Progressive Differential Evolution on Clustering Real World Problems

Vincent Berthier

Progressive Differential Evolution on Clustering Real World Problems

Vincent Berthier

TAO (Inria), LRI, UMR 8623 (CNRS - Univ. Paris-Sud)
Bat 660 Claude Shannon Univ. Paris-Sud, 91190 Gif-sur-Yvette, France
Email: {firstname.lastname}@inria.fr

Abstract. In this paper, we assess the performances of Differential Evolution on real-world clustering problems. To improve our results, we introduce Progressive Differential Evolution, a small modification of Differential Evolution which aims at optimizing a small number of parameters (*eg.* one cluster) at the beginning, and incrementally increase the number of optimized parameters.

1 Introduction

While many benchmarks used in the optimisation community to evaluate algorithms are based on purely artificial functions such as [20] and [10], it can only be the first step in what ultimately is aimed at solving real world problems. Some recent initiatives went in that direction (see [8] for example), proposing new ways to assess the performances of optimisation algorithms.

In this paper, by comparing our results on one such benchmark, we (i) show that the Differential Evolution algorithm is very efficient on clustering problems and (ii) propose Progressive Differential Evolution, which starts with a low number of parameters to optimise and gradually increases it.

Section 2 describes the benchmark we used to compare our results to other algorithms and Section 3 validates our approach. Section 4 recalls the Differential Evolution algorithm and the "DE/curr-to-best/1" variant we used while Section 5 introduces Progressive Differential Evolution. In Section 6 we compare our results to the state of the art.

2 Continuous Real-World Representative benchmark

Most of the existing testbeds used to evaluate optimisation algorithm compare their performances on artificial functions, such as the sphere, the ellipsoid or the Rosenbrock function to cite the most notable ones. With the improvements of the algorithms, more complex functions were introduced with some specific properties such as rotation, non separability, multimodality and so on, but ultimately, most testbeds are completely artificial.

While this is by no mean uninteresting, the ultimate goal in optimisation is to solve real world problems. The gap between artificial functions - as complex

as they are - to real world issues seems too large to directly apply what we know. As such, new testbeds, with some real world properties are advisable.

One of such propositions comes from [7] and revolves around clustering problems that have interesting properties to evaluate optimisation algorithms: challenging, scalable, easy to understand and implement, and most of all, their data can - and should - come from real world examples. Each cluster is used as a vector of coordinates in the parameters' space of data, which allows us to use optimisation algorithms on those problems.

The three problems used here are the Iris [6], the Ruspini [15] and the German Town [18] datasets, all of them widely used in the clustering community to evaluate the performances of their own algorithms, and rooted in the real world. More importantly, [12] computed the global optimum for those datasets from two to ten clusters, which allows us to assess the performances of the algorithms. The German Town points are defined in 3D, the Iris ones in 4D and for Ruspini it is in 2D.

Along with a k-means clustering algorithm, [7] studied the performances of three black-box algorithms: CMA-ES [9] (one with standard population size, one with an increased population), Nelder-Mead [13] and Random-Search. One of the conclusions is that even if the k-means algorithm converges very quickly, it is often beaten by CMA-ES (with increased population size) in term of quality of the solution found. Thus, complete black-box algorithms are able to outperform problem specific ones.

3 Implementation validation

In order to compare results obtained on our platform using Evolving Objects (see [11]), we ran the benchmark on two CMA-ES with the same configuration as [7]: one has default parameters, one has a population size of $\mu = 50$ and $\lambda = 100$. In both cases, we stopped a run when $f_{best} \leq f^* + \frac{f^*}{1e15}$ (*ie.* the optimum is reached), when the best fitness stagnated for too long or when the allocated budget was consumed. This budget was set to $2e5$ function evaluations (all budgets in this paper are expressed in terms of function evaluations).

As can be seen in Table 1, the mean fitnesses we were able to obtain are comparable to the ones reported in [7]: sometimes better, sometimes worse, but never by far (except in high dimension where the results are degraded, probably due to different parameters). This allows us to validate our implementation, and serves as a baseline for the rest of our work.

In the original paper, the number of function evaluations was reported with the mean fitnesses. The given explanation is that the main focus of the exercise being the fitness - and not so much failures or successes - the required number of function evaluations to get a result is not that important: each algorithm should have the time - the budget - to reach the optimum or at least converge.

While this is perfectly valid, we don't feel comfortable to do so as it weakens the comparison between algorithms. Instead of reporting the mean number of function evaluations used, we will prefer the SP1 measure as defined in [1] :

D	k	$f*$	CMA-ES(50,100)	CMA-ES(50,100) SP1	CMA-ES	CMA-ES SP1
G	02	6.02546e11	6.025472e11 (2.8e-04)	8.3400e03 (6.4e02)	1.172558e12 (7.6e11)	4.8798e04 (3.3e04)
	03	2.94506e11	4.486461e11 (1.5e11)	4.2083e04 (2.6e04)	8.196432e11 (1.2e12)	∞
	04	1.04474e11	3.362127e11 (1.4e11)	3.9970e05 (1.8e05)	7.629370e11 (4.7e11)	∞
	05	5.97615e10	2.049802e11 (1.4e11)	6.8410e05 (2.0e05)	7.488858e11 (1.2e12)	∞
	06	3.59085e10	1.585765e11 (1.5e11)	∞	8.818792e11 (6.3e11)	∞
	07	2.19832e10	1.051648e11 (1.1e11)	∞	6.463187e11 (7.5e11)	∞
	08	1.33854e10	1.068587e11 (9.3e10)	∞	7.005948e11 (7.1e11)	∞
	09	7.80442e09	2.780667e11 (3.1e11)	∞	1.003192e12 (9.7e11)	∞
	10	6.44647e09	5.869352e11 (5.2e11)	∞	7.677317e11 (6.5e11)	∞
I	02	1.52348e2	1.523480e02 (6.4e-14)	1.8344e04 (5.0e02)	1.523542e02 (3.0e-03)	∞
	03	7.88514e01	7.885144e01 (2.5e-14)	7.2048e04 (2.2e03)	1.279512e02 (1.2e02)	∞
	04	5.72285e01	5.836730e01 (3.9e00)	∞	9.728522e01 (3.5e01)	∞
	05	4.64462e01	4.766177e01 (1.7e00)	∞	1.330878e02 (1.3e02)	∞
	06	3.90400e01	4.149195e01 (2.9e00)	∞	1.292478e02 (1.3e02)	∞
	07	3.42982e01	4.037920e01 (3.5e00)	∞	7.892632e01 (4.5e01)	∞
	08	2.99889e01	3.739813e01 (4.2e00)	∞	7.750688e01 (5.4e01)	∞
	09	2.77861e01	3.831817e01 (5.3e00)	∞	8.018775e01 (7.6e01)	∞
	10	2.58341e01	5.653196e01 (6.9e01)	∞	9.553900e01 (1.0e02)	∞
R	02	8.93378e04	8.933783e04 (5.0e-12)	6.8260e03 (1.1e03)	8.933783e04 (3.1e-11)	3.5903e04 (5.0e03)
	03	5.10635e04	5.110393e04 (4.6e01)	2.0453e04 (5.3e03)	5.473043e04 (9.8e03)	∞
	04	1.28811e04	1.288105e04 (0.0e00)	∞	2.046652e04 (1.5e04)	∞
	05	1.01267e04	1.032449e04 (5.0e02)	∞	3.209521e04 (1.4e04)	∞
	06	8.57541e03	8.919118e03 (5.1e02)	2.5490e05 (1.7e04)	2.605724e04 (1.3e04)	∞
	07	7.12620e03	7.634386e03 (4.4e02)	7.7641e05 (4.9e04)	2.309534e04 (6.1e03)	∞
	08	6.14964e03	6.635902e03 (3.9e02)	∞	2.061007e04 (5.2e03)	∞
	09	5.18165e03	7.464273e03 (3.6e03)	∞	1.906988e04 (5.3e03)	∞
	10	4.44628e03	1.095691e04 (5.0e03)	∞	1.696298e04 (5.6e03)	∞

Table 1: Average fitness results and SP1 measure (mean and standard deviation) for CMA-ES and CMA-ES(50,100). An SP1 measure of ∞ means that the optimum could not be reached for any of the 50 runs. Results are give for the German Town (G), Iris (I) and Ruspini (R) datasets for all values of k.

$SP1 = \frac{\mathbb{E}(T_s)}{p_s}$, where $\mathbb{E}(T_s)$ is the expected number of function evaluations used in a successful run and p_s is the probability to get a success for a given run.

This measure has some disadvantages (*eg.* when the success probability is 0), but it allows a more accurate comparison between algorithms, in particular when using restarts. In such a way, two possible strategies (aiming for a 100% success rate no matter the cost or allowing restarts if the solution is not quickly found) are both possible and their performances can be compared without bias one way or another.

4 Differential Evolution

While the first work on this clustering benchmark obviously did not try to compare each and every possible optimisation algorithm, we felt that given the specifities of the problem, Differential Evolution (DE) [19] could perform quite well. This feeling is substantiated by [4] in which DE is said to perform very well on a lot of testbeds.

Built around crossovers, the DE algorithm replaces part of a given individual with two or more others. Many different variants of DE exist, each one defining

the crossovers rule. The one we chose was "DE/curr-to-best/1". For a given generation, we then have:

DE/curr-to-best/1: $U(0,1)$ is a random uniformly distributed number between 0 and 1, CR is the crossover rate parameter, f_1 and f_2 are two real numbers, $Best$ is the best individual in the generation, and f is the evaluation function, n is the dimension of a point in the given dataset.

for each individual I **do**
  $Y \leftarrow I$
  Randomly choose A and B, two individuals distinct from I and $Best$
  Randomly select an index $R \in \{1, \dots, n\}$
  for all $i \in \{1, \dots, n\}$ **do**
    if $i = R$ **or** $U(0,1) < CR$ **then**
      $Y(i) \leftarrow I(i) + f_1(A(i) - B(i)) + f_2(Best(i) - I(i))$
    end if
  end for
  if $f(Y) < f(I)$ **then**
    Replace I by Y
  end if
end for

The only difference from "DE/best/1" is thus the update formula, which is $Y(i) \leftarrow Best(i) + f_1(A(i) - B(i))$.

In the spirit of [7], we didn't try to tune the algorithm's parameters. Instead, in the absence of a standard recommendation, we set $CR = 0.5$, $f_1 = f_2 = 0.8$ for a population size of 30. The initialisation points were randomly drawn with a normal distribution of mean the average of the range of the variables, with a standard deviation of a third of that average. We used here the same stopping criteria as with CMA-ES in our previous experiment.

5 Progressive Differential Evolution

In some of our first trials, when studying the reasons for failures to reach the optimum, we reached the conclusion that in a third of the failed runs, this failure was due to falling in a local optimum. As can be seen on Figure 1 with a $3e4$ budget, in most cases the failures to reach the optimum are simply due to a lack of budget: the clusters found are not exactly at the optimum but centered around them. In fact, by increasing the budget, we saw that indeed, those points went to the optimum.

In the second case however, we can see that the points found are symmetrically opposed to the optimum solution, one cluster at the top, two at the bottom. This configuration on the Ruspini problem with $k = 3$ gives a fitness of ≈ 51155, which is only slightly worse than the optimum of ≈ 51063. As such, there is only a very small probability that any mutation would get to the real optimum close enough to improve the solution.

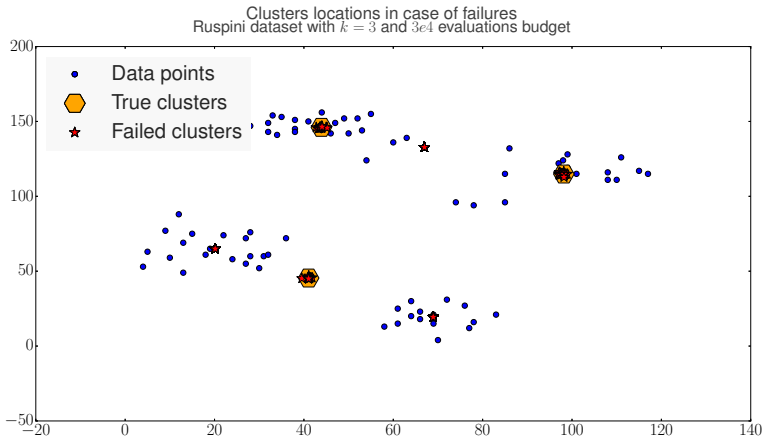


Fig. 1: Clusters position on failure cases, Ruspini dataset with $k = 3$

In order to avoid this, we introduced "Progressive Widening", known as the Sieves Method [16] in statistics. The basic idea is to start optimising a small number of clusters, and to increase that number at some point in the process:

PDE: k_{max} is the desired number of clusters, N is the dimensionality of each point, R determines the number of generations to do with k clusters

Initialise population
$k \leftarrow 1$
while not stop **do**
  for $i = 0$ to R **do**
    Run one generation of DE on the $k \cdot N$ first parameters
  end for
  $k \leftarrow min(k + 1, k_{max})$
end while

Here, we chose to use $R = 100$, which means that every one hundred generation, we increase the number of parameters to optimise until we reach $n \cdot k_{max}$.

Of course, the fact that we optimise k clusters doesn't mean that the others "disappear": they are still taken into account in the evaluation, but don't move from their initial position, which is the center of the search space. This means that even when training k clusters, there is always one more that can be selected as the nearest from a given point. While we could have completely removed them from the evaluation, we felt that this would have reduced the black-box context of the problem.

In fact, one could argue that we are only able to use Progressive Widening by weakening the black-box setting of the problem. Indeed, since we know the

dimension of the problem, we know that to add a cluster we have to add N parameters. We don't think this is an issue however, this knowledge being as much part of the specification of the problem as the definition of the search space.

6 Results

6.1 DE vs CMA-ES

D	k	$f*$	DE	DE SP1	PDE	PDE SP1
G	02	6.02546e11	6.025472e11 (5.0e-04)	5.6160e03 (6.6e02)	6.025472e11 (5.0e-04)	9.1242e03 (2.9e02)
	03	2.94506e11	3.006674e11 (4.3e10)	9.0893e03 (1.8e03)	2.945066e11 (0.0e00)	1.3496e04 (2.6e02)
	04	1.04474e11	1.500823e11 (8.1e10)	2.3898e04 (1.6e04)	1.044747e11 (0.0e00)	1.9849e04 (4.5e02)
	05	5.97615e10	7.423346e10 (3.6e10)	4.5462e04 (4.3e04)	6.065579e10 (6.3e09)	2.7124e04 (1.6e03)
	06	3.59085e10	4.776401e10 (3.8e10)	4.8107e04 (2.3e04)	3.611288e10 (1.4e09)	3.3399e04 (1.5e03)
	07	2.19832e10	3.176165e10 (1.6e10)	2.0357e05 (1.1e05)	2.423709e10 (5.1e09)	8.8896e04 (3.5e04)
	08	1.33854e10	2.182272e10 (8.3e09)	∞	1.639762e10 (4.1e09)	∞
	09	7.80442e09	1.562879e10 (6.3e09)	∞	1.127751e10 (2.9e09)	∞
	10	6.44647e09	1.281459e10 (6.2e09)	∞	8.793075e09 (4.3e09)	2.2032e06 (3.4e05)
I	02	1.52348e02	1.523480e02 (0.0e00)	8.9892e03 (2.7e03)	1.523480e02 (0.0e00)	1.3222e04 (3.5e02)
	03	7.88514e01	8.032188e01 (1.0e01)	2.0023e04 (1.3e04)	7.885212e01 (1.5e-03)	2.1965e04 (7.9e02)
	04	5.72285e01	5.867260e01 (5.2e00)	5.0221e04 (2.7e04)	5.722847e01 (4.8e-14)	2.6411e04 (5.6e03)
	05	4.64462e01	4.978281e01 (4.3e00)	1.3932e05 (8.1e04)	4.847058e01 (1.8e00)	1.7655e05 (5.8e04)
	06	3.90400e01	4.210588e01 (3.4e00)	1.9240e05 (7.7e04)	3.961203e01 (1.5e00)	1.2713e05 (4.7e04)
	07	3.42982e01	3.735682e01 (3.4e00)	1.3620e06 (3.7e05)	3.506627e01 (1.6e00)	4.1865e06 (0.0e00)
	08	2.99889e01	3.288639e01 (3.3e00)	1.0018e06 (1.2e05)	3.084912e01 (1.4e00)	9.5156e05 (1.3e05)
	09	2.77861e01	2.928749e01 (2.2e00)	1.6612e06 (1.1e05)	2.855921e01 (1.2e00)	1.3116e06 (2.8e05)
	10	2.58341e01	2.795759e01 (2.7e00)	3.5490e06 (0.0e00)	2.695760e01 (9.1e-01)	7.8765e06 (0.0e00)
R	02	8.93378e04	8.933783e04 (0.0e00)	5.9892e03 (2.3e03)	8.933783e04 (0.0e00)	1.0646e04 (4.0e02)
	03	5.10635e04	5.109841e04 (4.5e01)	2.0758e04 (9.1e03)	5.106348e04 (4.1e-11)	1.1740e04 (4.0e02)
	04	1.28811e04	1.288105e04 (0.0e00)	∞	1.288105e04 (0.0e00)	1.1175e06 (0.0e00)
	05	1.01267e04	1.015393e04 (1.9e02)	∞	1.013935e04 (1.1e01)	∞
	06	8.57541e03	8.664380e03 (2.5e02)	1.0505e05 (8.3e04)	8.660781e03 (1.1e02)	9.7329e04 (2.2e04)
	07	7.12620e03	7.179452e03 (1.4e02)	1.0829e05 (7.4e04)	7.193774e03 (1.0e02)	1.6640e05 (1.0e04)
	08	6.14964e03	6.246995e03 (3.6e02)	1.7645e05 (1.2e05)	6.168576e03 (3.4e01)	7.7184e04 (2.0e04)
	09	5.18165e03	5.441820e03 (4.2e02)	2.8236e05 (1.3e05)	5.314655e03 (1.9e02)	1.3664e05 (5.8e04)
	10	4.44628e03	4.694111e03 (4.3e02)	2.6013e05 (9.9e04)	4.622832e03 (8.4e01)	8.9633e05 (4.6e04)

Table 2: Average fitness results and SP1 measure (mean and standard deviation) for DE and PDE. An SP1 measure of ∞ means that the optimum could not be reach for any of the 50 runs. Results are give for the German Town (G), Iris (I) and Ruspini (R) datasets for all values of k.

The results we obtained with DE shown in Table 2 and Figure 3 were very good, often better - sometimes by far - than CMA-ES(50,100). The first striking result is that DE more consistently reaches the optimum solution: in only five cases (three on the German Town dataset, two on the Ruspini dataset) DE was not able to reach the optimum at least once in the 50 runs reported here.

As such, it comes as no surprise that the average fitness obtained by DE after 50 runs was improved in almost all cases (except on the Iris dataset when

$k \leq 6$ and on the Ruspini dataset with $k = 3$). While this improvement is not necessarily ground breaking on the Ruspini dataset for example, it is much more important on the German Town problem (see Figure 3a).

6.2 DE vs PDE

The effects of the Progressive Widening on DE were twofold: first, it globally improved the average fitness across the board: in all but one trial (Ruspini with $k = 7$), the mean fitness and associated standard deviation were better with Progressive Widening than without. Once more, this is most notable on the German Town problem. Furthermore, in only one case now (Iris dataset with $k = 10$) is CMA-ES the best: on all other cases, PDE gets better results.

The second effect (shown in Table 3) was the one we expected: the success rate improved, we find the optimum more often. Most notably, with $k = 3$ on the Ruspini dataset, we went up from a 62% success rate to a full 100%: we no longer fall in the local optimum reported in Figure 1, which was our goal when adding Progressive Widening to DE.

In five cases though the rates went down but only in two cases was this decrease important: from 58% to 20% on the Ruspini dataset with $k = 7$ (which is also the only case where the mean fitness obtained by DE is better than PDE) and from 32% to 6% still on the Ruspini dataset but with $k = 10$. Interestingly here, while the success rate decreased by almost 30%, the mean fitness obtained by PDE is still better than the one from DE.

In fact thanks to this, we can see that while the Progressive Widening works very well in most instances in order to avoid a local minimum, in some rare cases it is exactly the opposite, as we can see on Ruspini with $k = 10$. While the solution found is often very good - there is not a huge difference between DE and PDE mean fitness there - by plotting the proposed solution we see that when PDE fails to reach the optimum and stagnates, it is because it fell in a local minimum.

6.3 The cost of PDE

Given the fact that the budget and stopping criteria are the same for DE and PDE, the SP1 measures reported in Table 2 mostly reflect the differences in success rate we saw previously. In the few cases were both algorithm have (almost) the same success rate, we can see that the SP1 measure is higher (or worse) for PDE than for DE : the introduction of Progressive Widening is not without cost.

This is even more clearly illustrated in Figure 2, where some statistics on the fitnesses of 50 runs of DE and PDE are plotted. On the first few evaluations, PDE performs two orders of magnitude worse than DE, still one order of magnitude worse after $5e3$ evaluations, and it is not until at least $1.5e4$ evaluations that PDE performs at least as well as DE. While this is to be expected since until then not all clusters are optimised, it is still something to take into account.

(a) German Town dataset

k	CMA(50,100)	DE	PDE
02	100	100	100
03	48	98	100
04	10	76	100
05	18	74	98
06	0	74	88
07	0	38	74
08	0	0	0
09	0	0	0
10	0	0	8

(b) Iris dataset

k	CMA(50,100)	DE	PDE
02	100	100	100
03	100	86	84
04	0	56	100
05	0	28	28
06	0	32	50
07	0	4	2
08	0	6	8
09	0	4	8
10	0	2	2

(c) Ruspini dataset

k	CMA(50,100)	DE	PDE
02	100	100	100
03	56	62	100
04	0	0	2
05	0	0	0
06	24	46	36
07	16	58	20
08	0	42	64
09	0	32	52
10	0	32	6

Table 3: Success rate for CMA(50,100), DE and PDE

7 Conclusion

DE performs very well on clustering problems, even when compared to clustering algorithms or CMA-ES, the current state of the art on this benchmark. This, by itself, is a very impressive result.

Our proposed variant of DE, PDE, gets even better results in most cases illustrating the good impact the concept of Progressive Widening can have on a black box algorithm.

In addition, we propose a baseline for the SP1 measure that will allow more robust comparisons of algorithms on this benchmark in the future.

8 Further work

While still following the spirit of the original paper by not tuning the algorithms parameters, there are still many possibilities to try and improve the results. Some ways to do so include other mutations rules for DE (DE/rand/1, DE/best/1, *etc.*), using Adaptive Differential Evolution, or other variants.

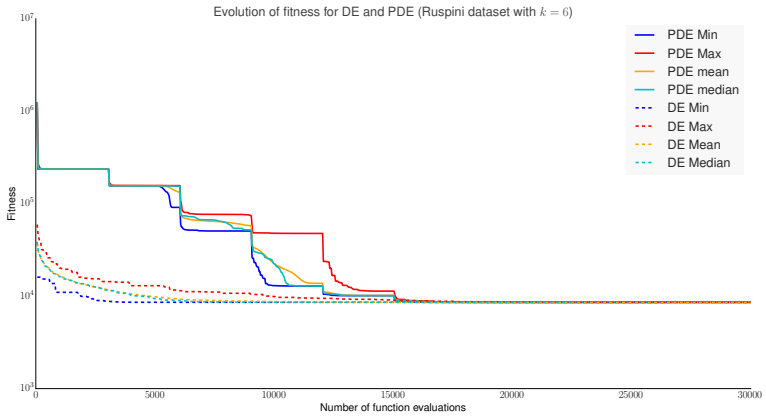
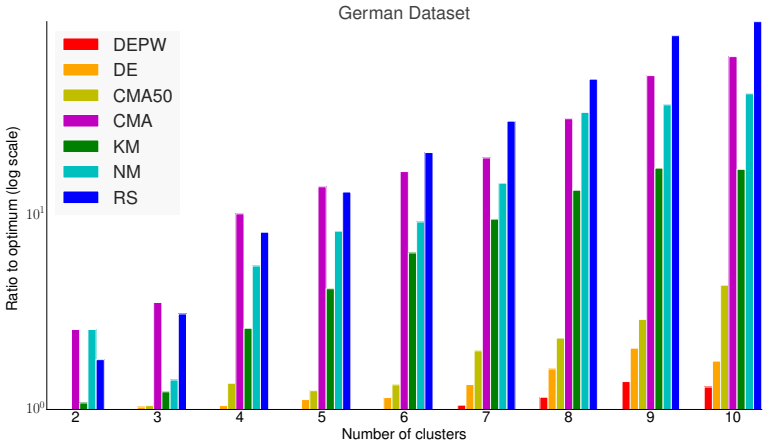


Fig. 2: Fitness statistics evolution on the Ruspini dataset with $k = 6$ with DE and PDE. The Progressive Widening has a clear cost at the beginning of the optimisation process.

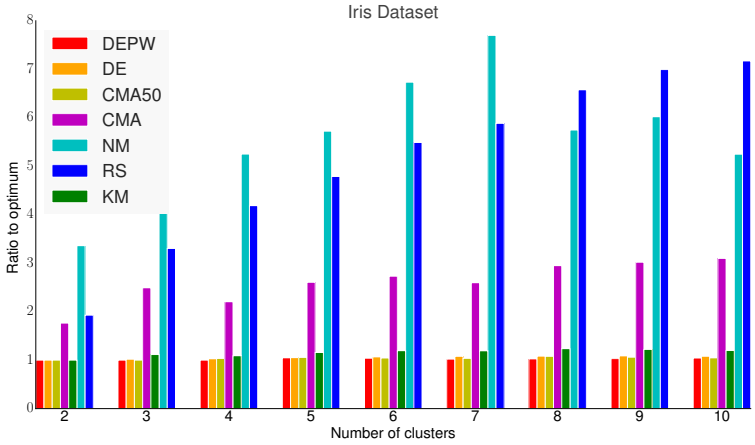
Of course, another way could be to use the progressive strategy on other algorithms when possible: for algorithms with covariance matrices such as CMA-ES, CMSA [3] or even the self-adaptive with covariance algorithm [14] such a change is not trivial. But for others like Particle Swarm Optimisation [5, 17] or the other members of the Self-Adaptive family [2] (isotropic or anisotropic, 1+1, *etc.*) this is quite straightforward.

The most interesting improvements could be done on the Progressive Widening concept. For example, knowing why in some instances it is more prone to fall in a local minimum would be interesting.

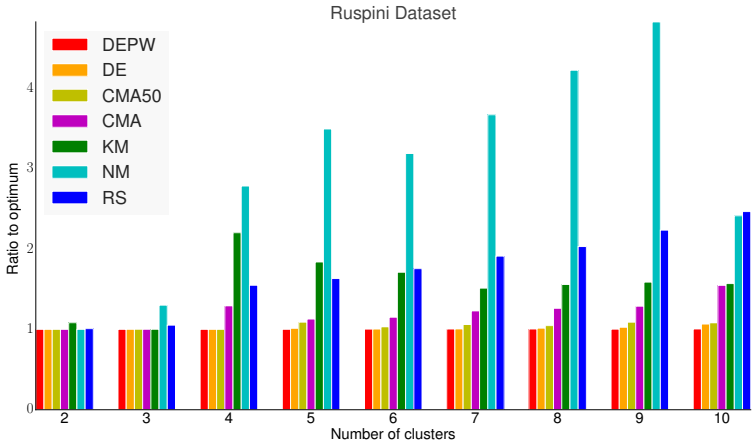
Furthermore, we have seen that the Progressive Widening is not without cost. To lessen that cost, instead of adding clusters (or parameters in the general case) at fixed timesteps we could design a rule that dynamically adds them when the fitness is reasonably stable. An intermediate step might be to add those parameters after an increasing number of timesteps (evaluations or generations) with a logarithmic rule for example, such that the more parameters are currently optimised, the more time is spent on them before adding more.



(a) German Towns dataset



(b) Iris dataset



(c) Ruspini dataset

Fig. 3: Performance as a ratio to the optimum ($\frac{\hat{f}}{f^*}$) of results reported in the original paper compared to DE and PDE with a $2e5$ budget. From left to right are PDE, DE, CMA-ES(50,100), CMA-ES, KM, NM and RS.

References

1. Auger, A., Hansen, N.: Performance evaluation of an advanced local search evolutionary algorithm. In: Evolutionary Computation, 2005. The 2005 IEEE Congress on. vol. 2, pp. 1777–1784. IEEE (2005), http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1554903
2. Beyer, H.G.: The Theory of Evolution Strategies. Natural Computing Series, Springer, Heideberg (2001)
3. Beyer, H.G., Sendhoff, B.: Covariance matrix adaptation revisited - the CMSA evolution strategy. In: Rudolph, G., Jansen, T., Lucas, S.M., Poloni, C., Beume, N. (eds.) Proceedings of PPSN. pp. 123–132 (2008)
4. Das, S., Suganthan, P.N.: Differential evolution: A survey of the state-of-the-art. IEEE Trans. on Evolutionary Computation 15(1), 4–31 (2011)
5. Eberhart, R., Kennedy, J.: A new optimizer using particle swarm theory. In: , Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 1995. MHS '95. pp. 39–43 (Oct 1995)
6. Fisher, R.A.: The Use of Multiple Measurements in Taxonomic Problems. Annals of Eugenics 7(2), 179–188 (1936), http://onlinelibrary.wiley.com/doi/10.1111/j.1469-1809.1936.tb02137.x/abstract
7. Gallagher, M.: Clustering Problems for More Useful Benchmarking of Optimization Algorithms. In: Dick, G., Browne, W.N., Whigham, P., Zhang, M., Bui, L.T., Ishibuchi, H., Jin, Y., Li, X., Shi, Y., Singh, P., Tan, K.C., Tang, K. (eds.) Simulated Evolution and Learning, pp. 131–142. No. 8886 in Lecture Notes in Computer Science, Springer International Publishing (Jan 2014), http://link.springer.com/chapter/10.1007/978-3-319-13563-2_12
8. Gould, N.I.M., Orban, D., Toint, P.L.: Cuter and sifdec: A constrained and unconstrained testing environment, revisited. ACM Trans. Math. Softw. 29(4), 373–394 (2003)
9. Hansen, N., Ostermeier, A.: Completely Derandomized Self-Adaptation in Evolution Strategies. Evolutionary Computation 9(2), 159–195 (Jun 2001)
10. Hansen, N., Auger, A., Ros, R., Finck, S., Posik, P.: Comparing Results of 31 Algorithms from the Black-Box Optimization Benchmarking BBOB-2009. In: ACM-GECCO Genetic and Evolutionary Computation Conference. Portland, United States (Jul 2010), https://hal.archives-ouvertes.fr/hal-00545727, pp. 1689-1696
11. Keijzer, M., Merelo, J.J., Romero, G., Schoenauer, M.: Evolving Objects: A General Purpose Evolutionary Computation Library. In: Collet, P., Fonlupt, C., Hao, J.K., Lutton, E., Schoenauer, M. (eds.) Artificial Evolution, pp. 231–242. No. 2310 in Lecture Notes in Computer Science, Springer Berlin Heidelberg (2002), http://link.springer.com/chapter/10.1007/3-540-46033-0_19
12. du Merle, O., Hansen, P., Jaumard, B., Mladenovic, N.: An Interior Point Algorithm for Minimum Sum-of-Squares Clustering. SIAM Journal on Scientific Computing 21(4), 1485–1505 (Jan 1999), http://epubs.siam.org/doi/abs/10.1137/S1064827597328327
13. Nelder, J.A., Mead, R.: A Simplex Method for Function Minimization. The Computer Journal 7(4), 308–313 (Jan 1965), http://comjnl.oxfordjournals.org/content/7/4/308
14. Rechenberg, I.: Evolutionstrategie: Optimierung Technischer Systeme nach Prinzipien des Biologischen Evolution. Fromman-Holzboog Verlag, Stuttgart (1973)

15. Ruspini, E.H.: Numerical methods for fuzzy clustering. Information Sciences 2(3), 319–350 (Jul 1970), http://www.sciencedirect.com/science/article/pii/S0020025570800561
16. Shen, X., Wong, W.H.: Convergence Rate of Sieve Estimates. The Annals of Statistics 22(2), 580–615 (Jun 1994), http://projecteuclid.org/euclid.aos/1176325486
17. Shi, Y., Eberhart, R.: A modified particle swarm optimizer. In: , The 1998 IEEE International Conference on Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence. pp. 69–73 (May 1998)
18. Spaeth, H.: Cluster analysis algorithms for data reduction and classification of objects (1980), http://cds.cern.ch/record/102044
19. Storn, R., Price, K.: Differential Evolution A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. Journal of Global Optimization 11(4), 341–359 (Dec 1997), http://link.springer.com/article/10.1023/A%3A1008202821328
20. Suganthan, P.N., Hansen, N., Liang, J.J., Deb, K., Chen, Y.P., Auger, A., Tiwari, S.: Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. Tech. Rep. AND KanGAL Report #2005005, IIT Kanpur, India (2005), http://public.cranfield.ac.uk/sims_staff/wcat/cec2005/sessions/