Algorithm Design in Evolutionary Computation: Parameter Identification and Control

N. Hansen
General thoughts on parameters

• Any algorithm has an arbitrary number of parameters
  for example: think of all the hidden “1”s parameters can be used to toggle any algorithms

• First task: agree on the relevant parameters
  this is a non-trivial task on-the-fly definition for relevant: a non-trivial tuning improves the performance (sometimes) by more than a factor of two
Classification of parameters

- **Off-line parameter identification** during the algorithm design
  e.g. based on benchmarks or the designers guts

- **Trial-and-error** parameter setting by the user
  not well-defined (this is not an algorithm)

- **Setting** prior to application, based on known or measured problem properties (features)
  e.g. using the problem dimension

- **On-line adaptation**
  - Restarts with different parameters (can replace a user's trial-and-error procedure)
  - Self-adaptation, CMA, reinforcement, ...

The latter two are an integral part of the algorithm itself

Tuning methods depend essentially on the “class”
Example for off-line identification

- Learning rate \( ccov \) for covariance matrix adaptation (CMA)
  - Using the most simple function that seems to make sense: sphere function \( \sum_{i} x_i^2 \) with initially anisotropic covariance matrix
  - Invariance properties and further empirical evidence (in the sense of hypothesis testing) suggest generality of the results

\[
C \leftarrow (1 - ccov) C + ccov \sum_{i}^{\mu} z_i z_i^T
\]

\[
= C + ccov \times \left( \sum_{i}^{\mu} z_i z_i^T - C \right)
\]
• Trade off: robustness (small learning rate ccov to the left) versus speed (large ccov). Lines (would) smoothly continue to the left.

• Remark: x-axis presentation
Choosing the parameter

- **Robustness** (required):
  - increasing ccov three times (faster learning) never leads to a failure
  - increasing ccov two times is better than decreasing ccov two times

- **Performance** (desirable): the performance loss is less than a factor of three
  - for ccov=0 the loss factor is roughly 1000

- **Adaptation quality** (learning accuracy, related to robustness): final condition number of the covariance matrix is smaller than ten.

This (a similar) list works in many cases cf. [Brockhoff et al 2010. Mirrored sampling and sequential selection for evolution strategies, PPSN XI]
CMA-ES on the sphere function

Object Variables, recent (10-D)

Principle Axes Lengths

Standard Deviations in Coordinates divided by sigma

Nikolaus Hansen
CMA-ES on the sphere function

Principle Axes Lengths

function evaluations

Nikolaus Hansen
Axis ratio of mutation ellipsoid = sqrt(condition number of covariance matrix)
Final result for ccov

- Identification needs to be done depending on the population size
  - more precisely: the amount of input information
- A trial-and-error process leads to

\[
ccov = 2 \frac{\mu_{\text{eff}} - 1 + 1/\mu_{\text{eff}}}{(n + 2)^2 + \mu_{\text{eff}}}
\]

- Inverse proportional to the degrees of freedom
- Proportional to the amount of input information
- Correction for small values
Off-line identification = algorithm design

• Major pitfall: we must seek for good generalization performance, i.e. prevent overtuning to test problems (= design for a too narrow problem class)

  an optimal parameter setting on a single problem is rarely of practical interest

• Using a single function works(!) and leaves many functions for hypothesis testing
Example: Evolution Path

- An evolution path (search path) is extensively used for on-line parameter adaptation in CMA-ES.

Conceptually, the path is the difference vector of the evolving population mean $m$.

<table>
<thead>
<tr>
<th>short</th>
<th>expected</th>
<th>long</th>
</tr>
</thead>
</table>

Evolution (search) paths with cumulation of six time steps in 2-D.
Example: Evolution Path

- An evolution path (search path) is extensively used for on-line parameter adaptation in CMA-ES conceptually, the path is the difference vector of the evolving population mean $m$

- Question: How many time steps should an evolution path comprise?

$$p \leftarrow (1 - c) p + \sqrt{c(2 - c)} \Delta m^t$$

$$\approx \sqrt{c} (m^t - m^{t-[1/c]}) =: \frac{1}{\sqrt{t_c}} (m^t - m^{t-t_c})$$

$1/c$ is roughly the effective number of time steps
Cumulation time horizon

\[ f(x) = x_1 + 10^6 \sum_{i=2}^{n} x_i^2 \]

The cigar function shows the most pronounced effect on cumulation.

#Fevals divided by dimension in 3,10,30,100-D
# Fevals divided by dimension in 3, 10, 30, 100-D
now the graphs are virtually invariant under the choice of the dimension

final choice:

\[
\frac{1}{c} = \frac{n + 4 + 2\mu_{\text{eff}}/n}{4 + \mu_{\text{eff}}/n}
\]
On-line adaptation

Another name for control

• Controlled variable(s), caveat: parametrization / representation
  – Step-size
  – Population size
  – Covariance matrix with many degrees of freedom
  – Mutation/recombination rate...

• Update rule, “depends” on representation

• Input variable(s) (used information)
  – survival probability for different settings
  – success, success rate, or progress measurement
  – evolution path, length of evolution path...
What is the difference between a state variable and an adaptive parameter?
Example: CMA-ES

Input: \( m \in \mathbb{R}^n, \sigma \in \mathbb{R}_+, \lambda \in \{2, 3, 4, \ldots \} \)

Set \( c_c \approx \frac{4}{n}, \ c_\sigma \approx \frac{4}{n}, \ c_1 \approx \frac{2}{n^2}, \ c_\mu \approx \frac{\mu_w}{n^2}, \ c_1 + c_\mu \leq 1, \)
\( d_\sigma \approx 1 + \sqrt{\frac{\mu_w}{n}}, \) set \( w_{i=1,\ldots,\lambda} \) such that \( \mu_w = \frac{1}{\sum_{i=1}^\mu w_i^2} \approx 0.3 \lambda \)

Initialize \( \mathbf{C} = \mathbf{I}, \) and \( \mathbf{p}_c = 0, \mathbf{p}_\sigma = 0 \)

While not terminate

\[ \mathbf{x}_i = m + \sigma \mathbf{y}_i, \quad \mathbf{y}_i \sim \mathcal{N}_i(0, \mathbf{C}) \text{, for } i = 1, \ldots, \lambda \] sampling

\[ m \leftarrow \sum_{i=1}^\mu w_i \mathbf{x}_{i:\lambda} = m + \sigma \mathbf{y}_w \] update mean

\[ \mathbf{p}_\sigma \leftarrow (1 - c_\sigma) \mathbf{p}_\sigma + \sqrt{1 - (1 - c_\sigma)^2} \sqrt{\mu_w} \mathbf{C}^{-\frac{1}{2}} \mathbf{y}_w \] path for \( \sigma \)

\[ \sigma \leftarrow \sigma \times \exp \left( \frac{c_\sigma}{d_\sigma} \left( \frac{\| \mathbf{p}_\sigma \|}{\mathbb{E}[\mathcal{N}(0, \mathbf{I})]} - 1 \right) \right) \] update of \( \sigma \)

\[ \mathbf{p}_c \leftarrow (1 - c_c) \mathbf{p}_c + \mathbb{I}_{[0,1.5]}(\frac{\| \mathbf{p}_\sigma \|}{\sqrt{n}}) \sqrt{1 - (1 - c_c)^2} \sqrt{\mu_w} \mathbf{y}_w \] path for \( \mathbf{C} \)

\[ \mathbf{C} \leftarrow (1 - c_1 - c_\mu) \mathbf{C} + c_1 \mathbf{p}_c \mathbf{p}_c^T + c_\mu \sum_{i=1}^\mu w_i \mathbf{y}_{i:\lambda} \mathbf{y}_{i:\lambda}^T \] update \( \mathbf{C} \)
Population size adaptation

- Can it work reliably?

multimodality as counterexample

Rastrigin function in 10-D

Population size 10 and 200

The advantage of the large population becomes visible only way after the run with small population size finished

Is a widely suitable input variable conceivable?
Population size adaptation

- Can it work reliably?

multimodality as counterexample

Schaffer function in 10-D

Population size 10 and 200

The advantage of the large population becomes visible only way after the run with small population size finished

Is a widely suitable input variable conceivable?

It would need to be able to “predict” the late outcome of a run very early
Population size adaptation

A solution

- Conduct several runs with different population sizes
  - Serially with increasing population size [Harik&Lobo 1999], [Auger&Hansen 2005]
  - Parallel (race) with different population sizes [Harik&Lobo 1999], [Hornby 2006]
Summary

1) Remove users trial-and-error procedure from the possibilities (in a scientific context)
   
   the algorithm designer should instead suggest a sequence of settings

2) Parameter identification in the context of algorithm design is difficult and tedious
   
   but it is worth the time

3) Parameter identification can be done successfully on a single function

4) The population size is hard to adapt
Thank you