Automatic and Interactive Tuning of Algorithms

Thomas Bartz-Beielstein\textsuperscript{1} Mike Preuss\textsuperscript{2}

\textsuperscript{1}Faculty of Computer Science and Engineering Science
Cologne University of Applied Sciences

\textsuperscript{2}Department of Computer Science
TU Dortmund

July 2011
Agenda

1. Introduction
2. SPO Toolbox (SPOT)
3. Further Approaches
4. Experimental Primer
Your Instructors Today

- Dr. Thomas Bartz-Beielstein is a professor for Applied Mathematics at Cologne University of Applied Sciences. He has published more than several dozen research papers, presented tutorials about tuning, and has edited several books in the field of Computational Intelligence. His research interests include optimization, simulation, and statistical analysis of complex real-world problems.

- Prof. Bartz-Beielstein and Mike Preuss invented the sequential parameter optimization, which was applied as a tuner for numerous optimization algorithms such as evolution strategies, differential evolution, or particle swarm optimization.

- Mike Preuss is research associate at the Computer Science Department, TU Dortmund. His main fields of activity are EAs for real-valued problems and their application in numerous engineering domains, the development of the experimental methodology for stochastic optimization, and Computational Intelligence techniques in computer games (see DETA track).
Objectives of the Tutorial

(O-1) *Tuning.* Making your algorithms faster (and more reliable)

(O-2) *Understanding and Learning.* Helping you to understand your algorithms (so that forthcoming versions will run even much faster)

(O-3) *Provide Experimental Guidelines.* Enables you to perform *solid* experiments one can learn from where theory is not applicable

(O-3) *Pros and Cons.* Consider benefits and disadvantages of state-of-the-art tuning approaches

(O-4) *Networking.* Meet and get into contact with others who are interested in tuning. Discuss open issues and interesting research projects in experimental research
Why Do We Need Experimentation?

- Practitioners need to solve problems, even if theory is not developed far enough
- Counterargument of practitioners: Tried that once, didn’t work (expertise needed to apply convincingly)
- We need to establish guidelines how to adapt the algorithms to practical problems (or let them Self *)
- Helps theoreticians to find exploitable (parameter/problem) relations

Experimental methodology is improving, we are leaving the phase of

a) Funny but useless performance figures
b) Lots of better and better algorithms that soon disappear again
Why Experimentation?

Why Do We Need Experimentation?

- Practitioners need so solve problems, even if theory is not developed far enough
- Counterargument of practitioners: Tried that once, didn’t work (expertise needed to apply convincingly)
- We need to establish guidelines how to adapt the algorithms to practical problems (or let them Self*)
- Helps theoreticians to find exploitable (parameter/problem) relations

Instead, we converge to

a) Deliberate and justified choice of parameters, problems, performance criteria—no more arbitrariness

b) Better generalizability (not quite resolved, but targetted)
Are We Alone (With This Problem)?

In natural sciences, experimentation is not in question

- Many inventions (batteries, x-rays, ... ) made by experimentation, sometimes unintentional
- Experimentation leads to theory, theory has to be useful (can we do predictions?)

In computer science, the situation seems different

- 2 widespread stereotypes influence our view of computer experiments:
  a) Programs do (exactly) what algorithms specify
  b) Computers (programs) are deterministic, so why statistics?

This is an experiment

Is this an experiment?
Algorithm Engineering
How Theoreticians Handle it...(Recently)

- Algorithm Engineering is *theory* + real data + concrete implementations + experiments
- Principal reason for experiments: Test validity of theoretical claims
- Are there important factors in practice that did not go into theory?
- Approach also makes sense for metaheuristics, but we start with no or little theory
- Measuring (counting evaluations) usually no problem for us
So What About Statistics?

Are the methods all there? Some are, but:

- Our data is usually not normal
- We can most often have lots of data
- This holds for algorithmics, also!
- These are not the conditions statisticians are used to
- In some situations, there is just no suitable test procedure

⇒ There is a need for more statistics and more statistical methods.

Catherine McGeogh:
*Our problems are unfortunately not sexy enough for the Statisticians...*
Algorithms, Parameters and the Reasoning for Tuning

- We have learned to think in parameters where others hide these as constants
- Knowledge about parameter interactions helps us to understand algorithms and problems
- This enables approaching the final question: Which algorithm (or which parameter set) do I apply for my problem?

But:
- How do we do it?
- Tuning is expensive, cannot be applied in all situations
- And what about the problems?

⇒ Experimental methodology and reliable tuning methods needed
First step: Archeology—Detect Factors

- “Playing trumpet to tulips” or “experimenter’s socks”
- Everything is in our hand: We need to carefully design what we want
- Factors in algorithms: Parameters
- Factors in problems? Not much work yet (global fitness distance correlation (FDC) doesn’t work)
- Revived interest: black-box optimization benchmark (BBOB), exploratory landscape analysis (ELA)

Figure: Schliemann in Troja
Components of an Experiment in Metaheuristics

- Algorithm (program)
- Parameter set
- Test problem
- Performance measure
- Termination criterion
- Initialization
- Control flow
- Data flow
- Problem design
Factors: Overview

- Java version
- Hardware
- Color of experimenter's socks
- Weather

**EXPECTED**
- Algorithm design
- Problem design
- Operating system

**POSSIBLE, UNWANTED**
- Color of experimenter's socks
- Weather
- Room temperature

**UNEXPECTED**
What is Exploratory Landscape Analysis (ELA)?

- At first the attempt to obtain problem knowledge (see Mersmann/Trautmann/Bischl/Preuss PPSN’10 / GECCO’11)
- Defines high-level (experience based) and low-level (mathematical) features
- Final goal to recommend matching algorithm on the base of problem features (groups of problems)
- Which features are suitable? Still much work to do...

1. Multimodality
   - $p < 0.001$
   - High: $n = 27$, $y = (0, 1)$

2. Variable scaling
   - $p = 0.002$
   - Low: $n = 3$, $y = (0, 1)$

3. Global structure
   - $p = 0.012$
   - Low: $n = 12$, $y = (0.083, 0.917)$

4. Deceptive
   - $n = 30$, $y = (0.833, 0.167)$

5. Multimodality
   - $n = 30$, $y = (0.833, 0.167)$

6. Deceptive
   - $n = 3$, $y = (0, 1)$

7. Global structure
   - $n = 12$, $y = (0.083, 0.917)$

Bartz-Beielstein, Preuss (Cologne, Dortmund)
Research Question

- Not trivial \(\Rightarrow\) many papers are not focused
- The (real) question is not: Is my algorithm faster than others on a set of benchmark functions?
- What is the added value? Difficult in Metaheuristics.
  - Wide variance of treated problems
  - Usually (nearly) black-box: Little is known

*Horse racing*: set up, run, comment...

Explaining observations leads to new questions:
- Multi-step process appropriate (also for tuning)
- Conjectures obtained from results shall itself be tested experimentally
- Range of validity shall be explored (problems, parameters, etc.)
Research Question

- Not trivial \(\Rightarrow\) many papers are not focused
- The (real) question is not: Is my algorithm faster than others on a set of benchmark functions?
- What is the added value? Difficult in Metaheuristics.
  - Wide variance of treated problems
  - Usually (nearly) black-box: Little is known

_Horse racing:_ set up, run, comment... _NO!_

Explaining observations leads to new questions:
- Multi-step process appropriate (also for tuning)
- Conjectures obtained from results shall itself be tested experimentally
- Range of validity shall be explored (problems, parameters, etc.)
How to Set Up Research Questions?

*What do We Aim For?*

It is tempting to create a new algorithm, but

- There are many existing algorithms not really understood well
- We shall try to aim at improving our knowledge about the ‘working set’
- Problem/algorithm interactions very interesting!
- When comparing, always ask if any difference is meaningful in practice

Usually, we do not know the ‘perfect question’ from the start

- An inherent problem with experimentation is that we do (should) not know the outcome in advance
- But it may lead to new, better questions
- Try small steps, expect the unexpected
Problems and Algorithms

- How to perform comparisons?
- Adequate statistics?
SASP – Single Algorithm, Single Problem

real-world setting

determine important factors

optimization

crucial: number of function evaluations

benefit: $\varepsilon$

Tuning

single algorithm
single problem

single algorithm
multiple problems

multiple algorithms
single problems

multiple algorithms
multiple problems
SAMP – Single Algorithm, Multiple Problems

Tuning

single algorithm
single problem

multipl algorithm
single problems

algorithm development
optimization
robustness
important factors
understanding
benefit

multiple algorithms
multiple problems

Bartz-Beielstein, Preuss (Cologne, Dortmund)
MASP – Multiple Algorithms, Single Problem

- Tuning
- multiple algorithms
- single problems

- research
- optimization
- similarites between algorithms
- benefit
- understanding

- single algorithm
  - single problem
  - multiple problems

- multiple algorithms
  - one algorithm, but different parameters
  - tuning and comparison

- beginner’s paper => rejected
MAMP – Multiple Algorithms, Multiple Problems

Tuning

- single algorithm
  - single problem
- multiple algorithms
  - single problems
- multiple algorithms
  - multiple problems

benefit:
- comparison
- huge complexity

research

expert paper => accepted
SPOT

- *Sequential parameter optimization toolbox* (SPOT)
- Developed over recent years by Thomas Bartz-Beielstein, Christian Lasarczyk, and Mike Preuss [BBLP05]
- Main goals of SPOT
  - Determination of improved parameter settings for optimization algorithms
  - Provide statistical tools for analyzing and understanding their performance
- Package can be downloaded from the comprehensive R archive network at [http://CRAN.R-project.org/package=SPOT](http://CRAN.R-project.org/package=SPOT)
<table>
<thead>
<tr>
<th><strong>SPOT-1</strong></th>
<th>Use the available budget (e.g., simulator runs, number of function evaluations) <em>sequentially</em>:</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Use information from the exploration of the search space to guide the search by building one or several meta models</em></td>
<td></td>
</tr>
<tr>
<td><em>Choose new design points based on predictions from the meta model(s)</em></td>
<td></td>
</tr>
<tr>
<td><em>Refine the meta model(s) stepwise to improve knowledge about the search space</em></td>
<td></td>
</tr>
<tr>
<td><strong>SPOT-2</strong></td>
<td>If necessary, try to cope with <em>noise</em> by improving confidence. Guarantee comparable confidence for search points</td>
</tr>
<tr>
<td><strong>SPOT-3</strong></td>
<td>Collect information to <em>learn</em> from this tuning process, e.g., apply explorative data analysis</td>
</tr>
<tr>
<td><strong>SPOT-4</strong></td>
<td>Provide mechanisms both for <em>interactive</em> and <em>automated tuning</em></td>
</tr>
</tbody>
</table>
SPOT Applications

- SPOT was successfully applied in the fields of
  - bioinformatics [Vol06, FMKH09]
  - environmental engineering [KZBB09, FBBD+10]
  - fuzzy logic [Yi08]
  - multimodal optimization [PRT07]
  - statistical analysis of algorithms [Las07, TM09]
  - multicriteria optimization [BBNWW09]
  - genetic programming [LB05]
  - particle swarm optimization [BBPV04, KGG07]
  - automated and manual parameter tuning [Fob06, SE09, HBBH+09, HHLBM10]
  - graph drawing [Tos06, Pot07]
  - aerospace and shipbuilding industry [NQBB06, RPQ09]
  - mechanical engineering [MMLBB07]
  - chemical engineering [HBK+08]

- Bartz-Beielstein [BB10] collects publications related to the sequential parameter optimization
SPOT Tasks

- SPOT provides tools to perform the following tasks:
  - *Initialize*. Generate an initial design: parameter region and constant algorithm parameters
  - *Run*. Start optimization algorithm with configurations of the generated design. The algorithm provides the results to SPOT
  - *Sequential step*. Generate a new design, based on information from the algorithms result. A prediction model is used in this step. Several generic prediction models are available in SPOT already. User-specified prediction models can easily be integrated
  - *Report*. Generate an analysis, based on information from the results. SPOT contains some scripts to perform a basic regression analysis and plots such as histograms, scatter plots, plots of the residuals, etc.
  - *Automatic* mode. In the automatic mode, the steps *run* and *sequential* are performed after an initialization for a predetermined number of times.
Factors, Designs and Predictors

- **Factors**
  - Numerical
  - Categorical (ordered and unordered)

- **Designs**
  - Classical fractional factorial designs
  - Space-filling designs, e.g., Latin hypercube designs

- **Predictors**
  - Linear regression
  - Regression trees
  - Tree based Gaussian process
  - Predictors for very expensive optimization runs (real-world problems)
    - Tree based Gaussian processes
    - DACE
  - Predictors for simple optimization runs
    - Classical regression models
    - Regression trees
  - Combinations are possible ⇒ Ensemble-based modeling, Meta predictors
SASP – Single Algorithm, Single Problem. 1+1 ES

test for termination

cloning

initialization and evaluation

evaluation

environmental selection

replacement

mutation
SASP – Single Algorithm, Single Problem

- Simple *evolution strategy* (ES), the so-called (1+1)-ES.

\[
\begin{align*}
t &:= 0; \\
\text{initialize}(\vec{x}, \sigma); \\
y_p &:= f(\vec{x}_p); \\
\text{repeat} & \\
\vec{x}_o &:= \vec{x}_p + \sigma (\mathcal{N}(0, 1), \mathcal{N}(0, 1), \ldots, \mathcal{N}(0, 1))^T; \\
y_o &:= f(\vec{x}_o); \\
\text{if } y_o \leq y_p \text{ then} & \\
\quad \vec{x}_p &:= \vec{x}_o; \\
\quad y_p &:= y_o; \\
\text{end} & \\
\text{modify } \sigma \text{ according to 1/5th rule}; \\
t &:= t + 1; \\
\text{until } \text{TerminationCriterion}(); \\
\text{return } (\vec{x}_p, y_p)
\end{align*}
\]
SASP – The 1/5th Rule

- The 1/5th rule states that $\sigma$ should be modified according to the rule

$$
\sigma(t + 1) := \begin{cases} 
\sigma(t)a, & \text{if } P_s > 1/5 \\
\sigma(t)/a, & \text{if } P_s < 1/5 \\
\sigma(t), & \text{if } P_s = 1/5 
\end{cases}
$$

(1)

- Factor $a$ is usually between 1.1 and 1.5 and $P_s$
- denotes the success rate [Bey01].
- Factor $a$ depends particularly on the measurement period $g$, which is used to estimate the success rate $P_s$.
- During the measurement period, $g$ remains constant.
- For $g = n$, where $n$ denotes the problem dimension, [Sch95] calculated $1/a \approx 0.817$.
- [Bey01] states that the “choice of $a$ is relatively uncritical” and that the 1/5th rule has a “remarkable validity domain.”
- Based on these theoretical results, we can derive certain scientific hypotheses: Given a spherical fitness landscape, the (1+1)-ES performs optimally, if the step-sizes $\sigma$ is modified according to the 1/5th rule as stated in Eq. 1.
SASP – The 1/5th Rule

1+1 Evolution Strategy

- If success rate is high, increase step size
- If success rate is low, decrease step size

1/5th rule

- initial step size
- measurement period
- multiplier
- SIGMANULL
- VARA
- VARG
SASP – JAVA Implementation

- The (1+1)-ES can be started using the jar file from the command line with the following arguments.

```java
java -jar simpleOnePlusOneES.jar
123
500
1.0E-10
de.fhkoeln.spot.objectivefunctions.Ball
10
"[10.0,10.0,10.0,10.0,10.0,10.0,10.0,10.0,10.0,10.0]"
1.0
1.2
10
0
2
```
This algorithm run produces the following output:

<table>
<thead>
<tr>
<th>n</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>914.1626739273763</td>
</tr>
<tr>
<td>6</td>
<td>862.8800439118534</td>
</tr>
<tr>
<td>8</td>
<td>678.7699432192874</td>
</tr>
<tr>
<td>10</td>
<td>658.5300029606245</td>
</tr>
<tr>
<td>12</td>
<td>507.565852817427</td>
</tr>
<tr>
<td>23</td>
<td>414.84675604291283</td>
</tr>
<tr>
<td>25</td>
<td>354.1769352944835</td>
</tr>
<tr>
<td>29</td>
<td>354.00455231919966</td>
</tr>
<tr>
<td>35</td>
<td>352.00724280533336</td>
</tr>
<tr>
<td>47</td>
<td>343.8357239425409</td>
</tr>
<tr>
<td>48</td>
<td>308.87305308760915</td>
</tr>
<tr>
<td>49</td>
<td>263.5663685407355</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>488</td>
<td>1.0013911476017589E-4</td>
</tr>
<tr>
<td>491</td>
<td>8.209403534791058E-5</td>
</tr>
<tr>
<td>498</td>
<td>6.647907206024157E-5</td>
</tr>
</tbody>
</table>
SASP – Experimental Model

- Define objective function, starting point, quality measure, and parameters used by the algorithm. Quality measure is defined in the CONF file.

<table>
<thead>
<tr>
<th>Name</th>
<th>Symbol</th>
<th>Factor name (algorithm design)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial stepsize</td>
<td>$\sigma(0)$</td>
<td>SIGMANULL</td>
</tr>
<tr>
<td>Stepsize multiplier</td>
<td>$a$</td>
<td>VARA</td>
</tr>
<tr>
<td>History</td>
<td>$g = n$</td>
<td>VARG</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Symbol</th>
<th>Name in the APD file</th>
</tr>
</thead>
<tbody>
<tr>
<td>Starting point</td>
<td>$\hat{x}_p$</td>
<td>xp0</td>
</tr>
<tr>
<td>Problem dimension</td>
<td>$n$</td>
<td>n</td>
</tr>
<tr>
<td>Objective function</td>
<td>$f(\bar{x}) = \sum x_i^2$</td>
<td>f</td>
</tr>
<tr>
<td>Quality measure</td>
<td>Expected performance, e.g., $E(y)$</td>
<td>-</td>
</tr>
<tr>
<td>Initial seed</td>
<td>$s$</td>
<td>seed</td>
</tr>
<tr>
<td>Budget</td>
<td>$t_{max}$</td>
<td>steps</td>
</tr>
</tbody>
</table>
## SASP – JAVA Implementation

<table>
<thead>
<tr>
<th>Name</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>seed</td>
<td>random seed (123)</td>
</tr>
<tr>
<td>steps</td>
<td>maximum number of evolution steps (500)</td>
</tr>
<tr>
<td>target</td>
<td>objective function threshold for preliminary evolution end (1e-10)</td>
</tr>
<tr>
<td>f</td>
<td>objective function class name (de.fhkoeln.spot.objectivefunctions.Ball)</td>
</tr>
<tr>
<td>n</td>
<td>problem dimension (10)</td>
</tr>
<tr>
<td>xp0</td>
<td>starting point (uniform = uniformly distributed random vector from [0.0, 1.0]^n, gaussian = normally distributed random vector from N(0,1), c(xp0_0, ..., xp0_n) = the vector [xp0_0, ..., xp0_n])</td>
</tr>
<tr>
<td>sigma0</td>
<td>initial step size (1.0)</td>
</tr>
<tr>
<td>a</td>
<td>step size multiplier (1.2)</td>
</tr>
<tr>
<td>g</td>
<td>history length (10)</td>
</tr>
<tr>
<td>px, py</td>
<td>individual printing mode (verbosity)</td>
</tr>
</tbody>
</table>
SPOT: Steps

- **Budget**
  - Mean
  - Median

- **Quality Measure**

- **Parameter**
  - Ranges
  - Types

- **Problem**
  - Objective Function
  - Dimension
  - Starting Point
  - Function Evaluations (Budget)

- **Algorithm**
  - Interface
  - Command Line
  - Configuration File
  - R Function

- **ROI**

- **CONF**
  - Budget
  - Quality Measure

- **SPOT**
  - Steps
SASP: APD File

px = 0
py = 1
steps = 100
target = 1e-10
f = "de.fhkoeln.spot.objectivefunctions.Ball"
n = 10
xp0 = "[1.0,0.0,1.0,0.0,1.0,0.0,1.0,0.0,1.0,0.0,1.0,0.0]"
seed = 123
sigma0 = 1
a = 1.2
g = 10
SASP: ROI File

<table>
<thead>
<tr>
<th>name</th>
<th>low</th>
<th>high</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIGMANULL</td>
<td>0.1</td>
<td>5</td>
<td>FLOAT</td>
</tr>
<tr>
<td>VARA</td>
<td>1</td>
<td>2</td>
<td>FLOAT</td>
</tr>
<tr>
<td>VARG</td>
<td>2</td>
<td>100</td>
<td>INT</td>
</tr>
</tbody>
</table>
SASP: Config File

```plaintext
alg.path="bin"
alg.func = "spotAlgStartOnePlusOneEsJava"
auto.loop.nevals = 50;
```
library(SPOT)
spot("sasp.conf","auto")

- rep generates report
SPOT report. Linear model. First order

Coefficients:

|            | Estimate | Std. Error | t value | Pr(>|t|) |
|------------|----------|------------|---------|----------|
| (Intercept)| 0.25718  | 0.04169    | 6.168   | 4.14e-07 *** |
| x1         | 0.09472  | 0.07141    | 1.326   | 0.193    |
| x2         | 0.52360  | 0.08231    | 6.362   | 2.29e-07 *** |
| x3         | 0.09960  | 0.09374    | 1.063   | 0.295    |

---

Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2482 on 36 degrees of freedom
Multiple R-squared: 0.5948, Adjusted R-squared: 0.5611
F-statistic: 17.62 on 3 and 36 DF,  p-value: 3.322e-07
SPOT: Linear model, first order

Direction of steepest ascent (at radius 1):

\[
\begin{array}{ccc}
    x1 & x2 & x3 \\
    0.1749637 & 0.9672308 & 0.1839900 \\
\end{array}
\]

Corresponding increment in original units:

\[
\begin{array}{ccc}
    \text{SIGMANULL} & \text{VARA} & \text{VARG} \\
    0.4286612 & 0.4836154 & 45.8135203 \\
\end{array}
\]
SPOT: Linear model, first order

\[ x_1 = \frac{(\text{SIGMANULL} - 2.55)}{2.45} \quad \text{Slice at } x_3 = -0.02 \]

\[ x_2 = \frac{(\text{VARA} - 1.5)}{0.5} \quad \text{Slice at } x_1 = 0 \]

\[ x_3 = \frac{(\text{VARG} - 251)}{249} \quad \text{Slice at } x_2 = -0.17 \]
### SPOT report. Linear model. First order interactions

Coefficients:  

| Estimate | Std. Error | t value | Pr(>|t|) |
|----------|------------|---------|----------|
| Intercept| 0.22034    | 0.03889 | 5.666    | 2.58e-06 *** |
| x1       | 0.10907    | 0.06571 | 1.660    | 0.10643       |
| x2       | 0.51567    | 0.07511 | 6.866    | 7.74e-08 ***   |
| x3       | 0.04128    | 0.08381 | 0.493    | 0.62560       |
| x1:x2    | 0.18405    | 0.14559 | 1.264    | 0.21502       |
| x1:x3    | 0.02256    | 0.17965 | 0.126    | 0.90080       |
| x2:x3    | 0.47349    | 0.16212 | 2.921    | 0.00626 **     |

---

Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2164 on 33 degrees of freedom  
Multiple R-squared: 0.7177, Adjusted R-squared: 0.6664  
F-statistic: 13.99 on 6 and 33 DF,  p-value: 7.525e-08
SPOT: Linear model, first order interactions

\[
x_1 = \frac{\text{SIGMANULL} - 2.55}{2.45}
\]
Slice at \(x_3 = -0.02\)

\[
x_2 = \frac{\text{VARA} - 1.5}{0.5}
\]
Slice at \(x_1 = 0\)

\[
x_3 = \frac{\text{VARG} - 251}{249}
\]
Slice at \(x_2 = -0.17\)
SPOT report. Linear model. Second order

Coefficients:

|                | Estimate | Std. Error | t value | Pr(>|t|) |
|----------------|----------|------------|---------|----------|
| (Intercept)    | 0.11276  | 0.07646    | 1.475   | 0.150687 |
| x1             | 0.03034  | 0.06139    | 0.494   | 0.624830 |
| x2             | 0.52373  | 0.07359    | 7.117   | 6.47e-08 *** |
| x3             | 0.13408  | 0.07551    | 1.776   | 0.085957 . |
| x1:x2          | 0.03211  | 0.14090    | 0.228   | 0.821292 |
| x1:x3          | 0.08633  | 0.16963    | 0.509   | 0.614518 |
| x2:x3          | 0.33404  | 0.15091    | 2.213   | 0.034611 * |
| x1^2           | 0.01888  | 0.12087    | 0.156   | 0.876895 |
| x2^2           | 0.53400  | 0.14064    | 3.797   | 0.000665 *** |
| x3^2           | -0.17550 | 0.14844    | -1.182  | 0.246387 |

Residual standard error: 0.1836 on 30 degrees of freedom
Multiple R-squared: 0.8153, Adjusted R-squared: 0.7599
F-statistic: 14.72 on 9 and 30 DF,  p-value: 9.303e-09
SPOT: Linear model, second order

- $x_1 = \frac{(\text{SIGMANULL} - 2.55)}{2.45}$
- $x_2 = \frac{(\text{VARA} - 1.5)}{0.5}$
- $x_3 = \frac{(\text{VARG} - 251)}{249}$

Slices at:
- $x_3 = -0.02$
- $x_2 = -0.17$
- $x_1 = 0$
SPOT: Linear model. Added variable plots
SPOT: Linear model. LASSO
Bad news?

- adequate model
- normality
- lack of fit
- extrapolation
SPOT: Linear model. Discussion

![Residuals vs Fitted](image)

![Normal Q–Q](image)

![Scale–Location](image)

![Residuals vs Leverage](image)
SPOT: Linear model. Discussion. Residuals

1. SIGMANULL

2. VARA

3. VARG
SPOT: Linear model. Discussion. Fit
Good news!

- use several models in parallel
- ensemble based prediction
- trend, direction of improvement
- minimize downside risk
- screening
SPOT: Random forest, importance

VARA <> 1.78753714511171
VARA <> 1.52055680404463
0.0069
88 obs
0.2421
12 obs
0.9641
10 obs

sasp.conf

normalized ROI

SIGMANULL
VARA
VARG

Y
0.0 0.3 0.6

normalized ROI
-1.0 0.0 1.0

SPOT: Sensitivity Analysis

- Sensitivity Analysis based on
  - Variance (ANOVA)
  - Regression models
  - DACE’s $\theta$
  - Regression trees
  - etc.

- Combinations are possible ⇒ Meta analysis
SPOT: EDA

- Interaction plots
- Main effect plots
- Regression trees
- Scatter plots

- Box plots
- Trellis plots
- Design plots
- ...

Plotted against TAU1
SPOT: EDA

S effect plot

RESTARTS effect plot

IPSF effect plot

ISD effect plot

LS effect plot

USPROP effect plot
Automated versus Interactive Tuning

- Automated:
  - Time consuming
  - Complex models
  - Limited insight
  - Fair
  - Nonspecialist

- Interactive:
  - Expert knowledge
  - Smart, simple models
  - Insight
  - Generalization
SPOT Open Questions

- Models?
  - (Linear) Regression models
  - Stochastic process models

- Designs?
  - Space filling
  - Factorial

- Statistical tools

- Significance

- Standards

- SPO is a methodology — more than just an optimization algorithm (Synthese)

- Recent trend: SPOT used as an optimizer

- SPOT Community:
  - Provide SPOT interfaces for important optimization algorithms
  - Simple and open specification
  - Currently available for several algorithms, more than a dozen applications

Bartz-Beielstein, Preuss (Cologne, Dortmund)  Automatic and Interactive Tuning  July 2011
SPOT is not alone...

Tuning methods are an active research area:

- Comparison of algorithms without parameter tuning is comparing unsuitable algorithms
- Tuning reveals parameter relevance and interactions

Recent methods:
- F-Race (Birattari, Stützle): Iterative bad parameter elimination
- REVAC (Nannen, Eiben, Smit): Meta-EDA
- ParamILS (Stützle, Hoos, Hutter): Iterative local search
- Probably more to come...
How to Tune On Real-World Problems?

Idea: We build a surrogate model of the problem and use it for algorithm tuning, then apply tuned algorithm to original problem

- Possible solution for expensive problems, but can this work?
- Yes, but we need enough points to capture the local structure of the problem

⇒ First successful study (Preuss/Rudolph/Wessing) GECCO’10
Idea: Modern EA’s as the CMA-ES need many restarts, why not do these with (slightly) different parameters?

- On non-trivial problem instances, this is nearly always an advantage
- Does not help on trivial or very hard instances (floor/ceiling effects)
What do Tuning Methods Deliver?

- A best configuration from \( \{ \text{perf}(\text{alg}(\text{arg}_t^{\text{exo}})) \} | 1 \leq t \leq T \} \) for \( T \) tested configurations
- A spectrum of configurations, each containing a set of single run results
- Detect unsuitable parameter configurations

![Graph showing LHS spectrum for spam filter problem]
A Simple, Visual Approach: Sample Spectra

- spectrum: SPO, hillclimber EA
- spectrum: SPO, swn-topology EA
- spectrum: SPO, niching EA
- spectrum: SPO, generic EA

reached performance (minimization) fraction in %

- 0.00
- 0.05
- 0.10

0
10
20
30

0
10
20
30

0
10
20
30

0
10
20
30
Efficiency vs. Adaptability

Most existing experimental studies focus on the efficiency of optimization algorithms, but:

- Adaptability (how expensive is it to adapt the algorithm) to a problem is usually not measured, although
- It is known as one of the important advantages of EAs (this is all tuning is about!)
- Measure suggested in [Pre09]
- Adaptability is the hardness of the tuning problem

Interesting, previously neglected aspects:

- Interplay between adaptability and efficiency?
- What is the problem spectrum an algorithm performs well on?
- Systematic investigation may reveal inner logic of algorithm parts (operators, parameters, etc.)
One-Parameter Effect Investigation

*Effect Split Plots: Effect Strengths*

- Sample set partitioned into 3 subsets (here of equal size)
- Enables detecting more important parameters visually
- Nonlinear progression 1–2–3 hints to interactions or multimodality
Choose the Appropriate Measure

- Design problem: Only best-of-n fitness values are of interest
- Recurring problem or problem class: Mean values hint to quality on a number of instances
- Cheap (scientific) evaluation functions: exploring limit behavior is tempting, but is not always related to real-world situations

In real-world optimization, $10^4$ evaluations is a lot, sometimes only $10^3$ or less is possible:

- We are relieved from choosing termination criteria
- We encourage more research on short runs (horizontal)
- Or tasks reachable with short runs (vertical)

Selecting a performance measure is a very important step
Convergence of Measuring Perspectives

- Vertical: Probably nearer to real-world situation
- Horizontal: Easier to interpret (BBOB), but targets must be fixed
- However, we have still distributions! Mean or median may be insufficient!
- Carlos Fonseca: Attainment surfaces?

(Thomas Bartz-Beielstein) (Anne Auger/Nikolaus Hansen)
Diagrams Instead of Tables
Reporting and Keeping Track of Experiments

Around 40 years of experimental tradition in EC, but:

- No standard scheme for reporting experiments (experimental protocols)
- Instead: one (“Experiments”) or two (“Experimental Setup” and “Results”) sections in papers, providing a bunch of largely unordered information
- Affects readability and impairs reproducibility

Keeping experimental journals helps:

- Record context and rough idea
- Report each experiment
- Running where (machine)
- Finished when (date/time), link to result file(s)

⇒ We suggest a 7-part reporting scheme (also well suited for tuning experiments)
Suggested Report Structure

ER-1: **Research Question** the matter dealt with

ER-2: **Pre-experimental planning** first—possibly explorative—program runs, leading to task and setup

ER-3: **Task** main question and scientific and derived statistical hypotheses to test

ER-4: **Setup** problem and algorithm designs, sufficient to replicate an experiment

ER-5: **Results/Visualization** raw or produced (filtered) data and basic visualizations

ER-6: **Observations** exceptions from the expected, or unusual patterns noticed, plus additional visualizations, no subjective assessment

ER-7: **Discussion** test results and necessarily subjective interpretations for data and especially observations

This scheme is well suited to report SPO experiments (but not only)
Floor and Ceiling Effects

- Floor effect: Compared algorithms attain set task very rarely
  ⇒ Problem is too hard
- Ceiling effect: Algorithms nearly always reach given task
  ⇒ Problem is too easy

If problem is too hard or too easy, nothing is shown. Tuning processes will fail because of insufficient feedback!

- Pre-experimentation is necessary to obtain reasonable tasks
- If task is reasonable (e.g. practical requirements), then algorithms are unsuitable (floor) or all good enough (ceiling), statistical testing does not provide more information
- Arguing on minimal differences is statistically unsupported and scientifically meaningless
Confounded Effects

Two or more effects or helper algorithms are merged into a new technique, which is improved

- Where does the improvement come from?
- It is necessary to test both single effects/algorithms, too
- Either the combination helps, or only one of them
- Knowing that is useful for other researchers!

complex machinery
Underestimated Randomness

- Idea: Find Pareto front of two tuning criteria
- Parameter changes not interpretable
- Validation failed
- Reason: Deviations much too high!

More difficulties: See also papers of the GECCO’09 workshop
Learning from Failures in Evolutionary Computation (LFFEC)
There Is a Problem With the Experiment

After all data is in, we realize that something was wrong (code, parameters, environment?), what to do?

- Current approach: Either do not mention it, or redo everything
- If redoing is easy, nothing is lost
- If it is not, we must either:
  - Let people know about it, explaining why it probably does not change results
  - Or do validation on a smaller subset: How large is the difference (e.g. statistically significant)?
- Do not worry, this situation is rather normal
- Thomke: There is nearly always a problem with an experiment
- Early experimentation reduces the danger of something going completely wrong
• SPO is not the final solution—it is one possible (but not necessarily the best) solution
• Goal: continue a discussion in EC, transfer results from statistics and the philosophy of science to computer science
• Standards for good experimental research
• Interesting directions: SAMP and MAMP, much work to do
• From the problem perspective: ELA, building on benchmarks as BBOB
• Please check
  http://www.gm.fh-koeln.de/campus/personen/lehrende/thomas.bartz-beielstein/00489/
  for updates, software, etc.
Suggested Readings

Reference Book on Experimental Research

- Experimental Methods for the Analysis of Optimization Algorithms
- See also Kleijnen [Kle08], Saltelli et al.


Thomas Bartz-Beielstein, Boris Naujoks, Tobias Wagner, and Simon Wessing.

Thomas Bartz-Beielstein, Konstantinos E. Parsopoulos, and Michael N. Vrahatis.
Design and analysis of optimization algorithms using computational statistics.

H.-G. Beyer.
*The Theory of Evolution Strategies*.

Oliver Flasch, Thomas Bartz-Beielstein, Artur Davtyan, Patrick Koch, Wolfgang Konen, Tosin Daniel Oyetoyan, and Michael Tamutan.
Comparing ci methods for prediction models in environmental engineering.
Thomas Fober, Marco Mernberger, Gerhard Klebe, and Eyke Hüllermeier.

Evolutionary construction of multiple graph alignments for the structural analysis of biomolecules.  

Thomas Fober.

Experimentelle Analyse Evolutionärer Algorithmen auf dem CEC 2005 Testfunktionensatz.  

Frank Hutter, Thomas Bartz-Beielstein, Holger Hoos, Kevin Leyton-Brown, and Kevin P. Murphy.

Sequential model-based parameter optimisation: an experimental investigation of automated and interactive approaches empirical methods for the analysis of optimization algorithms.  
In Thomas Bartz-Beielstein, Marco Chiarandini, Luis Paquete, and Mike Preuss, editors, *Empirical Methods for the Analysis of Optimization*


J. P. C. Kleijnen. *Design and analysis of simulation experiments*. 


Jörn Mehnens, Thomas Michelitsch, Christian Lasarczyk, and Thomas Bartz-Beielstein. Multi-objective evolutionary design of mold temperature control using DACE for parameter optimization.
Appendix references


Boris Naujoks, Domenico Quagliarella, and Thomas Bartz-Beielstein. Sequential parameter optimisation of evolutionary algorithms for airfoil design.


Mike Preuss. Adaptability of algorithms for real-valued optimization.

Mike Preuss, Günter Rudolph, and Feelly Tumakaka.
Solving multimodal problems via multiobjective techniques with Application to phase equilibrium detection.

Günter Rudolph, Mike Preuss, and Jan Quadflieg. Two-layered surrogate modeling for tuning optimization metaheuristics. Algorithm Engineering Report TR09-2-005, Faculty of Computer Science, Algorithm Engineering (Ls11), Technische Universität Dortmund, Germany, September 2009.


Heike Trautmann and Jörn Mehnen. Statistical methods for improving multi-objective evolutionary optimisation.
Suggested Readings


**Marko Tosic.**
Evolutionäre Kreuzungsminimierung.

**L.G. Volkert.**
Investigating ea based training of hmm using a sequential parameter optimization approach.

**Yu Yi.**
*Fuzzy Operator Trees for Modeling Utility Functions.*