Optimization-Group Project $(\mu/\mu, \lambda) - ES$ **with Search Path** Intermediate report

Answers:
Chen Xiaoxiao, FEI Dong, LI Honglin,
WANG Yuxiang

October 10, 2016

Abstract

This document details the intermediate results of our group project, including the up-to-date outcomes and the prospecting work.$(\mu/\mu, \lambda)-ES$

1 presentation of the algorithm

Inspired by biological evolution, the basic formulation of our algorithm is based on the application of mutation, recombination and selection in populations of candidate solution. When the result meets the predefined criterion, we consider achieving the expected solution. $(\mu/\mu, \lambda) - ES$ with Search Path uses self-adaptation step-size with an additional search path control.

2 Intermediate outcomes

2.1 implementation

We chose to implement this algorithm with language Python. To verify that the algorithm is working properly, we tried to realize the algorithm first in a simple way and did the unit test. Then under the coco framework, the ES with search path algorithm was implemented and test for a dimension of 20.

All the source codes and first benchmarking results are available on the site *github* under the project nbacxx23/ES-with-Search-Path

2.2 Parameters

- P a multiset of individuals, a population
- $\lambda \in \mathbb{N}$ number of off-springs

- $\mu \in \mathbb{N}$ number of parents
- $(lbounds, ubounds)$,search range $lbounds, ubounds \in \mathbb{R}$
- $\sigma \in \mathbb{R}^n_+$coordinate wise standard deviation (step size)
- $x\_final \in \mathbb{R}^n$ the final solution vector after the optimization process
- $x\_k \in \mathbb{R}^n$ the feasible solutions for each loop
- $E\_half\_normal\_dis$ expectation of half normal distribution
- E_muldim_normal estimation of the expectation of $\|N(0, I)\|$
- *budget* parameter given, the max number of loops
- $\boxed{\textit{happy} \text{ function to define whether the solution is acceptable}}$
- $\boxed{\textit{sel\_u\_best} \text{ function to select } \mu \text{ best solution}}$
- $\boxed{\textit{ES\_search\_path} \text{ the main algorithm}}$

2.3 Stopping criteria

In our algorithm, there are two stopping criteria:

- The function **def happy(x,x_final)** tells if the precision criterion is satisfied (return 1) or not (return 0). If the distance between two consecutive searches is less than e, it will be accepted. It makes sure whether the solutions convergence.
- The loop while stops if the budget = 0, which is the time stopping criterion in case of infinite iterations.

2.4 Performance on unit test

To verify the performance of the algorithm, a unit test is conducted. The objective function is

$$f(x) = 5x^2 + 8x + 25$$

with the initial $x = 45$, satisfying the precision of $e = 0.00000001$. We choose this function because is it easily calculated and hence to evaluate the performance of the algorithm.

In this unit test, the result is optimized in 100 steps, while the $f\_min = 21.800174433$ and $x\_min = -0.794093512284$. Given the analytic solution $f\_min = 21.8$ and $x\_min = -0.8$. In the Figure 1, $f(x)$ is convergent mainly at step 65.

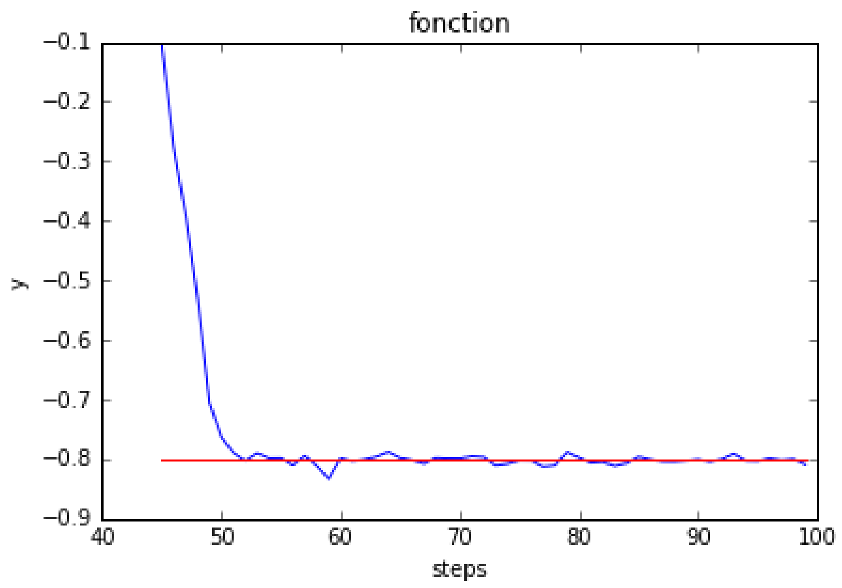

Figure 1: The result of the unit test

The result of the Benchmarking on the COCO plateform will be further analysed in the future work.

3 Remaining work

- Benchmarking intensively our algorithm with the COCO platform
- Improve the given parameter σ
- Increase the budget in experiment script
- Implement randomized independent restarts (if possible)
- Comprehend in-deep the theoretical part and understand better how to interpret coco's test results

3.1 Questions

- What are the given parameters of the objective function in coco?
- The solver class given by coco framework is wrapped, we are still a little confused with its interface.
- What is the best strategy of choosing the initial step-size compared with the search range?