "Multi-objective optimization using self-adaptive differential evolution algorithm", Evolutionary Computation 2009. CEC '09. IEEE Congress on, pp. 190-194, 2009.

[9] Fouskakis, D., & Draper, D. (2002). Stochastic optimization: a review. International Statistical Review, 70(3), pp. 315-349.

[10] Mathew, T. V. (2012). Genetic algorithm. Report submitted at IIT Bombay.

[11] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In Proc. of IEEE International Conference on Neural Networks, volume 4, pages 1942-1948, 1995.

[12] Petr Pošík and Václav Klemš. 2012. Benchmarking the differential evolution with adaptive encoding on noiseless functions. In Proceedings of the 14th annual conference companion on Genetic and evolutionary computation. ACM, 189--196.

[13] Caraffini, F., & Neri, F. (2018). Rotation invariance and rotated problems: An experimental study on differential evolution. In International Conference on the Applications of Evolutionary Computation (pp. 597-614). Springer, Cham.

[14] Klemš, V. Differential evolution with adaptive encoding (Doctoral dissertation, Master’s thesis, Czech Technical University in Prague, 2011. Available online, https://cyber.felk.cvut.cz/theses/papers/177.pdf) [Accessed 15 April 2022].

[15] El-Abd, M., and Kamel, M. S. (2009). Black-Box Optimization Benchmarking for Noiseless Function Testbed using Particle Swarm Optimization. In Proceedings of the 2009 Genetic and Evolutionary Computation Conference. ACM, New York, NY, 2009, 2269–2273