Comparing optimizers on a unit commitment problem

Vincent Berthier

Comparing optimizers on a unit commitment problem

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. This paper compares several black-box optimization algorithms on a unit commitment problem. Compared to existing testbeds, this one provides several scales, is real-world, and none of the compared algorithms were created by the author of the testbed. Differential Evolution basically performs best overall, though not for all test cases.

1 Introduction

Several testbeds were provided by [24, 1, 10, 8] for non-linear optimization. In the evolutionary computation community the most widely used might be [24]. We here propose an alternative set of experiments, on which we compare a set of optimisers.

2 Algorithms

Over the years, a number of evolutionary algorithms were developed, each intended to address a specific issue, none being able to give a good solution to every single problem.

Among this set of algorithm, there are some "stars", widely used in the continuous optimisation community. Some of those algorithms are:

- Self-Adaptive Evolution Strategies (SA-ES[2]), which come in three main flavours: isotropic, where there is only one mutation parameter; anisotropic with one mutation factor for each parameter; anisotropic with covariance matrix[21].
- Covariance Matrix Adaptation Evolution Strategy (CMA-ES[11]) where the mutation step sizes are guided by cumulative step-size adaptation and also features full covariance matrix adaptation.
- The "simple" $(1 + 1)$-ES, where the step size is updated according to the success or failure to improve.
- Nelder-Mead[17], a simplex method where one point is moved at each generation.
- Particle Swarm Optimisation[13, 22], where the position of each point - or particle - is influenced by those of its neighbours.

– Differential Evolution[23], an algorithm where mutations are done by crossovers with one or more individuals.

Thanks to their notoriety, many different implementations of those algorithm exist. In this paper, we will use those offered by the Evolving Object[12] library when possible, and extend it for those algorithms not yet there.

In many comparisons, an extensive work is done to tune each individual meta-parameters of those optimisers. Here, we elected to compare them by only using their defaults or recommended parameters: their ease of use "out of the box" is after all an important criteria in many real world situations. With N being the dimension of the problem, those parameters are:

– SA-ES: population size $\lambda = 12$ and parent population size $\mu = 3$. For anisotropic variant we used $\tau = 1\sqrt{2N}$, with $\beta = 0.0873$ for the covariance version.
– CMA-ES: population size $\lambda = 4 + \lfloor 3\log(N) \rfloor$, parent population size $\mu = \lambda/2$
– (1+1)-ES step size was multiplied by $r = 1.5$ in case of a successful mutation, and divided by $r = 1.5^{1/4}$ on a failure.
– Nelder-Mead: Population is of course $\mu = N + 1$, and mutation parameters are $\alpha = 1$, $\gamma = 2$, $\rho = -0.5$ and $\sigma = 0.5$.
– For PSO, the standard parameters are widely discussed ([18, 25, 5, 26, 6, 3]): We used a population $\mu = 30$, with 10 neighbours and update parameters $\omega = \frac{1}{2 \times log(2)}$, $\Phi_p = 0.5 + log(2)$, $\Phi_g = 0.5 + log(2)$, $velocity_{init} = 1.0$ and $velocity_{max} = 1.5$
– DE: The DE/curr-to-best/1 variant was used (mutate the selected individual with the best one of the generation) with mutation parameters $f_1 = 0.8$, $f_2 = 0.8$ and $cr = 0.5$.

3 Testbed

Our testbed has the following characteristics:

– It is a real problem, originally not designed for academic purpose. As a consequence, it has the same degree of partial separability as (at least some) real world problems.
– It includes several cases, with dimension ranging from 3 to several thousands. We should indeed include, later, bigger testcases.
– It is restricted to direct policy search for power systems. This category of problems is definitely an important one; we do not claim that our results have some validity beyond this scope.
– We compared algorithms and implementations in a neutral manner. We have no special interest for one algorithm or another, we just try to find which algorithm we should recommend as default in our optimization platform.
– This family of problems has a huge economical (billions of dollars per year) and environmental impact.

Our testbed has the following parameters:

- Number of time steps.
- Number of stocks (number of state variables).
- Parameters for the inflows and demand and their variabilities, which are held constant over our experiments.

It is available at `https://www.lri.fr/~teytaud/uctest/uctest.html`.

The number of decisions per time step is equal to the number of stocks (we decide how much water we use for each stock). The number of inputs for making each decision is one observation per stock (the level), plus the 4 calendar factors. Hence, the number of action variables is $nbActions = nbStocks$ and the number of input variables is $nbInputs = nbStocks + 4$. The number of parameters for a given problem can be computed as follows (and the detailed policies can be seen at `https://www.lri.fr/~teytaud/uctest/uctest.html`):

- Handcrafted policy: the number of parameters is always $N = 3$.
- Conformant planning (a sequence of decisions, applied independently of observations): $N = T \times NbActions = T \times NbStocks$ (one parameter per time step and per stock).
- Neural network (feedforward, one hidden layer): the number of parameters is $N = NbNeurons \times (NbInputs + NbActions + 1) + NbActions$:
  - $NbActions$ parameters for the biases for the output,
  - $NbInputs + 1$ for the input weights of each neuron,
  - and $NbActions$ connections between each hidden neuron and the output neurons).
- Fuzzy control: the number of parameters is $N = NbRules \times (2nbInputs + nbActions + 1)$ because each rule has $2NbInputs + 1$ parameters for the antecedent (one scale and one average value for each coordinate, plus one default rule weight) and $NbActions$ parameters for the succedent.

For example, with 25 stocks and 100 timesteps, the number of parameters are 3 for the handcrafted policy; 2500 for conformant planning; $2681 = 32(58 + 25) + 25$ for fuzzy systems with 2^5 rules; $1785 = 32(29 + 25 + 1) + 25$ for neural networks with 2^5 neurons.

The fuzzy rule used in the experiments uses a membership function product of coordinate-wise inverse distances. This was selected among various membership functions after preliminary experiments.

4 Summary of results

We summarize our results in tables below for 512 seconds of budget. Overall, DE performed best.

4.1 Overview and results per problem size

In Table 1 we give results averaged over all problems, and then for different sizes (< 10 parameters, 10 to 99, 100 to 999, and 1000+. Due to size constraints only

the average performances of the algorithms are shown, but the underlying study was performed based on average, worst, quartile and decile performances. We can see that in high dimension, CMA suffers due to its internal cost, as shown by the small number (relative to the other algorithms).

All problems average perf	
DE	0.79 +- 0.023
CMA	0.75 +- 0.035
SAiso	0.73 +- 0.023
PSO	0.66 +- 0.033
SA	0.66 +- 0.026
SAcov	0.66 +- 0.025
1+1	0.6 +- 0.027
NM	0.54 +- 0.036

Nb params $\leq$ 10 average perf	
PSO (1560k)	0.83 +- 0.092
DE (1545k)	0.82 +- 0.06
SAcov (1668k)	0.78 +- 0.064
CMA (1580k)	0.78 +- 0.054
SAiso (1580k)	0.78 +- 0.059
SA (1649k)	0.76 +- 0.077
NM (1548k)	0.67 +- 0.073
1+1 (1772k)	0.62 +- 0.071

10 $\leq$nb params $<$ 100 average perf	
DE (1622k)	0.87 +- 0.028
CMA (1660k)	0.82 +- 0.047
SAiso (1642k)	0.69 +- 0.04
PSO (1670k)	0.61 +- 0.04
1+1 (1629k)	0.59 +- 0.05
SAcov (1619k)	0.56 +- 0.056
SA (1618k)	0.55 +- 0.058
NM (1666k)	0.51 +- 0.046

100 $\leq$nbParams$<$ 1000 average perf	
DE (936k)	0.76 +- 0.035
CMA (641k)	0.75 +- 0.051
SAiso (932k)	0.74 +- 0.034
SA (926k)	0.68 +- 0.031
SAcov (875k)	0.68 +- 0.029
PSO (948k)	0.64 +- 0.053
1+1 (930k)	0.61 +- 0.041
NM (942k)	0.54 +- 0.056

1000 $\leq$nbParams$<$ 10000 average perf	
PSO (167k)	0.8 +- 0.088
SAiso (159k)	0.78 +- 0.064
SA (156k)	0.73 +- 0.056
DE (162k)	0.73 +- 0.075
SAcov (104k)	0.7 +- 0.054
CMA (5k)	0.59 +- 0.14
1+1 (159k)	0.58 +- 0.083
NM (152k)	0.54 +- 0.14

Table 1: Each result is linearly normalized so that 1 is the maximum (best) result, and 0 is the minimum (worst) result over all runs for this controller and this unit commitment problem (so higher is better). The numbers between parenthesis are the number of fitness evaluations performed in the given budget of 512s.

4.2 Per family of controllers and per problem size

For each testbed, we specify with which frequency an algorithm (in row) outperforms another one (in column). These results in Tables 2 and 3 are the same as the results above, but broken down on the different test cases. Due to size constraints results on the large testbed are not shown, but they are essentialy the same as the medium case, the only major difference is that PSO becomes the best algorithm on the Conformant Planning function.

5 Conclusions and further work

A short conclusion is that DE performs best overall, with also an excellent stability. This is consistent with the success of DE on several competitions - variants or combinations of DE have won the CEC 2006, CEC 2010 and CEC 2013 competitions[7, 14].

Still, there is no clear-cut conclusion; DE is a bit weaker with neural network controllers, and even algorithms which are usually not that stable (*eg.* Nelder-

Mead or $(1+1)$-ES) sometimes perform very well. In particular, the important special case of conformant planning is very well tackled by the simple $(1+1)$-ES.

PSO performed well in high dimensional problems. Nelder-Mead was surprisingly good in spite of long initialization (with a population linear in the dimension).

CMA performed very well in some cases, but was in general clearly outperformed by DE.Variants of CMA with limited covariance (*eg.* diagonal) might be considered to alleviate the dimensional problem.

For sure, this work is not intended to be some kind of "final" comparison. This is one test case, with the advantage that it is a real world and (ecologically and economically) important test case. Besides the fact that our test cases can lead to different conclusions, we do not take into account the limit in terms of parallelization, whereas parallelization is one of the main body of work around PSO [16, 15, 20, 4, 9].

The main further works are (i) including more algorithms (*eg.* Newuoa[19], variants of DE and memetic algorithms) (ii) including noisy optimization (iii) parallel setting, *eg.* constraining the population size to 1000 (iv) bigger test cases (we can without effort extend the test case to 100 stocks and 2000 time steps, which is consistent with some real world cases - unit commitment problems exist at various scales).

References

1. Auger, A., Finck, S., Hansen, N., Ros, R.: BBOB 2009: Comparison Tables of All Algorithms on All Noiseless Functions. Technical Report RT-0383 (Apr 2010), https://hal.inria.fr/inria-00471251
2. Beyer, H.G.: The Theory of Evolution Strategies. Natural Computing Series, Springer, Heideberg (2001)
3. Bratton, D., Kennedy, J.: Defining a standard for particle swarm optimization. In: IEEE Swarm Intelligence Symposium. pp. 120–127 (2007), http://dx.doi.org/10.1109/SIS.2007.368035
4. Chang, J.F., Chu, S.C., Roddick, J.F., Pan, J.S.: A parallel particle swarm optimization algorithm with communication strategies. J. Inf. Sci. Eng. 21(4), 809–818 (2005)
5. Clerc, M., Kennedy, J.: The particle swarm - explosion, stability, and convergence in a multidimensional complex space. Evolutionary Computation, IEEE Transactions on 6(1), 58–73 (2002)
6. Clerc, M.: Beyond standard particle swarm optimisation. IJSIR 1(4), 46–61 (2010), http://dblp.uni-trier.de/db/journals/ijsir/ijsir1.html#Clerc10
7. 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)
8. Gallagher, M.: Clustering problems for more useful benchmarking of optimization algorithms. In: Simulated Evolution and Learning - 10th International Conference, SEAL 2014, Dunedin, New Zealand, December 15-18, 2014. Proceedings. pp. 131–142 (2014), http://dx.doi.org/10.1007/978-3-319-13563-2_12
9. Gardner, M., McNabb, A.W., Seppi, K.D.: A speculative approach to parallelization in particle swarm optimization. Swarm Intelligence 6(2), 77–116 (2012)

10. 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)
11. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation 11(1) (2003)
12. Keijzer, M., Merelo, J.J., Romero, G., Schoenauer, M.: Evolving objects: A general purpose evolutionary computation library. In: Artificial Evolution. pp. 231–244 (2001), citeseer.ist.psu.edu/keijzer01evolving.html
13. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks. pp. 1942–1948 (1995)
14. LaTorre, A., Muelas, S., Pena, J.M.: Large scale global optimization: Experimental results with mos-based hybrid algorithms. In: Evolutionary Computation (CEC), 2013 IEEE Congress on. pp. 2742–2749 (June 2013)
15. Mahdad, B., Srairi, K., Bouktir, T., Benbouzid, M.: Fuzzy Controlled Parallel PSO to Solving Large Practical Economic Dispatch. In: IEEE (ed.) Proceedings of the 2010 IEEE International Conference of the IEEE Industrial Electronics Society. pp. 2695–2701. IEEE, Phoenix, United States (Nov 2010), http://hal.archives-ouvertes.fr/hal-00564733
16. McNabb, A., Monson, C., Seppi, K.: Parallel pso using mapreduce. In: Evolutionary Computation, 2007. CEC 2007. IEEE Congress on. pp. 7–14 (2007)
17. Nelder, J., Mead, R.: A simplex method for function minimization. Computer Journal 7 pp. 308–311 (1965)
18. Parsopoulos, K.E., Vrahatis, M.N.: Parameter selection and adaptation in unified particle swarm optimization. Mathematical and Computer Modelling 46(1-2), 198–213 (2007)
19. Powell, M.J.D.: Developments of newuoa for minimization without derivatives. IMA J Numer Anal pp. drm047+ (February 2008), http://dx.doi.org/10.1093/imanum/drm047
20. Schutte, J.F., Reinbolt, J.A., Fregly, B.J., Haftka, R.T., George, A.D.: Parallel global optimization with the particle swarm algorithm. JOURNAL OF NUMERICAL METHODS IN ENGINEERING 61, 2296–2315 (2003)
21. Schwefel, H.P.: Numerical Optimization of Computer Models. John Wiley & Sons, New-York (1981), 1995 – 2^{nd} edition
22. Shi, Y., Eberhart, R.C.: A Modified Particle Swarm Optimizer. In: Proceedings of IEEE International Conference on Evolutionary Computation. pp. 69–73. IEEE Computer Society, Washington, DC, USA (May 1998)
23. Storn, R., Price, K.: Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces. J. of Global Optimization 11(4), 341–359 (Dec 1997), http://dx.doi.org/10.1023/A:1008202821328
24. 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/
25. Trelea, I.C.: The particle swarm optimization algorithm: convergence analysis and parameter selection. Information Processing Letters 85(6), 317 – 325 (2003), http://www.sciencedirect.com/science/article/pii/S0020019002004477
26. Zambrano-Bigiarini, M., Clerc, M., Rojas, R.: Standard particle swarm optimisation 2011 at cec-2013: A baseline for future pso improvements. In: IEEE Congress on Evolutionary Computation. pp. 2337–2344. IEEE (2013), http://dblp.uni-trier.de/db/conf/cec/cec2013.html#Zambrano-BigiariniCR13

(a) DE outperforms everything for the specific policy

	$SAiso$	$(1+1)$	SA	$SACov$	CMA	NM	DE	PSO
$SAiso$		100.00	28.57	57.14	64.29	100.00	21.43	42.86
$(1+1)-ES$	0.00		0.00	0.00	7.14	14.29	7.14	0.00
$SA-ES$	71.43	100.00		57.14	71.43	92.86	14.29	50.00
$SA-ESCov$	42.86	100.00	42.86		78.57	92.86	21.43	50.00
$CMA-ES$	35.71	92.86	28.57	21.43		85.71	14.29	50.00
NM	0.00	85.71	7.14	7.14	14.29		0.00	0.00
DE	78.57	92.86	85.71	78.57	85.71	100.00		85.71
PSO	57.14	100.00	50.00	50.00	50.00	100.00	14.29	

(b) With the neural network, PSO is clearly the best algorithm

	$SAiso$	$(1+1)$	SA	$SACov$	CMA	NM	DE	PSO
$SAiso$		92.86	92.86	89.29	42.86	21.43	50.00	0.00
$(1+1)-ES$	7.14		46.43	50.00	35.71	10.71	3.57	0.00
$SA-ES$	7.14	53.57		75.00	28.57	10.71	10.71	0.00
$SA-ESCov$	10.71	50.00	25.00		28.57	10.71	10.71	0.00
$CMA-ES$	57.14	64.29	71.43	71.43		28.57	25.00	17.86
NM	78.57	89.29	89.29	89.29	71.43		78.57	3.57
DE	50.00	96.43	89.29	89.29	75.00	21.43		0.00
PSO	100.00	100.00	100.00	100.00	82.14	96.43	100.00	

(c) CMA is the best performing algorithm for Conformant Planning

	$SAiso$	$(1+1)$	SA	$SACov$	CMA	NM	DE	PSO
$SAiso$		100.00	71.43	71.43	0.00	42.86	7.14	100.00
$(1+1)-ES$	0.00		0.00	14.29	0.00	28.57	0.00	100.00
$SA-ES$	28.57	100.00		57.14	0.00	42.86	7.14	100.00
$SA-ESCov$	28.57	85.71	42.86		0.00	42.86	0.00	92.86
$CMA-ES$	100.00	100.00	100.00	100.00		100.00	64.29	100.00
NM	57.14	71.43	57.14	57.14	0.00		0.00	85.71
DE	92.86	100.00	92.86	100.00	35.71	100.00		100.00
PSO	0.00	0.00	0.00	7.14	0.00	14.29	0.00	

(d) For Fuzzy control, SA-iso is the best algorithm

	$SAiso$	$(1+1)$	SA	$SACov$	CMA	NM	DE	PSO
$SAiso$		100.00	100.00	100.00	100.00	100.00	92.86	100.00
$(1+1)-ES$	0.00		42.86	64.29	92.86	100.00	60.71	100.00
$SA-ES$	0.00	57.14		96.43	92.86	100.00	67.86	96.43
$SA-ESCov$	0.00	35.71	3.57		92.86	100.00	25.00	82.14
$CMA-ES$	0.00	7.14	7.14	7.14		71.43	0.00	28.57
NM	0.00	0.00	0.00	0.00	28.57		0.00	0.00
DE	7.14	39.29	32.14	75.00	100.00	100.00		96.43
PSO	0.00	0.00	3.57	17.86	71.43	100.00	3.57	

Table 2: Frequency (in percentage) where an algorithm (in row) outperforms another one (in column) in the small case (5 stocks, 25 timesteps).

(a) DE outperforms everything for the specific policy

	$SAiso$	$(1+1)$	SA	$SACov$	CMA	NM	DE	PSO
$SAiso$		100.00	50.00	64.29	64.29	100.00	7.14	42.86
$(1+1)-ES$	0.00		0.00	0.00	0.00	42.86	0.00	0.00
$SA-ES$	50.00	100.00		64.29	42.86	92.86	0.00	28.57
$SA-ESCov$	35.71	100.00	35.71		28.57	92.86	0.00	14.29
$CMA-ES$	35.71	100.00	57.14	71.43		100.00	0.00	21.43
NM	0.00	57.14	7.14	7.14	0.00		0.00	7.14
DE	92.86	100.00	100.00	100.00	100.00	100.00		92.86
PSO	57.14	100.00	71.43	85.71	78.57	92.86	7.14	

(b) With the neural network, PSO is clearly the best algorithm

	$SAiso$	$(1+1)$	SA	$SACov$	CMA	NM	DE	PSO
$SAiso$		75.00	82.14	78.57	10.71	3.57	35.71	0.00
$(1+1)-ES$	25.00		46.43	60.71	10.71	3.57	46.43	0.00
$SA-ES$	17.86	53.57		75.00	7.14	0.00	17.86	0.00
$SA-ESCov$	21.43	39.29	25.00		7.14	0.00	25.00	0.00
$CMA-ES$	89.29	89.29	92.86	92.86		21.43	60.71	10.71
NM	96.43	96.43	100.00	100.00	78.57		100.00	25.00
DE	64.29	53.57	82.14	75.00	39.29	0.00		7.14
PSO	100.00	100.00	100.00	100.00	89.29	75.00	92.86	

(c) DE is the best performing algorithm for Conformant Planning

	$SAiso$	$(1+1)$	SA	$SACov$	CMA	NM	DE	PSO
$SAiso$		100.00	64.29	71.43	64.29	92.86	35.71	64.29
$(1+1)-ES$	0.00		0.00	21.43	21.43	57.14	0.00	50.00
$SA-ES$	35.71	100.00		64.29	64.29	92.86	35.71	57.14
$SA-ESCov$	28.57	78.57	35.71		42.86	78.57	7.14	64.29
$CMA-ES$	35.71	78.57	35.71	57.14		64.29	28.57	42.86
NM	7.14	42.86	7.14	21.43	35.71		7.14	21.43
DE	64.29	100.00	64.29	92.86	71.43	92.86		64.29
PSO	35.71	50.00	42.86	35.71	57.14	78.57	35.71	

(d) For Fuzzy control, SA-iso is the best algorithm

	$SAiso$	$(1+1)$	SA	$SACov$	CMA	NM	DE	PSO
$SAiso$		89.29	89.29	100.00	100.00	100.00	89.29	96.43
$(1+1)-ES$	10.71		21.43	64.29	100.00	100.00	53.57	60.71
$SA-ES$	10.71	78.57		82.14	100.00	100.00	85.71	78.57
$SA-ESCov$	0.00	35.71	17.86		100.00	100.00	46.43	42.86
$CMA-ES$	0.00	0.00	0.00	0.00		42.86	0.00	7.14
NM	0.00	0.00	0.00	0.00	57.14		0.00	0.00
DE	10.71	46.43	14.29	53.57	100.00	100.00		53.57
PSO	3.57	39.29	21.43	57.14	92.86	100.00	46.43	

Table 3: Frequency (in percentage) where an algorithm (in row) outperforms another one (in column) in the medium case (15 stocks, 50 timesteps).