Tuning and Experimental Analysis in EC: What We Still Have Wrong

Thomas Bartz-Beielstein[1] Mike Preuss[2]

[1]Faculty of Computer Science and Engineering Science
Cologne University of Applied Sciences

[2]Department of Computer Science
TU Dortmund

July 2010

ACM 978-1-4503-0073-5/10/07.

Agenda

Instructor Biographies

- 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
- Mike Preuss is Research Associate at the Computer Science Department, University of Dortmund, where he also received his Diploma degree in 1998. His research interests focus on the field of EAs for real-valued problems. He is currently working on the adaptability and applicability of CI techniques for various engineering domains and computer games
- 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

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) *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 so solve problems, even if theory is not developed far enough
- How shall we 'sell' our algorithms?
- 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
- In Metaheuristics (us), this adaptation is always guided by experiment

As currently performed, experimentation often gets us

a) Some funny figures

b) Lots of better and better algorithms which soon disappear again

This procedure appears to be

a) Arbitrary (parameter, problem, performance criterion choice?)

b) Useless, as nothing is explained and generalizability is unclear

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?)

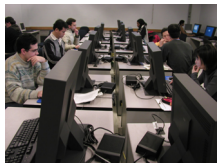
This is an experiment

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?



Is this an experiment?

Lessons From Other Sciences

In economics, experimentation was established quite recently (compared to its age)

- Modeling human behavior as the rationality assumption (of former theories) had failed
- No accepted new model available: Experimentation came in as substitute

Nonlinear behavior

In (evolutionary) biology, experimentation and theory building both have problems

- Active experimentation only possible in special cases, otherwise only observation
- Mainly concepts (rough working principles) instead of theories: there are always exceptions

$\Rightarrow$ Stochastical distributions, population thinking

Ernst Mayr

Experimentation at Unexpected Places

Since about the 1960s: Experimental Archaeology

- Gather (e.g. performance) data that is not available otherwise
- Task: Concept validation, fill conceptual holes

Viking bread baking (Lejre, Danmark)

Experimentation in management of technology and product innovation

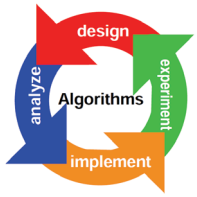
- Product cycles are sped up by 'fail-fast', 'fail-often' experimentation
- What-if questions may be asked by using improved computational ressources
- Innovation processes have to be tailored towards experimentation

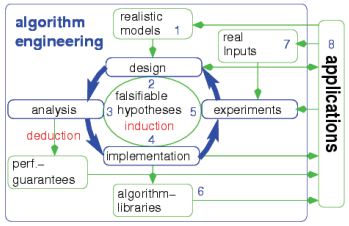
Stefan H. Thomke

Algorithm Engineering

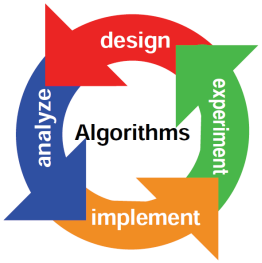
How Theoreticians Handle it...(Recently)

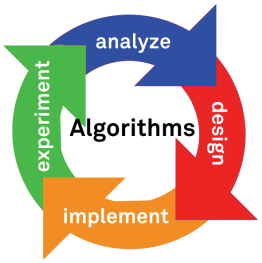
- Algorithm Engineering is *theory* + real data + concrete implementations + experiments
- Principal reason for experiments: Test validity of theoretical claims
- Are there important factors in practice that did not go into theory?
- Approach also makes sense for metaheuristics, but we start with no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





Or Algorithm Reengineering?





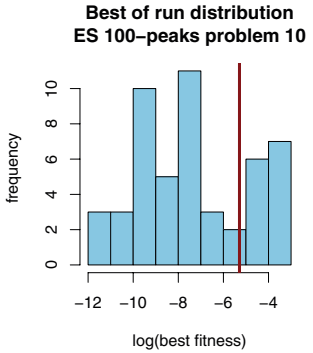
For the analysis of metaheuristics, algorithm reengineering may be more appropriate

- We start from an existing algorithm and redesign (simplify) it
- We stop if we can match existing theoretical (analysis) methods
- We check performance against original method via experiment

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...

Goals in Evolutionary Computation

(RG-1) *Investigation.* Specifying optimization problems, analyzing algorithms. What could be a reasonable research question? What is going to be explained? Does it help in practice? Enables theoretical advances?

(RG-2) *Comparison.* Comparing the performance of heuristics
Any reasonable approach here has to regard fairness

(RG-3) *Conjecture.* Good: demonstrate performance. Better: explain and understand performance
Needed: Looking at the behavior of the algorithms, not only results

(RG-4) *Quality.* Robustness (includes insensitivity to exogenous factors, minimization of the variability) [Mon01]
Invariance properties (e.g. CMA-ES): Find out, for what (problem, parameter, measure) spaces our results hold

A Totally Subjective History of Experimentation in Evolutionary Computation

- Palaeolithic: Mean values
- Yesterday: Mean values and simple statistics
- Today: Correct statistics, statistically meaningful conclusions
- Tomorrow: Scientific meaningful conclusions

Some Myth

- GAs are better than other algorithms (on average)
- Comparisons based on the mean
- One-algorithm, one-problem paper
- Everything is normal
- 10 (100) is a nice number
- One-max, Sphere, Ackley
- Performing good experiments is a lot easier than developing good theories

Today: Based on Correct Statistics

Example (Good practice?)

- Authors used
  - Pre-defined number of evaluations set to 200,000
  - 50 runs for each algorithm
  - Population sizes 20 and 200
  - Crossover rate 0.1 in algorithm A, but 1.0 in B
  - A outperforms B significantly in f_6 to f_{10}
- We need tools to
  - Determine adequate number of function evaluations to avoid floor or ceiling effects
  - Determine the correct number of repeats
  - Determine suitable parameter settings for comparison
  - Determine suitable parameter settings to get working algorithms
  - Draw meaningful conclusions
- Problems of today:
  Adequate statistical methods, but wrong scientific conclusions

High-Quality Statistics

- Fundamental to all comparisons - even to high-level procedures
- The basic procedure reads:

  Select test problem (instance) P
  Run algorithm A, say n times
  Obtain n fitness values: $x_{A,i}$
  Run algorithm B, say n times
  Obtain n fitness values: $x_{B,i}$

How Do We Set Up An Experiment?

- Set up experiments to show improved algorithm performance
- But why are we interested showing improved algorithm performance?
- Because the algorithm
  - does not find any feasible solution (effectiveness)
    or
  - has to be competitive to the best known algorithm (efficiency)
- How do we measure the importance or significance of our results?
- We need meta-measures:
  - First, we measure the performance
  - Next, we measure the importance of differences in performance
- Many statistics available, none of them is used by now
- Each measure will produce its own ranking
- Planning of experiments

$\Rightarrow$ Fix research question, fix experimental setup (in this order)

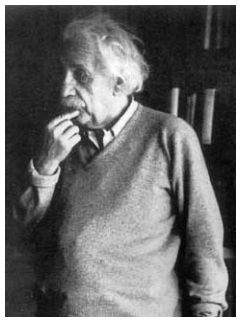
Research Question

- 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...
Horse racing: set up, run, comment...NO!

Explaining observations leads to new questions:

- Multi-step process appropriate
- Conjectures obtained from results shall itself be tested experimentally
- Range of validity shall be explored (problems, parameters, etc.)



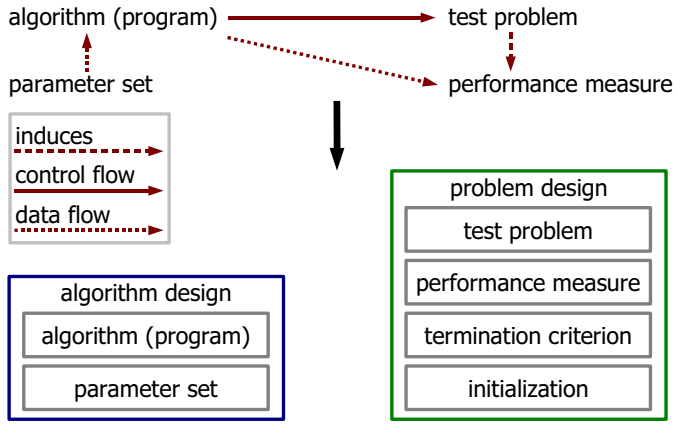
Einstein thinking

□

Tomorrow: Correct Statistics and Correct Conclusions

- Consider scientific meaning
- Severe testing as a basic concept (First Symposium on Philosophy, History, and Methodology of Error, June 2006)
- To discover the scientific meaning of a result, it is necessary to pose the right question in the beginning
- In the beginning: before we perform experiments
- Significance of an effect: Effect occurs even for small sample sizes, i.e., $n = 10$
- Clarify the model:
  - Diagnostic: understanding the algorithm
  - Prognostic: predicting the algorithm's performance
  - Data-driven: treat results from an experiment as a signal which indicates (statistical) properties
  - Theory-driven: verify certain assumptions, e.g., step-size adaptation rules
- Other categorizations possible
- Categories can be used as guidelines to avoid chaotic arrangements of assumptions and propositions

Components of an Experiment in Metaheuristics



First step: Archeology—Detect Factors

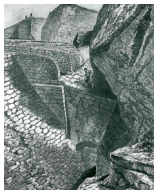


Figure: Schliemann in Troja

- "Playing trumpet to tulips" or "experimenter's socks"
- In contrast to field studies: Computer scientists have all the information at hand
- Generating more data is relatively fast
- First classification:
  - algorithm
  - problem

⇒ We have (beside others) a parameter problem, many EAs highly depend on choosing them 'right'

Classification

- Algorithm design
  - Population size
  - Selection strength
- Problem design
  - Search space dimension
  - Starting point
  - Objective function
- Vary problem design ⟹ effectivity (robustness)
- Vary algorithm design ⟹ efficiency (tuning)

Efficiency

- Tuning
- Problems
  - Many factors
  - Real–world problem: complex objective function (simulation) and only small number of function evaluations
  - Theoretical investigations: simple objective function and many function evaluations
- Screening to detect most influential factors

Factor Effects

- Important question: Does a factor influence the algorithm's performance?
- How to measure effects?
- First model:

$$Y = f(\vec{X}),$$

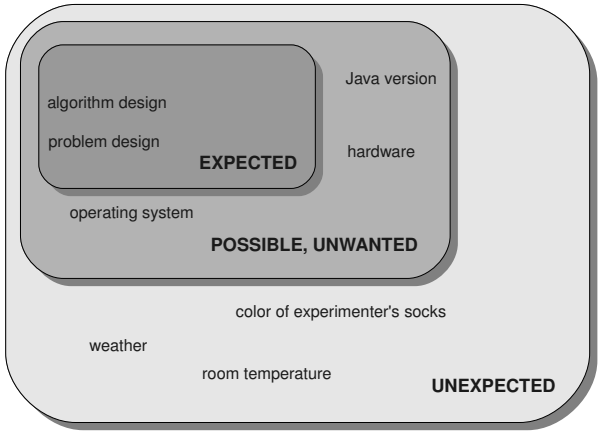
where
  - $\vec{X} = (X_1, X_2, \ldots, X_r)$ denote r factors from the algorithm design and
  - Y denotes some output (i.e., best function value from 1000 generations)
- Problem design remains unchanged
- Uncertainty analysis: compute average output, standard deviation, outliers ⇒ related to Y
- Sensitivity analysis: which of the factors are more important in influencing the variance in the model output Y? ⇒ related to the relationship between X_i, X_j and Y

Measures for Factor Effects

- How many factors are important?
- Practitioners observed: input factor importance distributed as the wealth in nations — a few factors produce nearly all the variance

- Overview
  - Variance
  - Derivation
  - DoE: Regression coefficients (β)
  - DACE: Coefficients (θ)

Factors: Overview



SPO Definition

Definition

SPO is a framework for tuning and understanding of algorithms by active experimentation. SPO employs methods from error statistics to obtain reliable results. It comprises the following elements:

SPOT-1 Scientific questions

SPOT-2 Statistical hypotheses

SPOT-3 Experiments

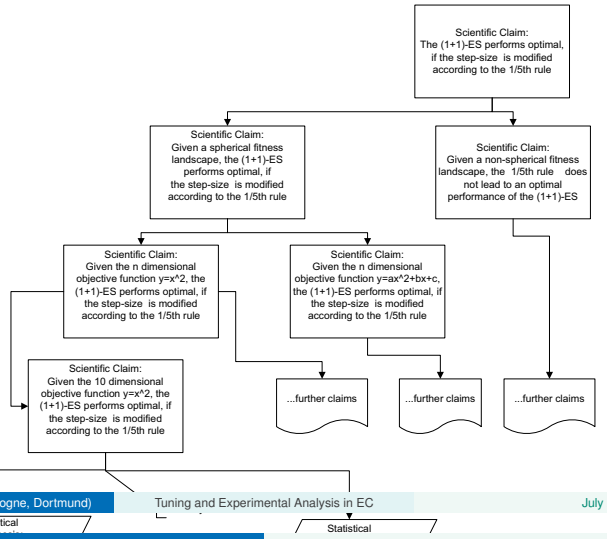
SPOT-4 Scientific meaning

□

SPO Overview

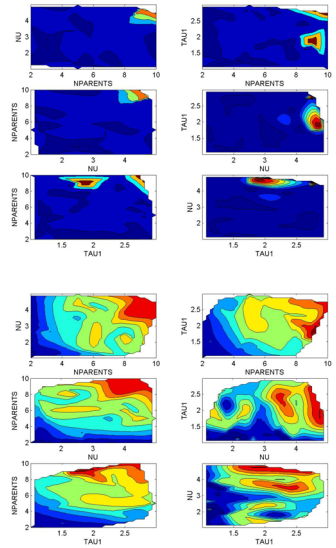
- Starting point: Scientific question (SPO-1)
  - Deals with assumptions about algorithms, e.g., influence of parameter values or new operators
- This (complex) question is broken down into (simple) statistical hypotheses (SPO-2) for testing, see [BB08] for an example
- Next, experiments can be performed for each hypothesis:
  - Select a model (e.g., regression trees) to describe a functional relationship
  - Select an experimental design
  - Generate data, i.e., perform experiments
  - Refine the model until the hypothesis can be accepted/rejected
- SPOT: Software programs for performing experiments (SPO-3)
- Finally, to assess the scientific meaning of the results from an experiment, conclusions are drawn from the hypotheses
- This is step (SPO-4) in the sequential parameter optimization framework

SPO-1 complex question broken down into simple statistical hypotheses SPO-2



Scientific and Statistical Hypotheses

- Scientific claim: "ES with small populations perform better than ES with larger ones on the sphere."
- Statistical hypotheses:
  - ES with, say $\mu = 2$, performs better than ES with $mu > 2$ if compared on problem design $\overset{(1)}{p}$
  - ES with, say $\mu = 2$, performs better than ES with $mu > 2$ if compared on problem design $\overset{(2)}{p}$
  - ...
  - ES with, say $\mu = 2$, performs better than ES with $mu > 2$ if compared on problem design $\overset{(n)}{p}$



SPO Workflow

1. *Pre-experimental* planning
2. *Scientific* thesis
3. *Statistical* hypothesis
4. Experimental *design*: Problem, constraints, start-/termination criteria, performance measure, algorithm parameters

---

5. *Experiments*
6. Statistical *models* and predictions (lm, dace, tgp, tree,...). Evaluation and visualization
7. Solution good enough?
   Yes: Goto step 8
   No: Improve the design (optimization). Goto step 5

---

8. *Acceptance/rejection* of the statistical hypothesis
9. Objective *interpretation* of the results from the previous step

SPOT

- *Sequential parameter optimization toolbox* (SPOT): one possible implementation of the SPO framework
- 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

SPOT: Definition

Definition (Sequential Parameter Optimization Toolbox)

SPOT implements features related to step (SPO-3) from the SPO framework

SPOT-1 Use the available budget (e.g., simulator runs, number of function evaluations) *sequentially*:
- Use information from the exploration of the search space to guide the search by building one or several meta models
- Choose new design points based on predictions from the meta model(s)
- Refine the meta model(s) stepwise to improve knowledge about the search space

SPOT-2 If necessary, try to cope with *noise* by improving confidence. Guarantee comparable confidence for search points

SPOT-3 Collect information to *learn* from this tuning process, e.g., apply explorative data analysis

SPOT-4 Provide mechanisms both for *interactive* and *automated tuning*

□

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 (see also Fig. 2):
  - *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.

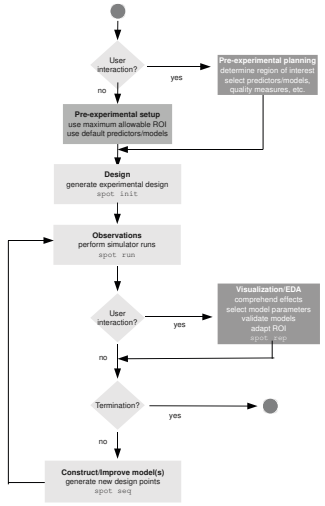


Figure: Spot process

SPO Region of Interest (ROI)

- *Region of interest* (ROI) files specify the region, over which the algorithm parameters are tuned

```
name low high isint pretty
NPARENTS 1 10 TRUE 'NPARENTS'
NU 1 5 FALSE 'NU'
TAU1 1 3 FALSE 'TAU1'
```

Figure: demo4.roi

SPO Configuration File

- *Configuration* files (CONF) specify SPO specific parameters, such as the regression model

```
new=0
defaulttheta=1
loval=1E-3
upval=100
spotrmodel='regpoly2'
spotcmodel='corrgauss'
isotropic=0
repeats=3
...
```

Figure: demo4.m

SPO Output File

- *Design* files (DES) specify algorithm designs
- Generated by SPO
- Read by optimization algorithms

```
TAU1 NPARENTS NU TAU0 REPEATS CONFIG SEED STEP
0.210507 4.19275 1.65448 1.81056 3 1 0 1
0.416435 7.61259 2.91134 1.60112 3 2 0 1
0.130897 9.01273 3.62871 2.69631 3 3 0 1
1.65084 2.99562 3.52128 1.67204 3 4 0 1
0.621441 5.18102 2.69873 1.01597 3 5 0 1
1.42469 4.83822 1.72017 2.17814 3 6 0 1
1.87235 6.78741 1.17863 1.90036 3 7 0 1
0.372586 3.08746 3.12703 1.76648 3 8 0 1
2.8292 5.85851 2.29289 2.28194 3 9 0 1
...
```

Figure: demo4.des

Algorithm: Result File

- Algorithm run with settings from design file
- Algorithm writes *result file* (RES)
- RES files provide basis for many statistical evaluations/visualizations
- RES files read by SPO to generate stochastic process models

```
Y NPARENTS FNAME ITER NU TAU0 TAU1 KAPPA NSIGMA RHO DIM CONFIG SEED
3809.15 1 Sphere  500 1.19954 0 1.29436 Inf 1 2 2  1 1
0.00121541  1 Sphere  500 1.19954 0 1.29436 Inf 1 2 2 1 2
842.939 1 Sphere 500 1.19954 0 1.29436 Inf 1 2 2  1 3
2.0174e-005 4 Sphere 500 4.98664 0 1.75367 Inf 1 2 2  2 1
0.000234033 4 Sphere 500 4.98664 0 1.75367 Inf 1 2 2  2 2
1.20205e-007  4 Sphere 500 4.98664 0 1.75367 Inf 1 2 2  2 3
...
```

Figure: demo4.res

SPO Toolbox Implementations

- SPOT Availability
  - Matlab
  - R: package
- R: several design generators and predictor as plugins
- Use and combine(!) prediction models

Case study: Simple Evolution Strategy

- Theoretical problem: 1/5th rule
- Old problem, several reference solutions
- How to chose an adequate method?

Case study: Simple Evolution Strategy

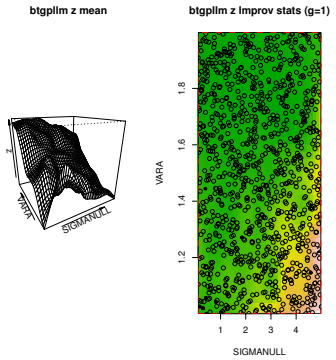
Table: $(1+1)$-ES parameters. The first three parameters belong to the algorithm design, whereas the remaining parameters are from the problem design.

Name	Symbol	Factor name
Initial stepsize	$\sigma(0)$	SIGMANULL
Stepsize multiplier	a	VARA
History	$g = n$	VARG
Starting point	$\vec{x_p}$	
Problem dimension	n	
Objective function	$f(\vec{x}) = \sum x_i^2$	
Quality measure	Expected performance, e.g., $E(y)$	
Initial seed	s	
Budget	$t_{\max}$	

Factors, Designs and Predictors

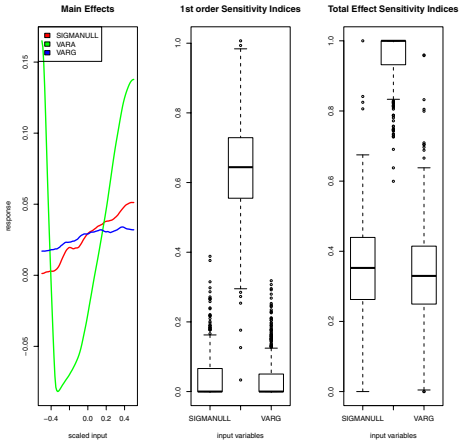
- Factors
  - Numerical
  - Categorical (ordered and unordered)
- Designs
  - Classical fractional factorial designs
  - Space-filling designs, e.g., Latin hypercube designs
- Predictors
  - Linear regression
  - Regression trees
  - Tree based Gaussian process

Predictors



- Predictors for very expensive optimization runs (real-world problems)
  - Tree based Gaussian processes
  - DACE
- Predictors for simple optimization runs (theoretical investigations)
  - Classical regression models
  - Regression trees
- Combinations are possible ⇒ Meta predictors

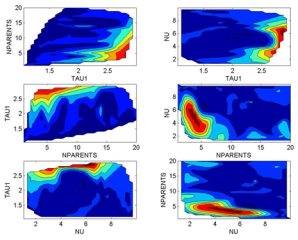
Sensitivity Analysis



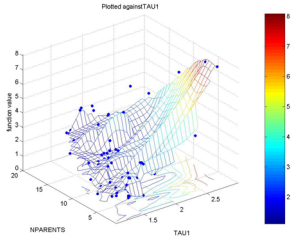
- Sensititvity Analysis based on
  - Variance (ANOVA)
  - Regression models
  - DACE's θ
  - Regression trees
  - etc.
- Combinations are possible ⇒ Meta analysis

SPO and EDA

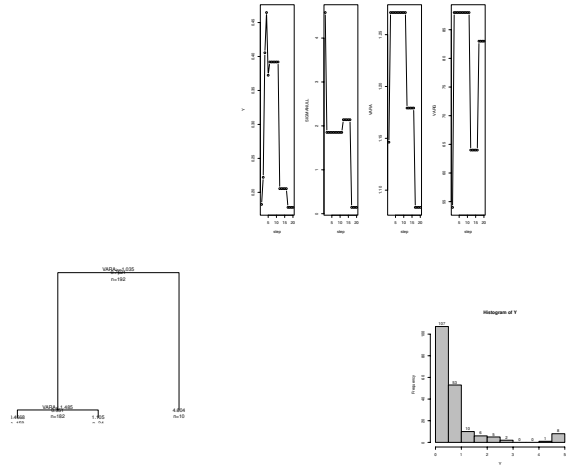
- Interaction plots
- Main effect plots
- Regression trees
- Scatter plots



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



SPOT Live Demo



SPO 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
- Probably more to come...

SPO Open Questions

- Models?
  - (Linear) Regression models
  - Stochastic process models
- Designs?
  - Space filling
  - Factorial
- Statistical tools
- Significance
- Standards

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

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

□

Empirical Analysis: Algorithms for Scheduling Problems

- Problem:
  - Jobs build binary tree
  - Parallel computer with ring topology
- 2 algorithms:

  Keep One, Send One (KOSO) to my right neighbor
  Balanced strategy KOSO*: Send to neighbor with lower load only
- Is KOSO* better than KOSO?

Empirical Analysis: Algorithms for Scheduling Problems

- Hypothesis: Algorithms influence running time
- But: Analysis reveals

  # Processors und # Jobs explain 74 % of the variance of the running time
  Algorithms explain nearly nothing
- Why?

  Load balancing has no effect, as long as no processor starves.
  But: Experimental setup produces many situations in which processors do not starve
- Furthermore: Comparison based on the optimal running time (not the average) makes differences between KOSO und KOSO*.
- Summary: Problem definitions and performance measures (specified as algorithm and problem design) have significant impact on the result of experimental studies

Floor and Ceiling Effects

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

If problem is too hard or too easy, nothing is shown

- 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

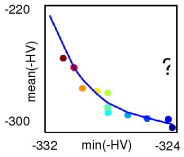
Two or more effects or helper algorithms are merged into a new technique, which is improved

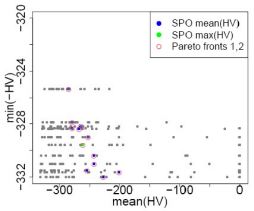
- Where does the improvement come from?
- It is necessary to test both single effects/algorithms, too
- Either the combination helps, or only one of them
- Knowing that is useful for other researchers!

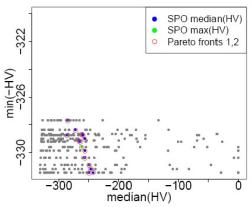
complex machinery

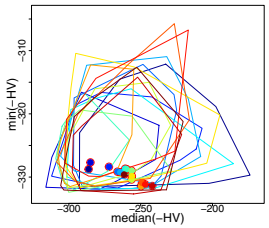
Underestimated Randomness

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









More difficulties: See also papers in 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?

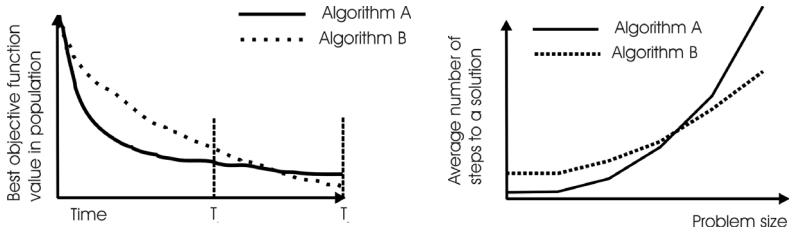
- Current approach: Either do not mention it, or redo everything
- If redoing is easy, nothing is lost
- If it is not, we must either:
  - Let people know about it, explaining why it probably does not change results
  - Or do validation on a smaller subset: How large is the difference (e.g. statistically 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

"Traditional" Measuring in EC

Simple Measures

- MBF: mean best fitness
- AES: average evaluations to solution
- SR: success rates, SR(t) ⇒ run-length distributions (RLD)
- best-of-n: best fitness of *n* runs

But, even with all measures given: Which algorithm is better?



(figures provided by Gusz Eiben)

Aggregated Measures

Especially Useful for Restart Strategies

Success Performances:

- SP1 [HK04] for equal expected lengths of successful and unsuccessful runs $\mathbb{E}(T^s) = \mathbb{E}(T^{us})$:

$$SP1 = \frac{\mathbb{E}(T_A^s)}{p_s} \tag{1}$$

- SP2 [AH05] ($= ERT(f_{target})$ from BBOB workshop GECCO'09) for different expected lengths, unsuccessful runs are stopped at FE_{max}:

$$SP2 = \frac{1-p_s}{p_s} FE_{max} + \mathbb{E}(T_A^s) \tag{2}$$

$$ERT(f_{target}) = \frac{\#FEs(f_{best} \geq f_{target})}{\#succ} \tag{3}$$

Probably still more aggregated measures needed (parameter tuning depends on the applied measure)

Choose the Appropriate Measure

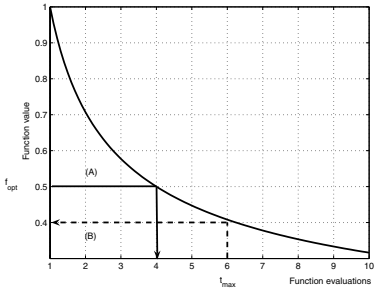
- Design problem: Only best-of-n fitness values are of interest
- Recurring problem or problem class: Mean values hint to quality on a number of instances
- Cheap (scientific) evaluation functions: exploring limit behavior is tempting, but is not always related to real-world situations

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

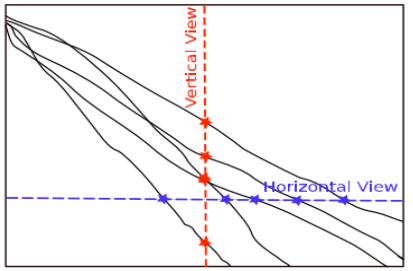
- We are relieved from choosing termination criteria
- Substitute models may help (Algorithm based validation)
- We encourage more research on short runs (horizontal)
- Or tasks reachable with short runs (vertical)

Selecting a performance measure is a *very* important step

Convergence of Measuring Perspectives



(Thomas Bartz-Beielstein)

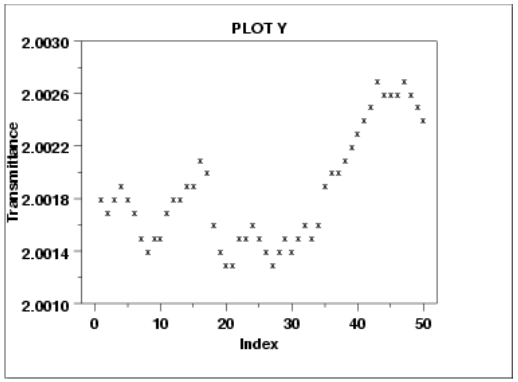


(Anne Auger/Nikolaus Hansen)

- Vertical: Probably nearer to real-world situation
- Horizontal: Easier to interpret (BBOB'09), but task(s) must be fixed
- However, we have still distributions! Mean or median may be insufficient!
- Carlos Fonseca: Attainment surfaces?

Diagrams Instead of Tables

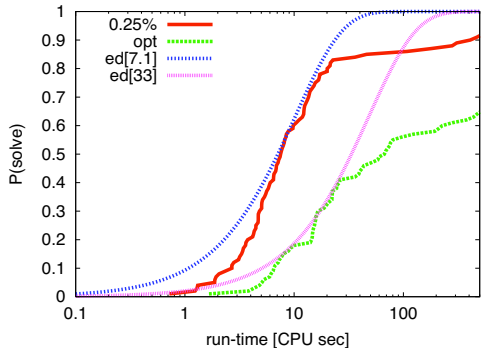
Would You Have Seen This From a Table?



Sequence plot

Visual Comparison With a Task Set

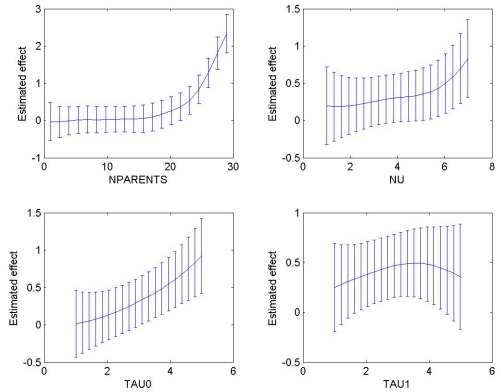
Run-length distributions



(courtesy of Thomas Stuetzle)

(Single) Effect Plots

Useful, but not Perfect

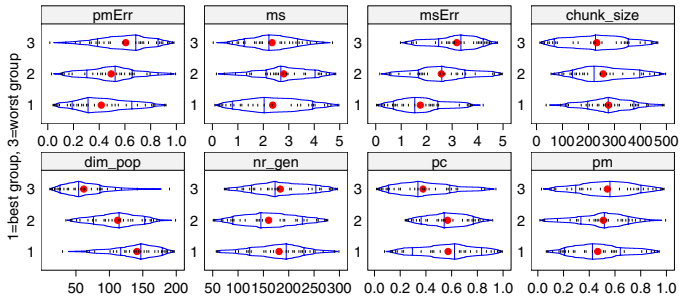


- Large variances originate from averaging
- The τ_0 and especially τ_1 plots show different behavior on extreme values (see error bars), probably distinct (averaged) effects/interactions

One-Parameter Effect Investigation

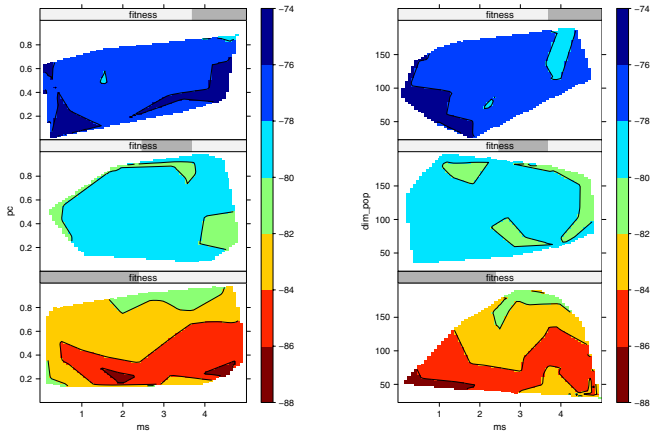
Effect Split Plots: Effect Strengths

- 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



Two-Parameter Effect Investigation

Interaction Split Plots: Detect Leveled Effects



□

Current "State of the Art"

Around 40 years of empirical tradition in EC, but:

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

Other sciences have more structured ways to report experiments, although usually not presented in full in papers. Why?

- Natural sciences: Long tradition, setup often relatively fast, experiment itself takes time
- Computer science: Short tradition, setup (implementation) takes time, experiment itself relatively fast

⇒ We suggest a 7-part reporting scheme

Suggested Report Structure

ER-1: **Focus/Title** 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)

The Art of Comparison

Orientation

The NFL[1] told us things we already suspected:

- We cannot hope for the one-beats-all algorithm (solving the general nonlinear programming problem)
- Efficiency of an algorithm heavily depends on the problem(s) to solve and the exogenous conditions (termination etc.)

In consequence, this means:

- The posed question is of extreme importance for the relevance of obtained results
- The focus of comparisons has to change from:

  Which algorithm is better?

  to questions like

  What exactly is the algorithm good for?
  How can we generalize the behavior of an algorithm?
  ⇒ *Rules of thumb, finally theory*

[1] no free lunch theorem

The Art of Comparison

Efficiency vs. Adaptability

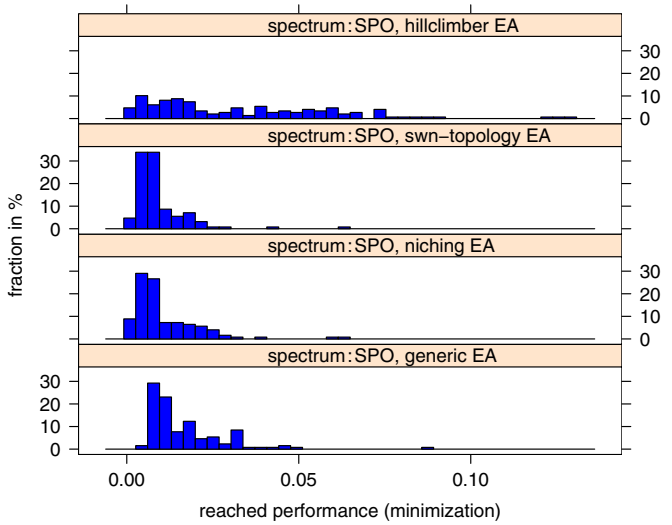
Most existing experimental studies focus on the efficiency of optimization algorithms, but:

- Adaptability to a problem is not measured, although
- It is known as one of the important advantages of EAs

Interesting, previously neglected aspects:

- Interplay between adaptability and efficiency?
- How much effort does adaptation to a problem take for different algorithms?
- What is the problem spectrum an algorithm performs well on?
- Systematic investigation may reveal inner logic of algorithm parts (operators, parameters, etc.)

A Simple, Visual Approach: Sample Spectra



What is the Meaning of Parameters?

Are Parameters "Bad"?

Cons:

- Multitude of parameters dismays potential users
- It is often not trivial to understand parameter-problem or parameter-parameter interactions

  $\Rightarrow$ Parameters complicate evaluating algorithm performances

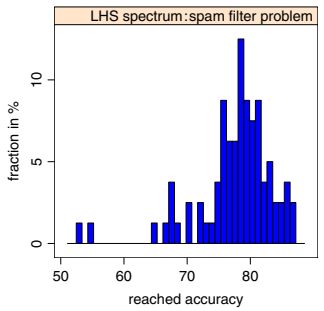
But:

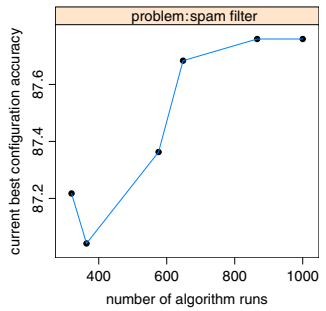
- Parameters are simple handles to modify (adapt) algorithms
- Many of the most successful EAs have lots of parameters
- New theoretical approaches: Parametrized algorithms / parametrized complexity, ("two-dimensional" complexity theory)

Tuning and Comparison

What do Tuning Methods (e.g. SPO) Deliver?

- A best configuration from $\{perf(alg(arg_t^{exo}))|1 \leq t \leq T\}$ for T tested configurations
- A spectrum of configurations, each containing a set of single run results
- A progression of current best tuning results





How do Tuning Results Help?

...or Hint to New Questions

What we get:

- A near optimal configuration, permitting top performance comparison
- An estimation of how good any (manually) found configuration is
- A (rough) idea how hard it is to get even better

No excuse: A first impression may be attained by simply doing an LHS

Yet unsolved problems:

- How much amount to put into tuning (fixed budget, until stagnation)?
- Where shall we be on the spectrum when we compare?
- Can we compare spectra ($\Rightarrow$ adaptability)?

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'
- 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

What If Available Comparison Data Is Unsufficient?

Many empirical papers provide not enough data to test against

- Testing against mean values is statistically not meaningful
- But giving lots of data is not always possible (page limit)
- Many online sources (e.g. ACM JEA) allow for storing data

We shall think of ways to make data available online

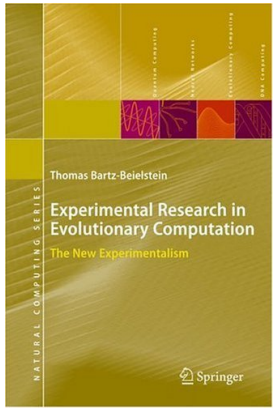
- Establish our own repositories? On journal pages?
- Or put data on our web pages? Formats?

It is very important to strengthen the aspect of *replication*!

Discussion

- 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
- Review process
- Research grants
- Meetings
- Building a community
- Teaching
- ...

Reference



- Please check
  `http://www.gm.fh-koeln.de/campus/personen/lehrende/thomas.bartz-beielstein/00489/`
  for updates, software, etc.

New Book on Experimental Research

- To appear 2010: Experimental Methods for the Analysis of Optimization Algorithms
- See also Kleijnen [Kle08], Saltelli et al.

`http://www.springer.com/computer/ai/book/978-3-642-02537-2`

Anne Auger and Nikolaus Hansen.
Performance Evaluation of an Advanced Local Search Evolutionary Algorithm.
In B. McKay et al., editors, *Proc. 2005 Congress on Evolutionary Computation (CEC'05)*, Piscataway NJ, 2005. IEEE Press.

Thomas Bartz-Beielstein.
Experimental Research in Evolutionary Computation—The New Experimentalism.
Natural Computing Series. Springer, Berlin, Heidelberg, New York, 2006.

Thomas Bartz-Beielstein.
How experimental algorithmics can benefit from Mayo's extensions to Neyman-Pearson theory of testing.
Synthese, 163(3):385–396, 2008.
DOI 10.1007/s11229-007-9297-z.

Thomas Bartz-Beielstein.
Sequential parameter optimization—an annotated bibliography.
Technical Report 04/2010, Institute of Computer Science, Faculty of Computer Science and Engineering Science, Cologne University of Applied Sciences, Germany, 2010.

Thomas Bartz-Beielstein, Marco Chiarandini, Luis Paquete, and Mike Preuss, editors.
Empirical Methods for the Analysis of Optimization Algorithms.
Springer, Berlin, Heidelberg, New York, 2010.

Thomas Bartz-Beielstein, Christian Lasarczyk, and Mike Preuß.
Sequential parameter optimization.
In B. McKay et al., editors, *Proceedings 2005 Congress on Evolutionary Computation (CEC'05), Edinburgh, Scotland*, volume 1, pages 773–780, Piscataway NJ, 2005. IEEE Press.

Thomas Bartz-Beielstein, Boris Naujoks, Tobias Wagner, and Simon Wessing.
In Hermann Locarek-Junge and Claus Weihs, editors, *Proceedings 11th IFCS Internat. Conference 2009. Classification as a tool for research*, Dresden, 2009.

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

Applied Numerical Analysis and Computational Mathematics (ANACM), 1(2):413–433, 2004.

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.
Technical Report 02/2010, Institute of Computer Science, Faculty of Computer Science and Engineering Science, Cologne University of Applied Sciences, Germany, 2010.

Thomas Fober, Marco Mernberger, Gerhard Klebe, and Eyke Hüllermeier.

Evolutionary construction of multiple graph alignments for the structural analysis of biomolecules.
Bioinformatics, 25(16):2110–2117, 2009.

Thomas Fober.
Experimentelle Analyse Evolutionärer Algorithmen auf dem CEC 2005 Testfunktionensatz.
Master's thesis, Universität Dortmund, Germany, 2006.

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 Algorithms*, pages 361–414. Springer, Berlin, Heidelberg, New York, 2009.

Frank Henrich, Claude Bouvy, Christoph Kausch, Klaus Lucas, Mike, Preuß, Günter Rudolph, and Peter Roosen.
Economic optimization of non-sharp separation sequences by means of evolutionary algorithms.
Computers and Chemical Engineering, 32(7):1411–1432, 2008.

F. Hutter, H. H. Hoos, K. Leyton-Brown, and K. P. Murphy.
Time-bounded sequential parameter optimization.
In *Proc. of LION-10*, 2010.
To appear.

Nikolaus Hansen and Stefan Kern.

Evaluating the cma evolution strategy on multimodal test functions.
In X. Yao, H.-P. Schwefel, et al., editors, *Parallel Problem Solving from Nature – PPSN VIII, Proc. Eighth Int'l Conf., Birmingham*, pages 282–291, Berlin, 2004. Springer.

Oliver Kramer, Bartek Gloger, and Andreas Goebels.
An experimental analysis of evolution strategies and particle swarm optimisers using design of experiments.
In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, GECCO '07, pages 674–681, New York, NY, USA, 2007. ACM.

J. P. C. Kleijnen.
Design and analysis of simulation experiments.
Springer, New York NY, 2008.

W. Konen, T. Zimmer, and T. Bartz-Beielstein.
Optimierte Modellierung von Füllständen in Regenüberlaufbecken mittels CI-basierter Parameterselektion Optimized Modelling of Fill Levels in Stormwater Tanks Using CI-based Parameter Selection Schemes.
at-Automatisierungstechnik, 57(3):155–166, 2009.

Christian W. G. Lasarczyk.

Genetische Programmierung einer algorithmischen Chemie.
PhD thesis, Technische Universität Dortmund, 2007.

Christian W. G. Lasarczyk and Wolfgang Banzhaf.
Total synthesis of algorithmic chemistries.
In *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 1635–1640, New York, NY, USA, 2005. ACM.

Jörn Mehnen, Thomas Michelitsch, Christian Lasarczyk, and Thomas Bartz-Beielstein.
Multi-objective evolutionary design of mold temperature control using DACE for parameter optimization.
International Journal of Applied Electromagnetics and Mechanics, 25(1–4):661–667, 2007.

D. C. Montgomery.
Design and Analysis of Experiments.
Wiley, New York NY, 5th edition, 2001.

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

In G. et al. Winter, editor, *Proc. Design and Optimization: Methods and Applications, (ERCOFTAC'06)*, pages 231–235. University of Las Palmas de Gran Canaria, 2006.

Niels Hendrik Pothmann.
Kreuzungsminimierung für k-seitige Buchzeichnungen von Graphen mit Ameisenalgorithmen.
Master's thesis, Universität Dortmund, Germany, 2007.

Mike Preuss, Günter Rudolph, and Feelly Tumakaka.
Solving multimodal problems via multiobjective techniques with Application to phase equilibrium detection.
In *Proceedings of the International Congress on Evolutionary Computation (CEC2007). Piscataway (NJ): IEEE Press. Im Druck*, 2007.

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.

Selmar K. Smit and A. E. Eiben.
Comparing Parameter Tuning Methods for Evolutionary Algorithms.

In *IEEE Congress on Evolutionary Computation (CEC)*, pages 399–406, May 2009.

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

International Journal of Computational Intelligence Research (IJCIR), 5(2):72–78, 2009.

Marko Tosic.
Evolutionäre Kreuzungsminimierung.
Diploma thesis, University of Dortmund, Germany, January 2006.

L.G. Volkert.
Investigating ea based training of hmm using a sequential parameter optimization approach.
In Gary G. Yen, Simon M. Lucas, Gary Fogel, Graham Kendall, Ralf Salomon, Byoung-Tak Zhang, Carlos A. Coello Coello, and Thomas Philip Runarsson, editors, *Proceedings of the 2006 IEEE Congress on Evolutionary Computation*, pages 2742–2749, Vancouver, BC, Canada, 16-21 July 2006. IEEE Press.

Yu Yi.

Fuzzy Operator Trees for Modeling Utility Functions.
PhD thesis, Philipps-Universität Marburg, 2008.