Automatic and Interactive Tuning of Algorithms

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 2011

GECCO'11, July 12-16, 2011, Dublin, Ireland.

ACM 978-1-4503-0690-4/11/07.

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, 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).
- 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) *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 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$\star$)
- 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 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$\star$)
- Helps theoreticians to find exploitable (parameter/problem) relations

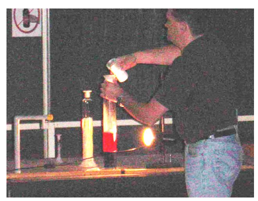
Instead, we converge to

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?)



This is an experiment

In computer science, the situation seems different

- 2 widespread stereotypes influence our view of computer experiments:

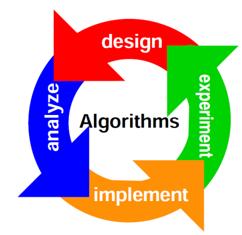
a) Programs do (exactly) what algorithms specify
b) Computers (programs) are deterministic, so why statistics?

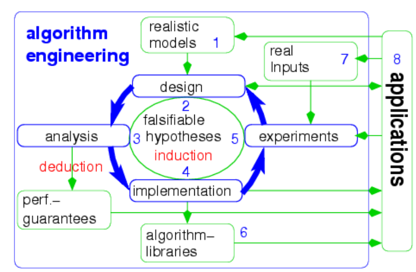
Is this an experiment?

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

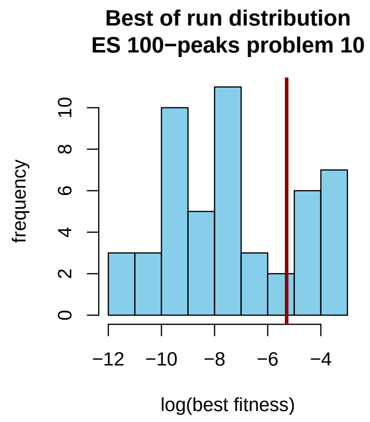




So What About Statistics?

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
- Consequently, we include (generic!) adaptability of algorithms to problems in the algorithm design
- Knowledge about parameter interactions helps us to understand algorithms and problems
- This helps us to approach 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

⇒ Experimental methodology and reliable tuning methods needed

□

Tuning: Goals

(TG-1) *Performance.* Important parameters; what should be optimized?
(TG-2) *Comparison.* Comparing the performance of heuristics
(TG-3) *Conjecture.* Good: demonstrate performance. Better: explain and understand performance
Needed: Looking at the behavior of the algorithms, not only results
(TG-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

First step: Archeology—Detect Factors

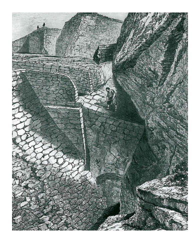
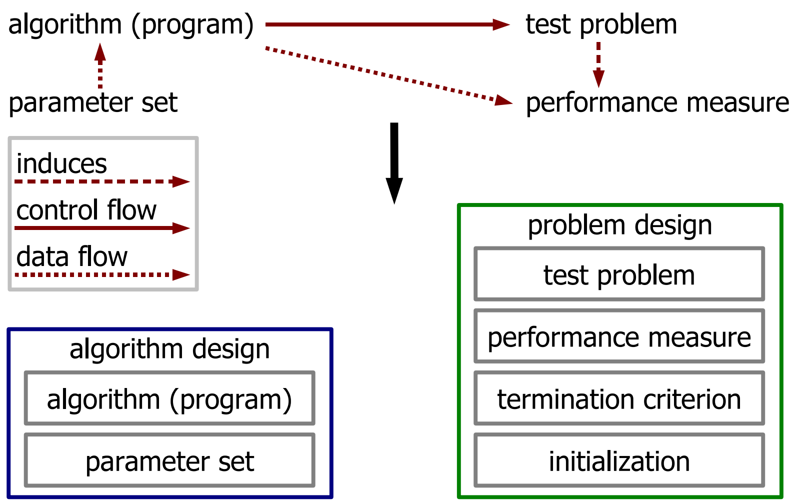


Figure: Schliemann in Troja

- “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’

Components of an Experiment in Metaheuristics



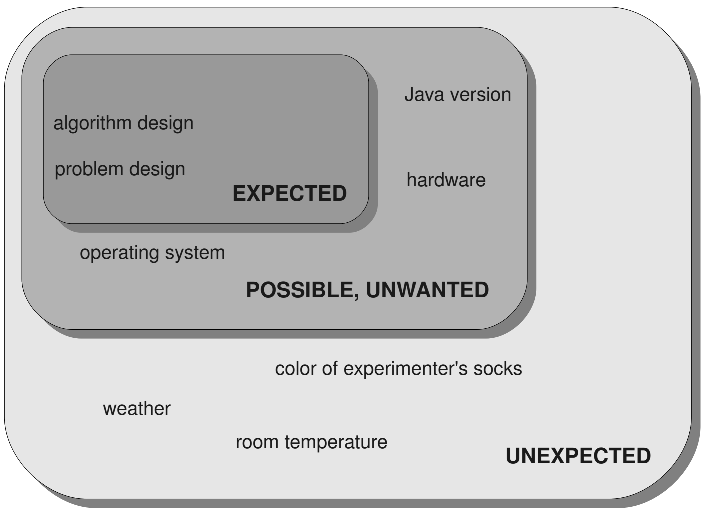
Classification

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

Factor Effects

- Important question: Does a factor influence the algorithm's performance?
- How to measure effects?
- First model:
$$Y = f(\vec{X}),$$
  where
  - $\vec{X} = (X_a, X_p)$, where X_a and X_p denote factors from the algorithm and problem design, respectively and
  - Y denotes some output (i.e., best function value from 1000 generations)
- Uncertainty analysis: compute average output, standard deviation, outliers $\Rightarrow$ related to Y
- Sensitivity analysis: which of the factors are more important in influencing the variance in the model output Y? $\Rightarrow$ related to the relationship between X_a, X_p and Y

Factors: Overview



Problems and Algorithms

- Tuning can be performed for
  - (SASP) One single algorithm and one single problem instance
  - (SAMP) One single Algorithm and multiple problems instances
  - (MASP) Multiple algorithms and one single problem instance
  - (MAMS) Multiple algorithms and multiple problem instances
- How to perform comparisons?
- Adequate statistics?

SASP – Single Algorithm, Single Problem

- Typical real-world setting
- Determine important factors
- Optimization
- Crucial: number of function evaluations
- Benefit:
  - $ $ $

SAMP – Single Algorithm, Multiple Problems

- Algorithm development
- Determine important factors
- Optimization
- Robustness
- Benefit:
  - Determine important factors
  - Understanding

MASP – Multiple Algorithms, Single Problem

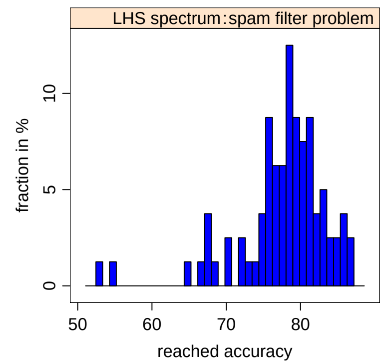
- Research (beginner's paper $\Rightarrow$ rejected)
- Optimization
- Note: multiple algorithms $\sim$ one algorithm with different parameters
- Tuning and comparison
- Benefit:
  - Similarities between algorithms
  - Understanding

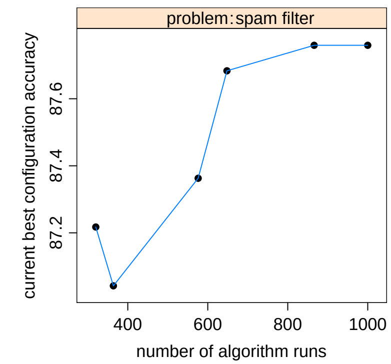
MAMP – Multiple Algorithms, Multiple Problems

- Research (expert paper $\Rightarrow$ accepted)
- Comparison
- Huge complexity
- Benefit:
  - Accepted paper

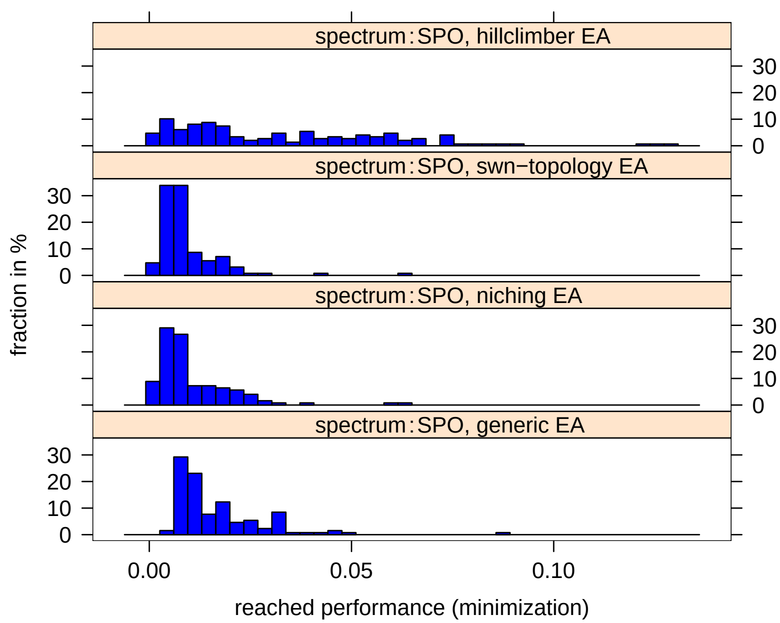
What do Tuning Methods Deliver?

- 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
- Detect unsuitable parameter configurations

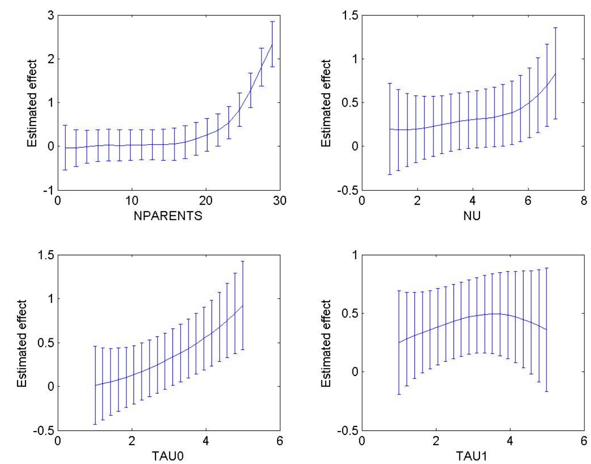




A Simple, Visual Approach: Sample Spectra



(Single) Effect Plots

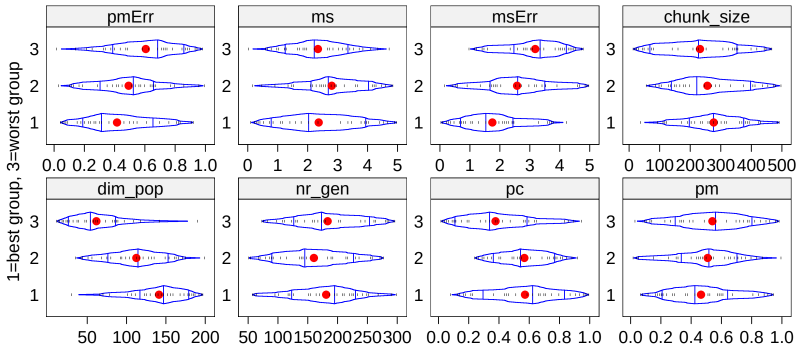


- Large variances originate from averaging
- The τ_0 and especially τ_1 plots show different behavior on extreme values (see error bars), probably distinct (averaged) effects/interactions

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



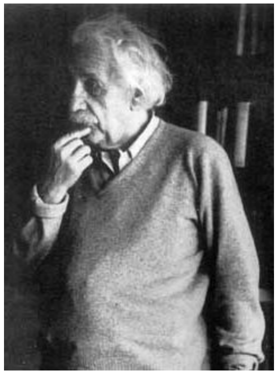
Research Question

- 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.)



Einstein thinking

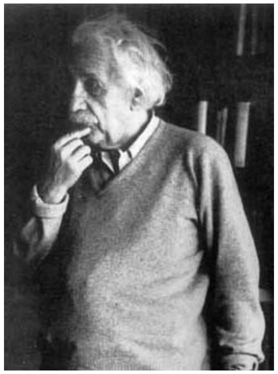
Research Question

- 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.)



Einstein thinking

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

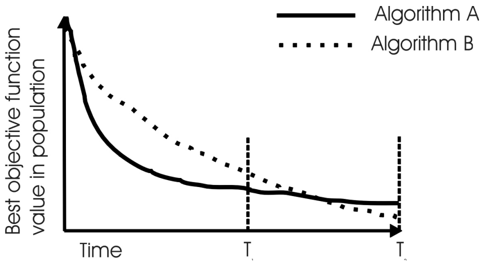
- An inherent problem with experimentation is that we do (should) not know the outcome in advance
- But it may lead to new, better questions
- Try small steps, expect the unexpected

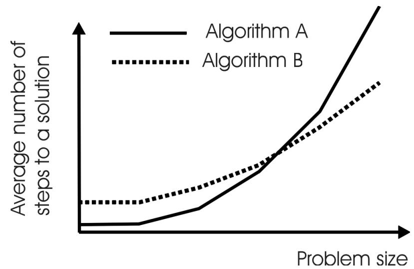
“Traditional” Measuring in EC

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 $\mathbb{E}(T^s) = \mathbb{E}(T^{us})$:

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

- ERT(f_{target}) as used in the BBOB setup for different expected lengths, unsuccessful runs are stopped at FE_{max}:

$$ERT(f_{target}) = \frac{\#FEs(f_{best} \geq f_{target})}{\#succ} \tag{2}$$

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

Choose the Appropriate Measure

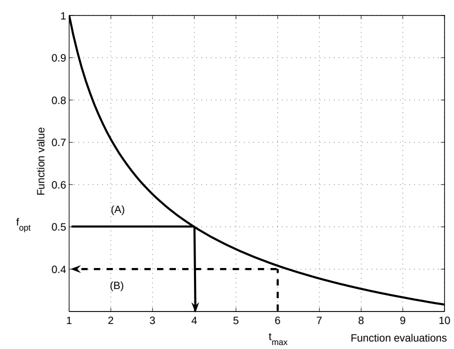
- Design problem: Only best-of-n fitness values are of interest
- Recurring problem or problem class: Mean values hint to quality on a number of instances
- Cheap (scientific) evaluation functions: exploring limit behavior is tempting, but is not always related to real-world situations

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

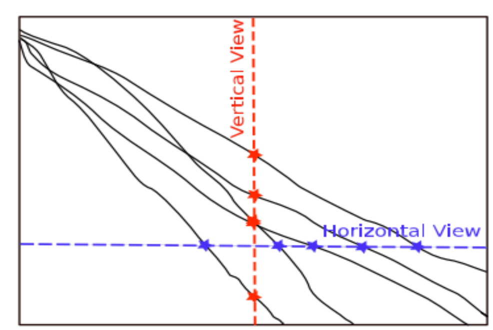
- We are relieved from choosing termination criteria
- Substitute models may help (Algorithm based validation)
- We encourage more research on short runs (horizontal)
- Or tasks reachable with short runs (vertical)

Selecting a performance measure is a *very* important step

Convergence of Measuring Perspectives



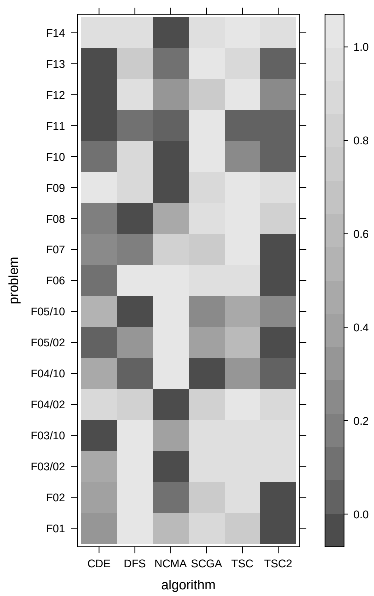
(Thomas Bartz-Beielstein)



(Anne Auger/Nikolaus Hansen)

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

Diagrams Instead of Tables



Method	Peak ratio		Basin ratio		Peak accuracy		Distance accuracy	
	Best	Avg.	Best	Avg.	Best	Avg.	Best	Avg.
***F*1, 1 global optimum, 9 local ones**								
TSC2	**0.99**	**0.84**	1	0.88	**0.13**	1.84	**0.04**	0.79
CDE	0.88	0.79	0.98	**0.93**	0.52	**1.59**	0.11	**0.41**
TSC [14]	0.85	0.64	0.83	0.66	4.52	7.7	1.29	3.26
NCMA-ES	0.8	0.49	0.9	0.59	1.85	8.89	0.88	3.87
SCGA	0.66	0.18	0.99	0.264	8.74	18.59	0.98	11.56
DFS	0.37	0.16	0.37	0.16	14.46	20.93	5.24	11.52
***F*2, 2 global, 4 local optima**								
TSC2	1	**0.77**	1	**0.77**	**6.93e-04**	**2.91**	0.02	2.09
NCMA-ES	1	0.59	1	0.61	1.72e-03	3.9	0.02	3.19
CDE	1	0.75	1	0.76	0.02	3.3	0.1	**1.99**
SCGA	0.96	0.32	1	0.35	0.39	6.37	0.44	7.02
DFS	0.67	0.26	0.67	0.26	4.64	7.27	2.73	6.22
TSC [14]	0.63	0.46	0.66	0.44	3.93	6.18	3.44	6.18
***F*3, 2 dimensions, 1 optimum**								
NCMA-ES	1	1	1	1	**4.6e-68**	3.92e-6	**6.48e-35**	5.84e-4
CDE	1	1	1	1	9.47e-40	4.48e-04	1.96e-20	5.25e-03
TSC2	1	1	1	1	5.85e-12	1.81e-07	1.61e-06	9.32e-05
SCGA	1	1	1	1	1.53e-11	2.86e-07	2.41e-06	1.65e-04
TSC [14]	1	1	1	1	2.48e-10	**1.75e-07**	4.9e-06	**9.08e-05**
DFS	1	1	1	1	2.55e-09	4.17e-06	4.23e-05	8.12e-04
***F*3, 10 dimensions, 1 optimum**								
CDE	1	**0.83**	1	1	**2.66e-25**	0.11	**4.07e-13**	**0.15**
NCMA-ES	1	0.73	1	1	1.28e-17	**0.08**	2.51e-09	0.19
TSC2	1	0.73	1	1	2.36e-06	0.15	0.001	0.23
TSC [14]	1	0.74	1	1	2.79e-06	0.12	0.003	0.51
SCGA	1	0.72	1	1	1.03e-05	1.43	0.003	0.45
DFS	1	0.72	1	1	3.12e-05	0.14	0.005	0.22
***F*4, 2 dimensions, 1 global optimum/ many local ones**								
NCMA-ES	1	0.86	1	0.88	**0**	0.19	**9.05e-9**	0.14
DFS	1	0.98	1	0.98	9.13e-08	0.02	7.24e-06	0.02
SCGA	1	**0.99**	1	**0.99**	1.4e-07	**0.01**	1.46e-05	**0.01**
CDE	1	0.88	1	0.98	4.29e-07	0.11	3.93e-05	0.03
TSC2	1	0.8	1	0.94	2.23e-06	1.63	8.23e-05	0.05
TSC [14]	1	0.74	1	0.93	5.04e-05	1.73	5.1e-04	0.07
***F*4, 10 dimensions, 1 global optimum/ many local ones**								
SCGA	1	**0.35**	1	**0.66**	0.002	18.42	0.003	1.71
TSC2	1	0.04	1	0.27	0.002	39.78	0.003	2.57
DFS	1	0.31	1	0.44	0.003	**8.93**	0.003	**1.44**
TSC [14]	0.97	0.03	1	0.28	0.03	51.46	0.03	6.08
CDE	0.9	0.12	0.97	0.19	0.09	18.68	0.04	1.68
NCMA-ES	0	0	0	0	26.9	23.6	2.46	3.32
***F*5, 2 dimensions, 1 global optimum/ many local ones**								
TSC2	**0.77**	**0.26**	**0.97**	**0.67**	14.74	369.93	**9.4e-04**	0.49
DFS	0.7	0.21	0.73	0.24	58.09	164.85	1.34	3.05
TSC [14]	0.63	0.19	0.73	0.29	273.64	934.45	0.96	1.07
SCGA	0.47	0.21	0.6	0.31	81.47	317.24	0.11	2.51
CDE	0	0.003	1	0.96	**20.65**	**134.64**	0.01	**0.07**
NCMA-ES	0	0	0	0	1700	1840	1.62	0.71
***F*5, 10 dimensions, 1 global optimum/ many local ones**								
NCMA-ES	0	0	0	0.01	1810	1900	**9.44**	**8.82**
TSC2	0	0	0	0	770.48	1234.14	9.64	11.24
DFS	0	0	0	0	**569.6**	**870.64**	11.08	12.85
CDE	0	0	0	0	1076.2	1301.02	11.99	9.32
SCGA	0	0	0	**0.3**	762.8	1311.85	12.95	11.49
TSC [14]	0	0	0	0	961.59	1151.36	33.33	32.37

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: **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)

□

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

SPOT: Definition

Definition (Sequential Parameter Optimization Toolbox)

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

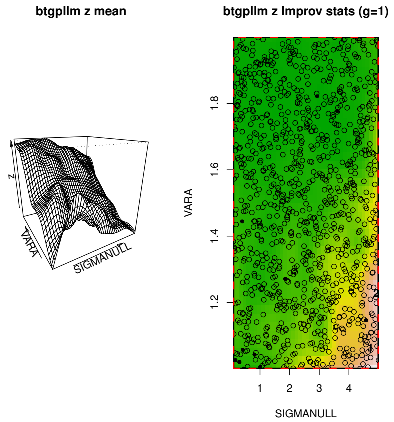
SPOT Tasks

- SPOT provides tools to perform the following tasks:
  - *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

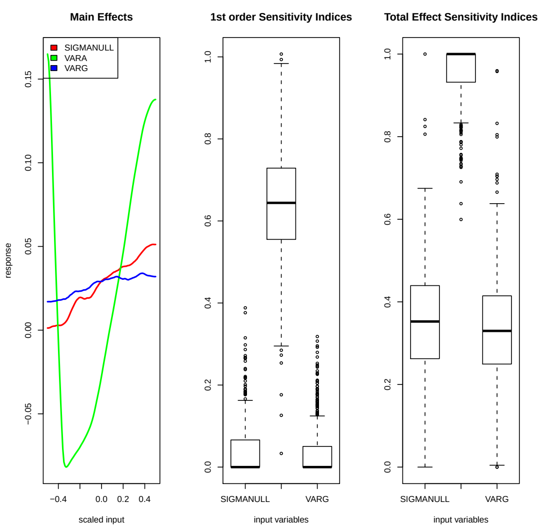


- Predictors for very expensive optimization runs (real-world problems)
  - Tree based Gaussian processes
  - DACE
- Predictors for simple optimization runs (theoretical investigations)
  - Classical regression models
  - Regression trees
- Combinations are possible ⇒ Ensemble-based modeling, Meta predictors

SPOT Applications

- SASP demo
- SAMP demo
- MASP demo
- MAMP demo

SPOT: Sensitivity Analysis

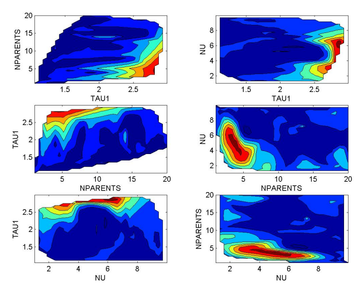


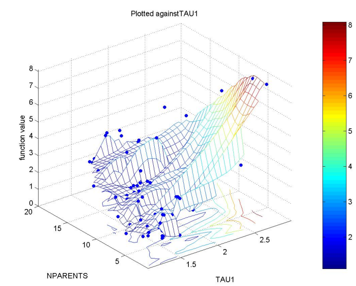
- Sensititvity Analysis based on
  - Variance (ANOVA)
  - Regression models
  - DACE's θ
  - Regression trees
  - etc.
- Combinations are possible ⇒ Meta analysis

SPOT: EDA

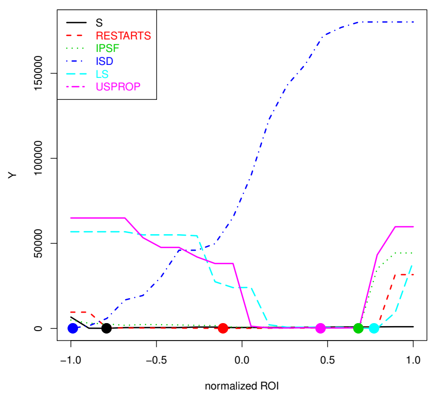
- Interaction plots
- Main effect plots
- Regression trees
- Scatter plots

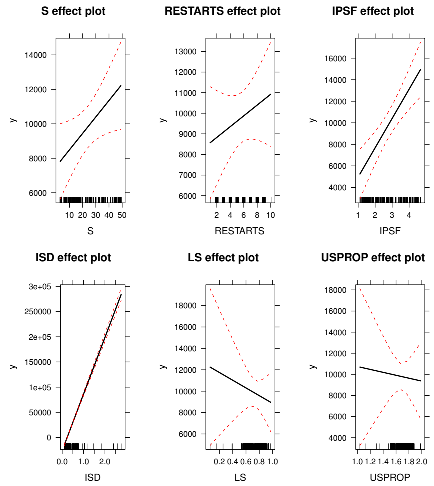
- Box plots
- Trellis plots
- Design plots
- ...





SPOT: EDA





Automated versus Interactive Tuning

- Interactive
  - Expert knowledge
  - Simple models and designs, e.g., classical fractional factorial designs and response surface
  - Insight
  - Understanding
- Automated
  - Time consuming
  - Complex models, e.g., (tree based) Gaussian process
  - Limited insight

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
- Probably more to come…

SPOT 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)
- Recent trend: SPOT used as an optimizer

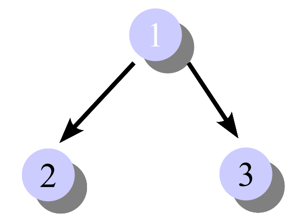
□

A Famous Example: The Rosenberg Study

- 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?

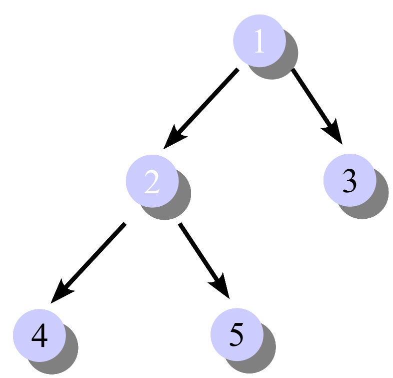
A Famous Example: The Rosenberg Study

- 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?



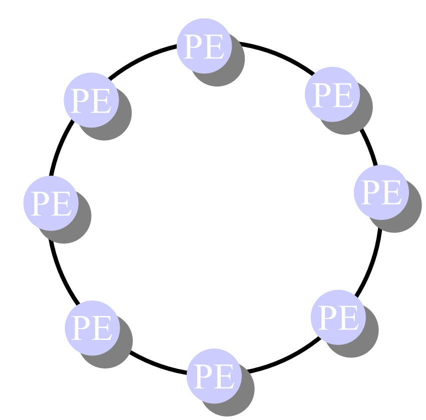
A Famous Example: The Rosenberg Study

- 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?



A Famous Example: The Rosenberg Study

- 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?



A Famous Example: The Rosenberg Study

- 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?

Experimental Analysis: What is the Problem?

- 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

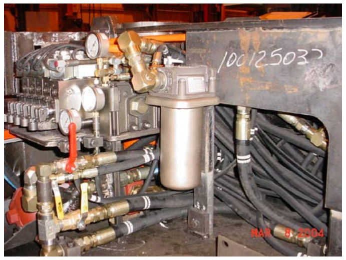
If problem is too hard or too easy, nothing is shown.
Tuning processes will fail because of insufficient feedback!

- 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

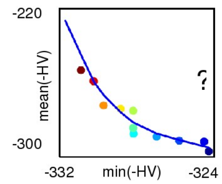
Two or more effects or helper algorithms are merged into a new technique, which is improved

- 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!

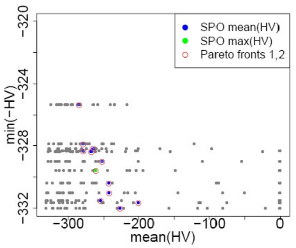


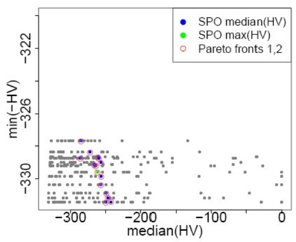
complex machinery

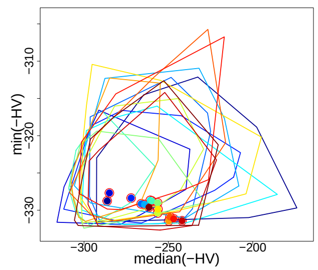
Underestimated Randomness



- 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

Efficiency vs. Adaptability

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

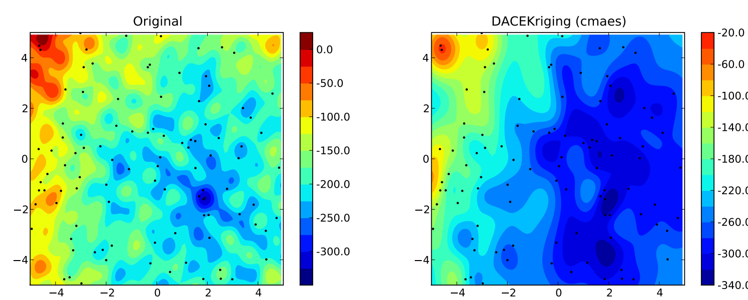
- Adaptability (how expensive is it to adapt the algorithm) to a problem is usually not measured, although
- It is known as one of the important advantages of EAs (this is all tuning is about!)
- Measure suggested in [Pre09]
- Adaptability is the hardness of the tuning problem

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.)

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



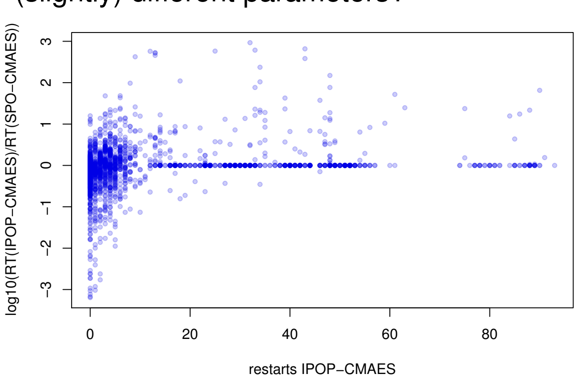
- Possible solution for expensive problems, but can this work?
- Yes, but we need enough points to capture the local structure of the problem

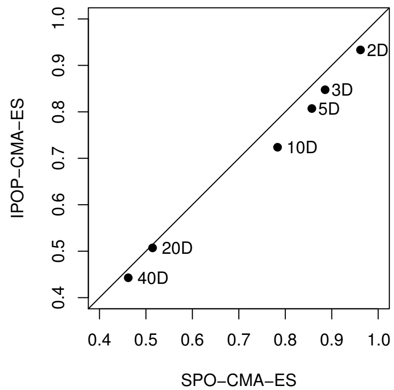
1375 ⇒ First successful study (Preuss/Rudolph/Wessing) GECCO'10

Realtime Tuning

(Wessing/Preuss/Rudolph), GECCO'11

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)

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

- 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.
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.
Adaptability of algorithms for real-valued optimization.
In Mario Giacobini et al., editors, *Applications of Evolutionary Computing, EvoWorkshops 2009. Proceedings*, volume 5484 of *Lecture Notes in Computer Science*, pages 665–674, Berlin, 2009. Springer.

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.