COMP2024 (G52AIM)
ARTIFICIAL INTELLIGENCE METHODS (20 CR)
COURSEWORK REPORT - GROUP 16

Leader - Muhammad Salman Malik (20120791), Kugan Reddy Nadaraja (20006742), Wen Jye Chai (20113681), Muhammad Fayyadh Azrin (20113641), Pang Cheng Han (20113642)

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

Literature Review of Optimizers

Particle Swarm Optimization Algorithm

Particle Swarm Optimization (PSO), is one of many problem solving algorithms which works in a way that it optimizes a problem by iteration, where it iteratively improves a "candidate solution" depending on the measure of quality. It is considered a stochastic population-based optimization method which was proposed by Kennedy and Eberhart (1995) and is becoming more and more popular due to its simplistic approach towards problem solving. A self-organizing algorithm which exists in swarm-intelligence(SI) plays an important role where it is defined as "a process where global order or coordination arises out of the local interactions between the components of an initially disordered system" (Yudong Zhang, 2015), where the process is spontaneous. Therefore agents outside or inside of the system do not administer it.

Bonabeau et al. explained self-organization into three simple parts. The first part is that it's a strong dynamical nonlinearity, which involves two main types of feedback, positive and negative. The positive feedback will aid in advertising the advantageous structure, whilst the negative feedback helps to stabilize the collective pattern counteracting with the positive feedback. The second part is that PSO includes a good balance between exploration and exploitation. Finally, it has multiple interactions where the agents in the swarm gather data from neighbouring agents which will lead to a fast spread of information throughout the whole network. The searching process of PSO via swarm particles updates each other in each iteration and in order to find the best solution, each particle will go towards the direction of its formerly best (*pbest*) position and global best (*gbest*) position in the swarm.

$$pbest\,(i,t) = \arg\min_{k=1,\dots,t} \left[f\left(P_i\,(k)\right) \right], \quad i \in \{1,2,\dots,N_P\},$$
$$gbest\,(t) = \arg\min_{\substack{i=1,\dots,N_P \\ k=1,\dots,t}} \left[f\left(P_i\,(k)\right) \right], \qquad (1)$$

where i being particle index, Np the total number of particles, t current number of iterations, f the fitness function, and P denoting the position. The velocity V and position P of the particles will be updated via the following equations:

$$V_i\,(t+1) = \omega V_i\,(t) + c_1 r_1 \left(pbest\,(i,t) - P_i\,(t)\right) + c_2 r_2 \left(gbest\,(t) - P_i\,(t)\right), \qquad (2)$$

$$P_i\,(t+1) = P_i\,(t) + V_i\,(t+1), \qquad (3)$$

where V being the velocity, w inertia weight (which is used to stabilize the global exploration and local exploitation) $r_1$and $r_2$are uniformly distributed random variables in the range [0,1], and $c_1$and $c_2$denoting positive constant parameters also known as "acceleration coefficients".

Commonly the velocity parameter would be assigned with an upper bound, this is due to particles flying out from the supposedly search space, hence why velocity clamping is needed for this reason. An alternative from velocity clamping would be the constriction coefficient strategy, which was introduced by Clerc and Kennedy , where the velocities will be constricted as well. Based on formula (2) above, the "inertia" denotes previous velocity,

for the purpose of providing sufficient momentum for the particles to move and roam around the search space. The "cognitive" component represents individual particles thinking of each particle, resulting in the particles to advance in the direction of their own best position that they had discovered so far, and the collaborative effect of each particle communicating with one another to discover the best global optimal solution can be identified as "cooperation".

Like any other SI-based optimizers, the PSO algorithm also has its drawbacks such as sensitivity to parameters, slow convergence, and also needing a high amount of resources to use. The problems mentioned above can happen due to the fact that the PSO algorithm is not able to properly handle the relationship between exploitation(local) and exploration(global), thus resulting it to converge to local minimum.

The following literature has concluded that PSO is a worthy candidate to consider, especially when dealing with single objective continuous optimization problems.

BIPOP-CMAES

In the section below, we will be reviewing a different evolutionary algorithm approach known as covariance matrix adaptation evolutionary algorithm. In this algorithm, new search points are generated using a multivariate normal search distribution with a covariance matrix that is not restricted to a priori. The changes in distribution are deterministically linked to the object parameter variations.(Hansen, 2006) To put it another way, the search distribution is tailored to suit the contour lines of the objective function that needs to be optimised. Before we delve into the logic behind CMA-ES, we should first discuss the multivariate normal distribution. A normal distribution $N(m, C)$ is often defined by the its mean, $m \in \mathbb{R}^n$ and its covariance matrix, $C \in \mathbb{R}^{n\,x\,n}$. The surface of the equidensity contour of the distribution can be identified as ellipsoids that are centered as its mean.(Hansen, 2006) Hence, we can conclude that it to be an ellipsoids distribution where the eigenvector of Creflects the directions of the principal axes of the ellipsoids and its corresponding eigenvalues are the squared relative length of the axes. The eigendecomposition of Ccan be denoted as $V\Lambda V^T$where Λ is the diagonal matrix of eigenvalues and V is the matrix with eigenvectors of Calong its columns. The normal distribution can then be defined is such way:

$$N(m, C) \sim m + V\Lambda N(0, I) \sim m + VN(0, \Lambda)$$

where:

$\sim$, denotes the same distribution on the left and right side.

I , denotes the identity matrix.

m , denotes the mean value of the distribution.

C , denotes the covariance matrix of the distribution.

In this evolution strategy, the population of the offsprings is produced by sampling a multivariate normal distribution :

$$x_k^{(g+1)} \sim N\left(m^{(g)}, \left(\sigma^{(g)}\right)^2 C^{(g)}\right) \quad for\ k = 1, \dots, \lambda$$

where:

$N\left(m^{(g)}, \left(\sigma^{(g)}\right)^2 C^{(g)}\right) \sim m^{(g)} + V^{(g)}\sigma^{(g)}N(0, \Lambda)$, is the multivariate normal search distribution.

g , denotes the generation number of the population.

$x_k^{(g+1)}$, denotes the k-th search points from generation $g + 1$.

$m^{(g)}$, denotes the mean value of the search distribution at generation g.

$\sigma^{(g)}$, denotes the standard deviation and step size at generation g.

$C^{(g)}$, denotes the covariance matrix at generation g.

In mean equation, we implements truncation selection by choosing $\mu < \lambda$ out of λ offsprings. We also implement different weights for each selected point. In other words, we assumed that the higher rank selected point should lead to better results and thus should be allocated greater weightage than lower rank selected points. The measure to choose μ_{eff} known as variance effective selection mass. A reasonable setting would be the weights of each point, $w_i \propto \mu - i + 1$,and $\mu \approx \lambda/2$. Next, we shall discuss the covariance matrix, in order to achieve fast

search, the population size of our sample must be small and hence it is not reliable to obtain a good covariance matrix when the data size is small. One way to resolve this conflict is to include information from the previous generations and the covariance matrix is improved and modified. This covariance matrix update is called as rank-1-update. (Hansen, 2006) We used the evolution path to exploit the correlations between consecutive steps and update our covariance matrix. We also used rank-μ- update to retrieve the information within the population of one generation and update our covariance matrix.

Lastly, we will be looking at the step size which will be used to control our CMA update of covariance matrix. We want the step size to be increased when the single steps are correlated and pointing to similar directions. Nevertheless, we want the step size to be decreased when single steps are anti-correlated as they cancel each other out. We implement a method called cumulative path length control whereby the evolution path is used to determine the step size. The length of the evolution path is an intuitive well validated measure for overall step length. In our assignment, we implemented a variant type of CMA-ES, namely BI-population CMA-ES. After a first single run with default population size, the strategy will be restarted using one of the two regimes that are evaluating by calculating the budget of its function evaluation. (Loshchilov, Schoenauer, & Sèbag, 2013) The regimes with smaller budgets will be used to restart the strategy. Under the first regime the population size is doubled in each restart with some fixed initial step-size. On The other hand, in the second regime, strategy is restarted with some small population size and step size.

Differential Evolution Algorithm

Differential evolution (DE) is a population-based optimization algorithm that employs an evolutionary computation approach to optimize a problem by iteratively attempting to evolve a candidate solution in terms of the given quality metric. Initially proposed in 1995 by Storn and Price in 1995, DE does not require many control variables but showcases good performance in convergence.

DE is made up of three fundamental steps. (1) A population of N solutions [$x = (x_1, x_2, \ldots, x_m)$] with a large enough population in the m-dimensional space is generated and randomly distributed throughout the function domain and the solutions are evaluated by finding f(x). (2) Current population is replaced with a population with better fitness values. (3) The replacements are repeated until either acceptable results are obtained or the termination conditions are met. In order to achieve step 2, DE creates a candidate solution to replace each of the population's existing solutions. A crossover between the current solution and other randomly chosen solutions from the existing population yields the candidate solution. The crossover itself is probabilistic in nature. If the candidate solution is a better fit than the current solution, it replaces it; otherwise, the current solution remains and moves on to the next iteration.

The optimization problem is only used as a quality indicator when a candidate solution is available and the gradient is not needed. Initialization, fitness assessment, mutation, crossover and selection are the key operations in DE that mimic the natural evolution process. The distinction between DE and PSO is in the process of coming up with new solutions where DE generates new solutions by combining multiple solutions with the candidate solution. Also, in DE, a new solution is chosen only if it has a higher fitness, with the population's average fitness being the same or improving in every iteration. Any enhancement to the solution is automatically eligible for selection as a mutant vector for the next target vector.

Ability to perform simple vector subtractions to generate a random direction is one of DE's advantages. In addition, in solutions that have not yet converged, further variation in the population can be achieved, resulting in a more diverse search across solution space. While DE outperforms some evolutionary algorithms in terms of competitive efficiency, it is still highly dependent on the settings of its control parameters such as the crossover rate, population size (NP), mutation and crossover strategy as well as the scaling factor.

Controlled Experimentation of Optimizers

The following constraints were used across all optimizers used in the experimentation as a way to standardize them:

1. Budget constraint of $50k \times dim$ evaluations (5k was used for the competition submission)
2. Experiments were run for 15 instances
3. Fixed random seed of 0 was used

4. Experiments were carried out on dimensions = [2, 3, 5, 10, 20, 40]
5. Experiments were tested against the same set of 24 benchmark functions
6. A precision value of 1e-8 was used

The following graphs were obtained by using the post-processing tools available from the COCO framework: https://github.com/numbbo/coco.

Individual Parameter Settings

The original *gbest* model of the Particle Swarm Optimization(PSO) was used, with a design choice of absorbing boundaries to handle any particles leaving the search space (El-Abd, Mohammed, 2009), function evaluations for each dimension. For the parameters of the algorithm, if not explicitly stated, the default values used for the coefficients w, c_1, c_2 and the population size are *0.792, 1.4944, 1.4944* and *50* respectively.

As for Differential Evolution(DE), the default values used for the parameters scaling factor for each generation, binomial crossover rate, mutation strategy, crossover strategy and the population size are F ~ U(0.5, 1), CR = 0.5, 'best', 'bin' and NP = 5 dimension.

In BIPOP-CMAES, the initial default setting for population size is set based on the dimensions with the standard deviation, σ is 2.

Experiment to determine how population size affects performance in PSO

Due to the nature of BIPOP-CMAES and DE, not much tweaking of parameters was required, unlike PSO whose performance depends on the value of its parameters such as population size. The following experiments for PSO were conducted to find out how different parameter values will affect the algorithm, with this information we can use the best values from each experiment as the base values for comparisons with other optimizers. The experiments were conducted with population sizes of *10, 40, 50, 100, 200 and 100×DIM* for dimensions = [2, 3, 5, 10, 20].

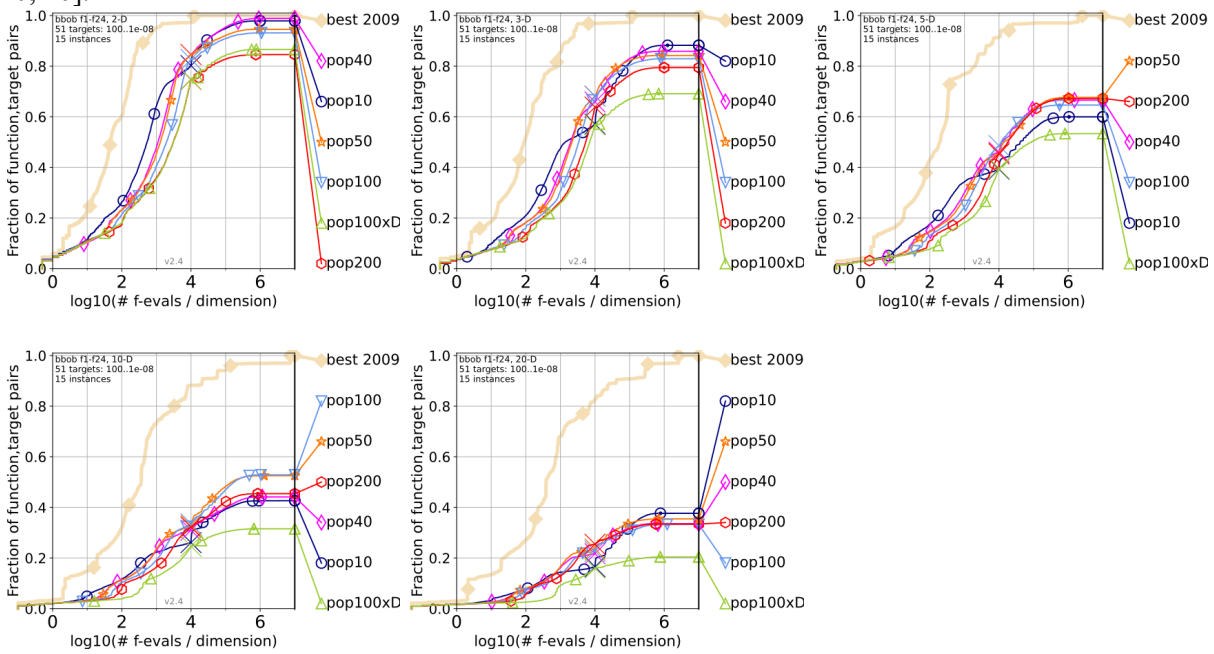

Figure 1: Runtime distributions (ECDFs) over all targets for different population sizes on different dimensions.

The ECDFs show that in general, population sizes 10, 40 and 50 perform the best out of all the population sizes. In most cases, a population size of 100*dim resulted in the least performance. This shows that the increase in population size does not necessarily correspond to an increase in the proportion of target pairs reached. A particular pattern that the PSO algorithm follows according to observations made from the ECDFs is that after ~10^5 function

evaluations for any dimension, the percentage of target pairs reached will tend to become constant. This may suggest that increasing the number of function evaluations might not necessarily mean that it could reach more target pairs, as many of the particles might get stuck at a local minima or maxima after a particular number of function evaluations. This phenomenon can also be used to explain why using a larger population size doesn't result in better efficiency of the optimizer.

Runtime of different algorithms against dimension

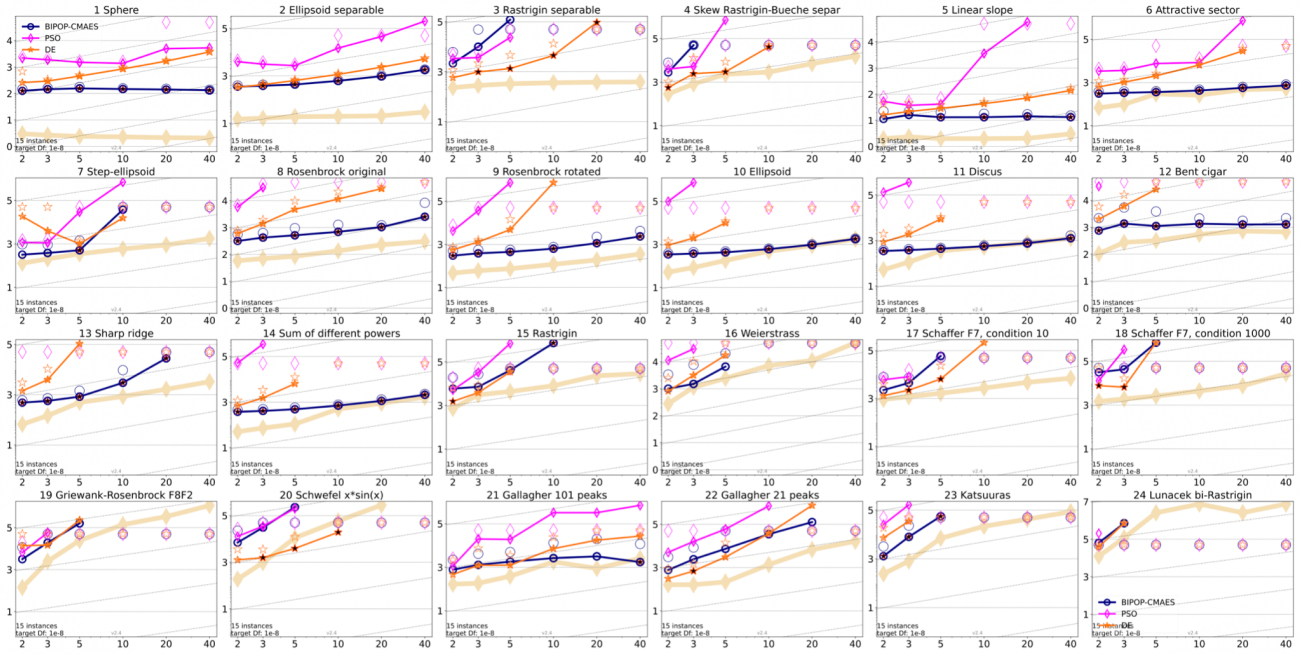

Fig. 2. **Expected running time (ERT in number of f−evaluations as $\log_{10}$ value), divided by dimension for ε = 1e-8 versus dimension. Legend: ○: BIPOP-CMAES, ◇: DE, ⋆: PSO**
. Statistically significant results (compared to other algorithms) with p < 0.01, i.e. 1% significance, are indicated by black stars.

The analysis and observations are tabulated as follows:

PSO
• PSO struggles with reaching f_{target} for dimensions > 3. • In fact PSO only managed to reach the f_{target} values for dimension 40 for only 3 benchmarks: *Sphere* (Fig. 2-1). *Ellipsoid Separable* (Fig. 2-2),. *Gallagher 101 Peaks* (Fig. 2-21) • PSO only managed to beat BIPOP-CMAES once in the Rastrigin Separable benchmark in dimensions 2 & 3. • Both in the Ellipsoid as well as Linear Slope functions, the number of function evaluations decreased from dimensions 2 until 5, which suggests that the hyperparameters could have been tuned in a better way. • Completely failed on Sharp ridge (fig. 2-13).
DE
• Consistently defeated PSO across all benchmark functions in all the dimensions except for the Step-ellipsoid benchmark in dimensions 2 & 3. • DE, despite being surpassed by BIPOP-CMAES on the majority of the benchmarks in terms of scalability, did manage to beat BIPOP-CMAES on Rastrigin Separable, having statistically better results in dimensions = [3, 5, 10, 20]. • Achieved significant results in almost all dimensions in 5 different benchmarks: *Rastrigin separable* (Fig. 2-3), *Skew Rastrigin-Bueche separ* (Fig. 2-4), *Schaffer* functions (Fig. 2-17 and 2-18), as well as in *Schwefel x.sin(x)* (Fig. 2-20).
BIPOP-CMAES

- BIPOP-CMAES considerably outperformed the other 2 algorithms in general, reaching f_{target} 11 times successfully even in 40-D, in contrast with DE and PSO, both of which only reached it 3 times. We can conclude from this that the efficiency of DE and PSO really suffers in high dimensions such as 20 and 40.
- In some cases, such as Fig 6, 10, 11 and 14, the *Attractive sector, Ellipsoid, Discus and Sum of different powers* respectively, the ERT was extremely close to that of the reference algorithm *best2009's* solutions in the higher dimensions.

** There were also cases where the results of 2 of the algorithms were almost identical to each other. E.g. 1. BIPOP-CMAES and DE on *Lunacek bi-Rastrigin*, (Fig. 2-24) and 2. BIPOP-CMAES and PSO on *Schwefel x.sin(x)* (Fig. 2-20).

ECDFs of optimizer results on different benchmarks for dimension = 5

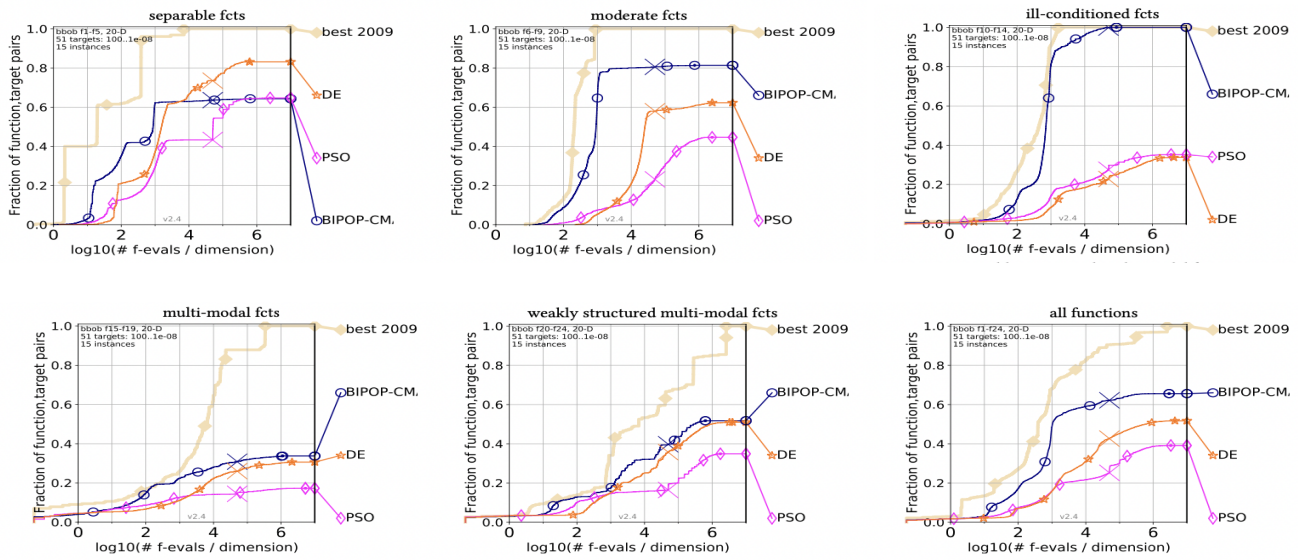

Fig. 3. Bootstrapped empirical cumulative distribution (ECDF) of the number of objective function evaluation divided by the dimension (FEvals/DIM) for 51 targets with target precision in $10^{[-8..2]}$ for all functions and subgroups in 5-D. As reference algorithm, the best algorithm from BBOB 2009 is shown as light thick line with diamond markers.

For the ill-conditioned functions, BIPOP-CMAES significantly outperforms DE and PSO, almost reaching the optimization efficiency of best 2009. PSO is clearly the worst-performing algorithm across all types of function groups. There isn't a considerable difference between BIPOP-CMAES and DE in the case of *multi-modal* and *weakly structured multi-modal functions*, while there is a clear distinction, with no overlapping, between the graphs for BIPOP-CMAES and DE for the *moderate functions*.

Statistical Analysis:

In order to calculate statistical significance of the results, the Wilcoxon Rank-Sum test was performed for each of the optimizers in dim=5, which is indicated by a star next to it. The expected runtime ratio is calculated by dividing the algorithm expected runtime by its respective best ERT measured during BBOB-2010 in the same dimension. The different target Δf -values are shown in the top row. #succ reflects the number of trials that reached the (final) target $f$opt $+10^{-8}$. If the target in the last column was never met, the median number of conducted function evaluations is also given in italics. with Bonferroni correction by the number of functions. Best results are printed in bold.

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f1	11	12	12	12	12	12	12	15/15
BIPOP-C	**3.2**(3)	**9.2**(3)$^{\star 2}$	**16**(2)$^{\star 4}$	**22**(5)$^{\star 4}$	**29**(3)$^{\star 4}$	**42**(5)$^{\star 4}$	**58**(6)$^{\star 4}$	15/15
PSO	3.4(2)	25(13)	76(31)	149(26)	223(44)	380(41)	543(47)	15/15
DE	4.8(8)	24(8)	43(10)	63(14)	84(11)	127(12)	167(13)	15/15
f2	83	87	88	89	90	92	94	15/15
BIPOP-C	15(4)	17(2)	18(3)	**19**(2)	**20**(2)	**21**(1)$^{\star 4}$	**22**(2)$^{\star 4}$	15/15
PSO	44(12)	53(10)	63(8)	75(9)	86(11)	106(10)	130(11)	15/15
DE	**11**(3)*	**14**(1)$^{\star 2}$	**16**(0.9)	19(1)	22(2)	27(2)	32(3)	15/15
f3	716	1622	1637	1642	1646	1650	1654	15/15
BIPOP-C	**1.1**(1.0)	34(31)	372(316)	371(582)	370(325)	369(204)	368(603)	5/15
PSO	3.2(2)	17(40)	67(77)	68(115)	68(155)	69(89)	71(152)	11/15
DE	1.2(0.3)	**1.2**(0.3)$^{\star 2}$	**3.1**(5)$^{\star 2}$	**3.3**(4)$^{\star 2}$	**3.4**(5)$^{\star 2}$	**3.7**(3)$^{\star 2}$	**4.0**(3)$^{\star 3}$	15/15
f4	809	1633	1688	1758	1817	1886	1903	15/15
BIPOP-C	2.4(3)	478(358)	∞	∞	∞	∞	∞ *3e5*	0/15
PSO	3.4(2)	114(194)	2116(1962)	2032(3875)	1967(2133)	1897(2021)	1881(1248)	1/15
DE	**1.2**(0.4)	**2.8**(2)$^{\star 4}$	**7.6**(11)$^{\star 4}$	**7.5**(10)$^{\star 4}$	**7.4**(7)$^{\star 4}$	**7.4**(9)$^{\star 4}$	**7.6**(9)$^{\star 4}$	15/15
f5	10	10	10	10	10	10	10	15/15
BIPOP-C	**4.8**(2)$^{\star 2}$	**6.4**(2)$^{\star 3}$	**6.5**(2)$^{\star 4}$	**6.6**(2)$^{\star 4}$	**6.6**(1)$^{\star 4}$	**6.6**(2)$^{\star 4}$	**6.6**(2)$^{\star 4}$	15/15
PSO	12(6)	19(10)	20(12)	21(10)	21(12)	21(10)	21(12)	15/15
DE	8.5(3)	14(5)	14(4)	14(2)	14(3)	14(4)	14(2)	15/15
f6	114	214	281	404	580	1038	1332	15/15
BIPOP-C	**2.0**(1)	**1.9**(0.8)$^{\star 4}$	**2.1**(0.5)$^{\star 4}$	**1.9**(0.4)$^{\star 4}$	**1.6**(0.2)$^{\star 4}$	**1.2**(0.2)$^{\star 4}$	**1.2**(0.1)$^{\star 4}$	15/15
PSO	3.7(3)	91(294)	75(225)	57(3)	43(1)	29(63)	26(48)	14/15
DE	6.3(3)	7.3(1)	8.9(2)	8.7(2)	7.9(2)	6.4(1)	6.8(0.8)	15/15
f7	24	324	1171	1451	1572	1572	1597	15/15
BIPOP-C	**5.6**(3)	**1.5**(2)$^{\star 2}$	**1.5**(2)	**1.3**(1)$^{\star 2}$	**1.5**(1)*	**1.5**(2)*	**1.5**(1)*	15/15
PSO	8.8(4)	4.3(3)	86(229)	74(129)	93(170)	93(119)	92(327)	10/15
DE	12(7)	3.1(1)	1.9(0.8)	2.7(0.7)	2.7(0.7)	2.7(0.6)	2.9(0.8)	15/15
f8	73	273	336	372	391	410	422	15/15
BIPOP-C	**3.4**(0.9)$^{\star 3}$	**4.6**(4)$^{\star 2}$	**5.2**(1)$^{\star 3}$	**5.4**(1)$^{\star 4}$	**5.4**(3)$^{\star 4}$	**5.7**(2)$^{\star 4}$	**5.9**(3)$^{\star 4}$	15/15
PSO	17(12)	92(19)	153(237)	287(237)	466(241)	9093(8841)	∞ *2e5*	0/15
DE	10(3)	16(27)	23(6)	28(24)	32(20)	41(20)	51(4)	15/15
f9	35	127	214	263	300	335	369	15/15
BIPOP-C	**6.5**(4)$^{\star 4}$	**7.3**(4)$^{\star 4}$	**6.6**(2)$^{\star 4}$	**6.4**(2)$^{\star 4}$	**6.1**(2)$^{\star 4}$	**6.0**(1)$^{\star 4}$	**5.9**(1)$^{\star 4}$	15/15
PSO	36(12)	761(996)	609(608)	842(1010)	1492(1439)	5339(7090)	9850(1e4)	1/15
DE	22(7)	48(11)	42(10)	43(9)	44(9)	52(45)	60(44)	15/15
f10	349	500	574	607	626	829	880	15/15
BIPOP-C	**3.3**(0.8)$^{\star 4}$	**2.9**(0.3)$^{\star 4}$	**2.8**(0.2)$^{\star 4}$	**2.8**(0.2)$^{\star 4}$	**2.8**(0.2)$^{\star 4}$	**2.3**(0.1)$^{\star 4}$	**2.4**(0.2)$^{\star 4}$	15/15
PSO	777(1097)	1717(2889)	∞	∞	∞	∞	∞ *2e5*	0/15
DE	28(9)	27(7)	32(7)	38(9)	43(6)	44(4)	51(8)	15/15
f11	143	202	763	977	1177	1467	1673	15/15
BIPOP-C	**8.5**(3)$^{\star 2}$	**7.5**(1)$^{\star 4}$	**2.2**(0.2)$^{\star 4}$	**1.8**(0.1)$^{\star 4}$	**1.6**(0.1)$^{\star 4}$	**1.4**(0.1)$^{\star 4}$	**1.3**(0.1)$^{\star 4}$	15/15
PSO	125(175)	269(222)	140(85)	175(65)	309(280)	2528(2769)	∞ *2e5*	0/15
DE	34(14)	46(15)	18(5)	19(4)	19(3)	21(4)	24(6)	15/15
f12	108	268	371	413	461	1303	1494	15/15
BIPOP-C	**11**(5)$^{\star 4}$	**7.6**(8)$^{\star 3}$	**7.1**(8)$^{\star 4}$	**7.5**(7)$^{\star 4}$	**8.4**(10)$^{\star 4}$	**3.6**(4)$^{\star 4}$	**3.6**(4)$^{\star 4}$	15/15
PSO	619(587)	2585(4665)	∞	∞	∞	∞	∞ *2e5*	0/15
DE	85(65)	84(88)	105(105)	125(106)	133(74)	65(39)	69(41)	14/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f13	132	195	250	319	1310	1752	2255	15/15
BIPOP-C	**4.2**(2)$^{\star 4}$	**4.9**(1)$^{\star 4}$	**5.3**(2)$^{\star 4}$	**5.1**(2)$^{\star 4}$	**1.4**(0.4)$^{\star 4}$	**1.8**(0.9)$^{\star 4}$	**1.6**(0.7)$^{\star 4}$	15/15
PSO	495(476)	3549(4808)	1.4e4(8500)	∞	∞	∞	∞ *2e5*	0/15
DE	15(4)	26(7)	39(11)	51(13)	19(5)	27(5)	48(30)	6/15
f14	10	41	58	90	139	251	476	15/15
BIPOP-C	1.4(2)	**2.8**(1)$^{\star 3}$	**3.7**(0.8)$^{\star 4}$	**4.0**(0.9)$^{\star 4}$	**4.7**(1.0)$^{\star 4}$	**5.5**(0.8)$^{\star 4}$	**4.3**(0.4)$^{\star 4}$	15/15
PSO	**1.3**(1)	6.4(3)	16(6)	26(7)	34(7)	299(303)	∞ *2e5*	0/15
DE	1.9(3)	6.8(4)	11(3)	12(2)	15(4)	39(10)	48(9)	15/15
f15	511	9310	19369	19743	20073	20769	21359	14/15
BIPOP-C	**1.8**(2)$^{\star 2}$	**4.0**(4)	11(10)	10(12)	10(12)	10(27)	10(10)	10/15
PSO	8.7(4)	113(117)	182(174)	179(453)	176(190)	170(190)	166(167)	1/15
DE	6.5(4)	4.9(3)	**8.7**(10)	**8.6**(8)	**8.5**(5)	**8.3**(7)	**8.1**(10)	12/15
f16	120	612	2662	10163	10449	11644	12095	15/15
BIPOP-C	2.1(2)	**5.4**(9)	**2.5**(3)*	**1.2**(0.8)$^{\star 2}$	**1.9**(2)	**1.9**(2)*	**2.5**(2)	15/15
PSO	**2.1**(2)	6.3(3)	78(212)	100(216)	98(133)	93(79)	90(51)	0/15
DE	5.6(4)	31(26)	14(17)	6.7(8)	6.9(7)	6.4(6)	7.2(4)	14/15
f17	5.0	215	899	2861	3669	6351	7934	15/15
BIPOP-C	4.0(5)	**2.1**(0.4)$^{\star 2}$	**2.0**(4)	1.5(3)	2.6(4)	12(13)	38(42)	9/15
PSO	**2.7**(6)	2.9(2)	47(139)	60(110)	84(110)	160(246)	∞ *2e5*	0/15
DE	5.4(4)	3.1(1)	2.4(1)	**1.5**(0.7)	**1.6**(0.6)	**1.9**(1)	**2.4**(2)$^{\star 4}$	15/15
f18	103	378	3968	8451	9280	10905	12469	15/15
BIPOP-C	**1**(0.7)$^{\star 2}$	**1.0**(0.4)$^{\star 4}$	**1.3**(0.8)	**2.7**(1)	7.3(9)	56(96)	140(275)	1/15
PSO	2.1(0.8)	54(3)	59(96)	119(74)	378(290)	∞	∞ *2e5*	0/15
DE	2.7(2)	5.2(3)	3.6(4)	2.9(4)	**3.8**(4)	**10**(11)	**89**(54)	1/15
f19	1	1	242	1.0e5	1.2e5	1.2e5	1.2e5	15/15
BIPOP-C	**19**(20)	**1370**(936)	**301**(312)	**5.8**(10)	**6.7**(4)	**6.6**(6)	**6.6**(12)	4/15
PSO	31(32)	5512(9011)	2717(2943)	∞	∞	∞	∞ *2e5*	0/15
DE	34(25)	4436(2612)	675(923)	8.0(5)	9.5(11)	9.5(4)	9.4(10)	3/15
f20	16	851	38111	51362	54470	54861	55313	14/15
BIPOP-C	**2.8**(2)	13(12)	30(41)	22(33)	21(56)	21(26)	21(26)	3/15
PSO	5.5(5)	3.9(3)	26(35)	20(38)	19(29)	18(16)	18(29)	3/15
DE	8.3(5)	**1.9**(1)$^{\star 2}$	**0.46**(0.6)*	**0.35**(0.4)*	**0.33**(0.2)*	**0.34**(0.4)$^{\star 2}$	**0.35**(0.4)$^{\star 2}$	15/15
f21	41	1157	1674	1692	1705	1729	1757	14/15
BIPOP-C	4.1(10)	4.5(7)	5.2(7)	5.2(6)	5.2(7)	5.2(6)	5.1(6)	15/15
PSO	**2.9**(3)	17(2)	55(75)	55(111)	55(38)	55(181)	55(107)	11/15
DE	3.7(3)	**2.5**(5)	**2.5**(3)	**2.7**(4)	**2.8**(3)	**3.1**(2)	**3.4**(3)	15/15
f22	71	386	938	980	1008	1040	1068	14/15
BIPOP-C	8.1(16)	17(28)	39(29)	37(86)	36(30)	35(87)	34(97)	14/15
PSO	11(31)	164(324)	306(399)	294(255)	287(311)	281(362)	278(236)	7/15
DE	**4.7**(4)	**4.7**(3)	**11**(26)	**11**(11)	**11**(3)	**12**(11)	**14**(24)	15/15
f23	3.0	518	14249	27890	31654	33030	34256	15/15
BIPOP-C	1.9(3)	**11**(8)	**10**(11)*	**10**(9)$^{\star 2}$	**8.8**(13)$^{\star 2}$	**8.4**(9)$^{\star 2}$	**8.2**(6)$^{\star 2}$	10/15
PSO	3.1(3)	19(21)	43(40)	132(65)	∞	∞	∞ *2e5*	0/15
DE	**1.3**(1)	53(53)	127(263)	65(137)	57(75)	112(87)	∞ *2e5*	0/15
f24	1622	2.2e5	6.4e6	9.6e6	9.6e6	1.3e7	1.3e7	3/15
BIPOP-C	**1.4**(1)$^{\star 2}$	**2.7**(2)	∞	∞	∞	∞	∞ *3e5*	0/15
PSO	11(5)	∞	∞	∞	∞	∞	∞ *2e5*	0/15
DE	9.3(7)	3.7(8)	**0.57**(0.5)	∞	∞	∞	∞ *2e5*	0/15

Fig. 4. Expected Running Time (ERT) ratios are calculated by dividing ERT by its respective best ERT obtained.

For most of the precision values in f1-f14, BIPOP-CMAES had significant results, with $p = 0.05$ or $p = 10^{-k}$ when the number k following the star is larger than 1, compared to the other 2 algorithms, with the exception of f3 and f4, where DE had significant results. PSO had the worst performance out of all the algorithms in terms of reaching the target value, having either 0 or 1 successful trials out of 15 in 14 of the benchmarks compared to 3 for both BIPOP-CMAES and DE. Based on the previous tests, we can say that the 2 most powerful algorithms out of the chosen ones are BIPOP-CMAES and DE. Therefore, we performed further tests to draw comparisons between these 2 algorithms.

Ratio of function evaluations for BIPOP-CMAES against DE per function

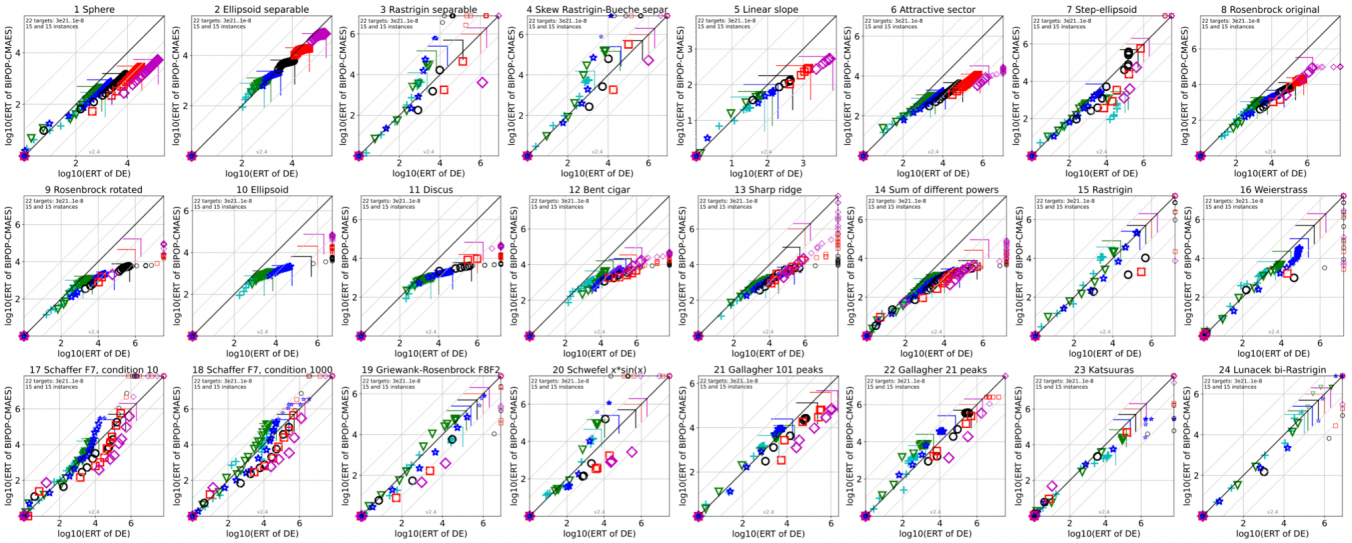

Fig. 5. ERT in $\log_{10}$ of number of function evaluations) of BIPOP-CMAES (y−axis) versus DE (x−axis) for 22 target values $\Delta f \in [3 \times 10^{21}, 10^{-8}]$ in each dimension on functions $f_1 - f_{24}$. Legend: Markers represent dimension: 2:+, 3:▽, 5:∗, 10:°, 20:□, 40:◇.

We can interpret the above plots by the following heuristic: if the markers are closer to the right or bottom edges, then that would indicate BIPOP-CMAES required fewer evaluations and thus, was more efficient. On the other hand, if the markers were closer to the top or left edges, then DE was the more efficient algorithm in that particular dimension. From f5-f14, which are the moderate as well as ill-conditioned functions, almost all the results are skewed towards the right side of the diagonal for higher dimensions suggesting that BIPOP-CMAES was more efficient. For the Ellipsoid separable function, *Fig.4-2,* both all the markers seem to align along the diagonal, suggesting that they both required similar function evaluations in reaching the target across the different dimensions.

Another observation that can be derived is for the Schaffer functions, where BIPOP-CMAES appeared to perform better compared to DE in higher dimensions while DE performed better in dimensions 5 and lower. Finally, we can observe that DE didn't manage to reach the target values for functions f9-f16 when the dimension ≥ 10, most of the functions therein falls into the ill-conditioned function category.

As for the competition, the scores obtained by the 3 optimizers chosen were 2390 for BIPOP-CMAES, 298 for PSO and 972 for DE. Therefore, BIPOP-CMAES was chosen as our final optimizer submission for the competition.

References

Abbas, Q., Ahmad, J., & Jabeen, H., (2015). *A Novel Tournament Selection Based Differential Evolution Variant for Continuous Optimization Problems.*

Bonyadi, M. R., & Michalewicz, Z. (2017). *Particle Swarm Optimization for Single Objective Continuous Space Problems: A Review. Evolutionary Computation, 25(1), 1–54. https://doi.org/10.1162/evco_r_00180*

El-Abd, Mohammed, & S. Kamel, Mohamed (2009). *Black-Box Optimization Benchmarking for Noiseless Function Testbed Using Particle Swarm Optimization*.

Georgioudakis, M. & Plevris, V., (2020). *A Comparative Study of Differential Evolution Variants in Constrained Structural Optimization.*

Hansen, N. (2006). *The cma evolution strategy: A comparing review. Towards a New Evolutionary Computation, 75-102. doi:10.1007/11007937_4*

Loshchilov, I., Schoenauer, M., & Sèbag, M. (2013). *Bi-population CMA-ES ALGORITHMS with SURROGATE models and line searches. Proceeding of the Fifteenth Annual Conference Companion on Genetic and Evolutionary Computation Conference Companion - GECCO '13 Companion.* doi:10.1145/2464576.2482696

Zhang, Y., Wang, S., & Ji, G. (2015). A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications. Mathematical Problems in Engineering, 2015, 1–38. https://doi.org/10.1155/2015/931256

Related Link

Github link that compiled the result of our analysis
https://github.com/032310/AIM_Coursework