The Future of Experimental Research

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

Sunday, September 14th 2008

PPSN´08, September 13-17, 2008, Dortmund, Germany.

Overview

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

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

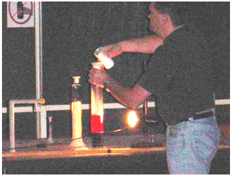
This procedure appears to be

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

⇒ 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

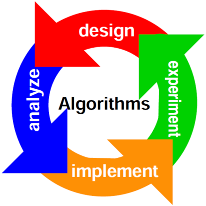
- Product cycles are sped up by ‘fail-fast’, ‘fail-often’ experimentation
- What-if questions may be asked by using improved computational ressources
- Innovation processes have to be tailored towards experimentation

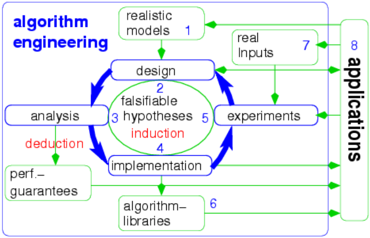
Stefan H. Thomke

Algorithm Engineering

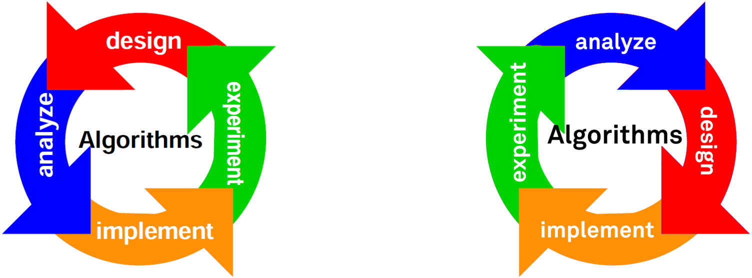
How Theoreticians Handle it...(Recently)

- 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?



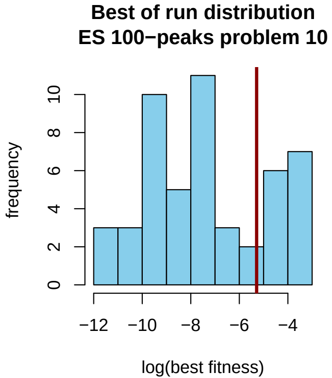
For the analysis of metaheuristics, algorithm reengineering may be more appropriate

- 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.

Cathy McGeogh:
Our problems are unfortunately not sexy enough for the Statisticians...

Advertisement

technische universität dortmund

WEA 2008 SEA 2009 SEA 2010

8th International Symposium on Experimental Algorithms
http://www.sea2009.org

June 3-6, 2009
Faculty of Computer Science, Technische Universität Dortmund, Germany

Call for Papers

SEA, previously known as WEA (Workshop on Experimental Algorithms), is an international forum for researchers in the area of experimental evaluation and engineering of algorithms, as well as in various aspects of computational optimization and its applications.

Previous meetings were held in Riga (Latvia, 2001), Ascona (Switzerland, 2003), Angra dos Reis (Brazil, 2004), Santorini (Greece, 2005), Menorca Island (Spain, 2006), Rome (Italy, 2007), and Cape Cod (USA, 2008).

PROGRAM COMMITTEE

Mark de Berg (TU Eindhoven)
Gerth S. Brodal (MADALGO, Arhus)
Sándor P. Fekete (TU Braunschweig)
Carlos M. Fonseca (U. Algarve)
Giuseppe F. Italiano (U. Roma "Tor Vergata")
Alex López-Ortiz (U. Waterloo)
Petra Mutzel (TU Dortmund)
Panos M. Pardalos (U. Florida)
Mike Preuß (TU Dortmund)
Rajeev Raman (U. Leichester)
Mauricio G. C. Resende (AT&T Labs)
Peter Sanders (U. Karlsruhe)
Matt Stallman (NCSU)
Laura Toma (Bowdoin College)
Jan Vahrenhold (TU Dortmund, chair)
Xin Yao (U. Birmingham)

PLENARY SPEAKERS

(To be confirmed.)

STEERING COMMITTEE

Edoardo Amaldi (Politecnico di Milano)
David A. Bader (Georgia Inst. of Technology)
Josep Diaz (T.U. of Catalonia)
Giuseppe F. Italiano (U. Roma "Tor Vergata")
David Johnson (AT&T Labs)
Klaus Jansen (U. Kiel)
Kurt Mehlhorn (MPII Saarbrücken)
Ian Munro (U. Waterloo)
Sotiris Nikoletseas (U. Patras / CTI)
José Rolim (chair) (U. Geneva)
Paul Spirakis (U. Patras / CTI)

ORGANIZING COMMITTEE

Gundel Jankord (TU Dortmund)
Norbert Jesse (TU Dortmund)
Mike Preuß (TU Dortmund)
Jan Vahrenhold (TU Dortmund)

SCOPE

The main theme of the symposium is the role of experimentation and of algorithm engineering techniques in the design and evaluation of algorithms and data structures. Submissions should present significant contributions supported by experimental evaluation, methodological issues in the design and interpretation of experiments, the use of (meta-)heuristics, or application-driven case studies that deepen the understanding of a problem's complexity.

Contributions solicited cover a variety of topics including but not limited to:
- Algorithm Engineering
- Analysis of Algorithms
- Approximation Techniques
- Bioinformatics
- Combinatorial Structures and Graphs
- Communication Networks
- Computational Geometry
- Computational Learning Theory
- Computational Optimization
- Cryptography and Security
- Data Structures
- Distributed and Parallel Algorithms
- Experimental Techniques and Statistics
- Graph Drawing
- Information Retrieval
- Logistics and Operations Management
- Machine Learning and Data Mining
- Mathematical Programming
- Metaheuristic Methodologies
- Multiple Criteria Decision Making
- Network Analysis
- On-line Problems
- Randomized Techniques
- Robotics
- Software Repositories and Platforms

IMPORTANT DATES

Submission deadline:	January 19, 2009 (11:59 pm PST)
Author notification:	March 6, 2009
Camera ready due:	March 20, 2009
Symposium:	June 3-6, 2009

SUBMISSIONS

Authors are invited to submit high-quality manuscripts reporting original unpublished research and recent developments in the topics related to the symposium. Simultaneous submission to other conferences or workshops with published proceedings is not allowed.
All papers will be peer reviewed and comments will be provided to the authors.
The submission system can be accessed via http://www.sea2009.org.

PROCEEDINGS

Accepted papers will appear in the SEA 2009 proceedings published by Springer in the LNCS series.

SPECIAL ISSUE

Selected papers from SEA 2009 will be considered for a special issue of the ACM Journal of Experimental Algorithmics (JEA, http://www.jea.acm.org).

CONTACT

Jan Vahrenhold, Faculty of Computer Science, Technische Universität Dortmund, 44221 Dortmund, Germany. sea2009@cs.tu-dortmund.de

- The well established WEA (workshop on experimental algorithms) goes SEA (symposium)
- Originally, an algorithm engineering conference, but also open for *experimentally sound* Metaheuristic and OR based papers
- SEA 2009 will be in Dortmund!
- PC includes Xin Yao, Carlos Fonseca, Mauricio Resende, and Mike Preuss

Goals in Evolutionary Computation

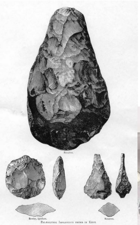
(RG-1) *Investigation.* Specifying optimization problems, analyzing algorithms. What could be a reasonable research question? What is going to be explained? Does it help in practice? Enables theoretical advances?

(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

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

- Fantastic tools to generate statistics:
  R, S-Plus, Matlab, Mathematica, SAS, ec.
- Nearly no tools to interpret scientific significance
- Stop! You might claim that more and more authors use *p*-values
- *p*-value to tackle the fundamental problem in every experimental analysis:
  Is the observed value, e.g., difference, meaningful?
- Next: Problems related to the *p*-value

High-Quality Statistics

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

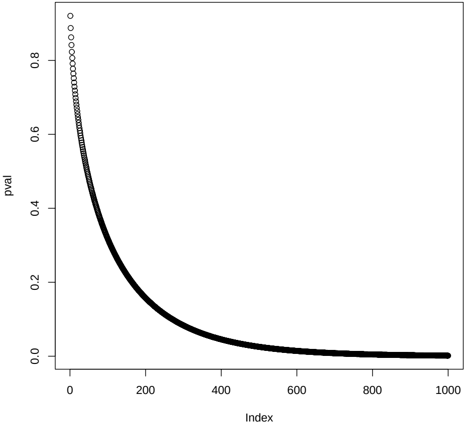
  Select test problem (instance) P
  Run algorithm A, say n times
  Obtain n fitness values: $x_{A,i}$
  Run algorithm B, say n times
  Obtain n fitness values: $x_{B,i}$

R-demo

- ```
  > n=100
  > run.algorithm1(n)
    [1]  99.53952  99.86982 101.65871...
  > run.algorithm2(n)
    [1]  99.43952  99.76982 101.55871...
  ```
- Now we have generated a plethora of important data - what is the next step?
- Select a test (statistic), e.g., the mean
- Set up a hypothesis, e.g., there is no difference
  ```

R-demo. Analysis

- Minimization problem
- For reasons of simplicity: Assume known standard deviation $\sigma = 1$
- Compare difference in means:

$$d(A, B, P, n) = \frac{1}{n} \sum_{i=1}^{n} (x_{A,i} - x_{B,i})$$

- Formulate hypotheses:

  H_0: $d <= 0$ there is no difference in means vs.
  H_1: $d > 0$ there is a difference (B is better than A)

R-demo. Analysis

- ```
  > n=5
  > run.comparison(n)
  [1] 0.8230633
  ```
- Hmmm, that does not look very nice. Maybe I should perform more comparisons, say $n = 10$
- ```
  > n=10
  > run.comparison(n)
  [1] 0.7518296
  ```
- Hmmm, looks only slightly better. Maybe I should perform more comparisons, say $n = 100$
- ```
  > n=100
  > run.comparison(n)
  [1] 0.3173105
  ```
- I am on the right way. A little bit more CPU-time and I have the expected results.
  ```
  > n=1000
  > run.comparison(n)
  [1] 0.001565402
  ```
- Wow, this fits perfectly.
  ```

Scientific? The Large n Problem

Figure: Nostradamus: Astronomy considered scientific — astrology not



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)

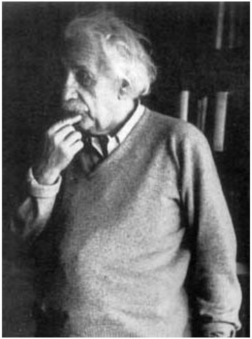
Research Question

- 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...

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

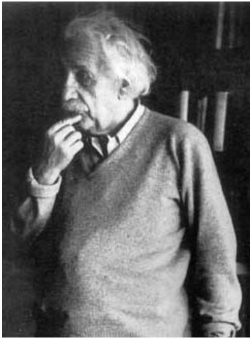
Research Question

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



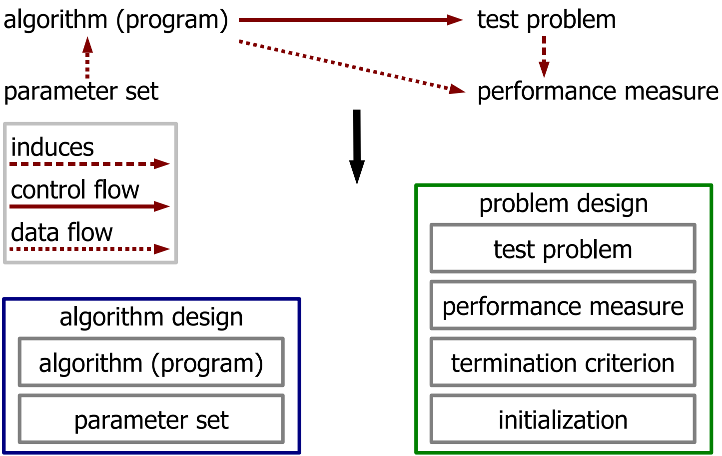
Einstein thinking

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

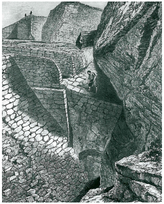


Figure: Schliemann in Troja

- “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

$\Rightarrow$ 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 $\Longrightarrow$ effectivity (robustness)
- Vary algorithm design $\Longrightarrow$ 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 $\Rightarrow$ 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_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

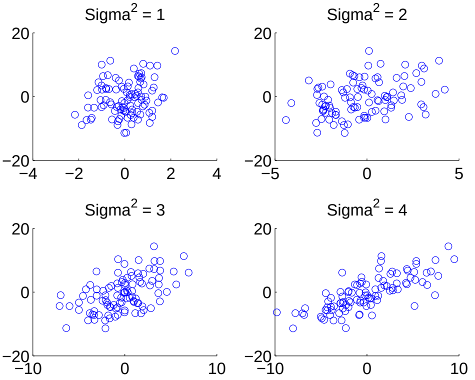
- Overview
  - Variance
  - Derivation
  - DoE: Regression coefficients (β)
  - DACE: Coefficients (θ)

Measures: Variance

Example (Toy problem)

$$Y = f(\vec{X}) = \sum_{i=1}^{r} \alpha X_i$$

- $X_i \sim N(0, \sigma_i^2)$
- $r = 4$, $\sigma_i^2 = i$



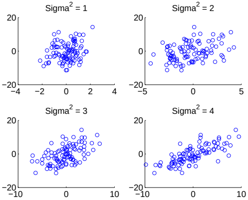
Measures: Variance

Example (Toy problem)

$$Y = f(\vec{X}) = \sum_{i=1}^{r} \alpha X_i$$

- Effect should produce shape or pattern
- Effect of factor

$$\frac{V_i(E_{-i}(Y|X_i))}{V(Y)}$$



- $Y = f(\vec{X}) = \sum_{i=1}^{r} \alpha X_i$ far too simple
- Which of the factors can be fixed without affecting Y
- Detect important less important factors
- Interactions

Measures: Derivation or Regression Based

- Derivation based measures
  - Evaluate the function at a set of different points in the problem domain
  - Define the *effect* of the ith factor as ratio

$$\frac{f(X_1, X_2, \ldots, X_i + h, \ldots, X_r) - f(X_1, \ldots X_r)}{h}$$

- Regression based measures
  - Relate the *effect* of the ith factor to its regression coefficient

$$Y = \beta_0 + \sum_{i=1}^{r} \beta_i X_i$$

  - Related: Kriging based measures

SPO Overview

Phase I Experiment construction

Phase II SPO core: Parameter optimization

Phase III Evaluation

- Phase I and III belong to the experimental methodology (how to perform experiments)
- Phase II is the parameter handling method, shall be chosen according to the overall research task (default method is provided)
- SPO is not *per se* a meta-algorithm: We are primarily interested in the resulting algorithm designs, not in the solutions to the primordial problem

SPO Workflow

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

---

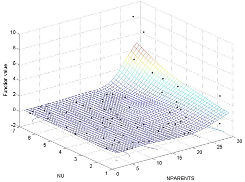
5. *Experiments*
6. Statistical *model* and prediction (DACE). Evaluation and visualization
7. Solution good enough?
   - Yes: Goto step 8
   - No: Improve the design (optimization). Goto step 5

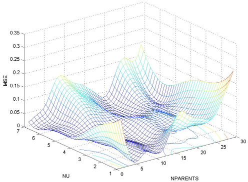
---

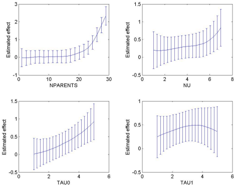
8. *Acceptance/rejection* of the statistical hypothesis
9. Objective *interpretation* of the results from the previous step

SPO in Action

- Sequential Parameter Optimization Toolbox (SPOT)
- Introduced in [BB06]







- Software can be downloaded from `http://ls11-www.cs.uni-dortmund.de/people/tom/ExperimentalResearchPrograms.html`

SPO Installation

- Create a new directory, e.g., `g:\myspot`
- Unzip SPO toolbox: `http://ls11-www.cs.uni-dortmund.de/people/tom/spot03.zip`
- Unzip MATLAB DACE toolbox: `http://www2.imm.dtu.dk/~hbn/dace/`
- Unzip ES package: `http://ls11-www.cs.uni-dortmund.de/people/tom/esmatlab03.zip`
- Start MATLAB
- Add `g:\myspot` to MATLAB path
- Run `demoSpotMatlab.m`

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

Summary: SPO Interfaces

- SPO requires CONF and ROI files
- SPO generates DES file
- Algorithm run with settings from DES
- Algorithm writes *result file* (RES)
- RES files read by SPO to generate stochastic process models
- RES files provide basis for many statistical evaluations/visualizations (EDA)

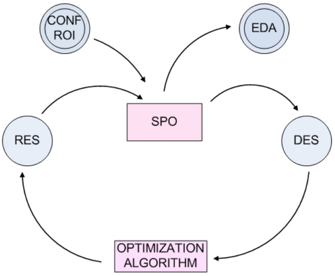


Figure: SPO Interfaces

Case study: Real-world optimization

- Real-world problem: Prediction
- Data-driven modeling
- New problem, no reference solutions
- How to chose an adequate method?
- How to tune the chosen prediction model?
- Take a look at the problem first
- Here: Prediction of fill levels in stormwater tanks

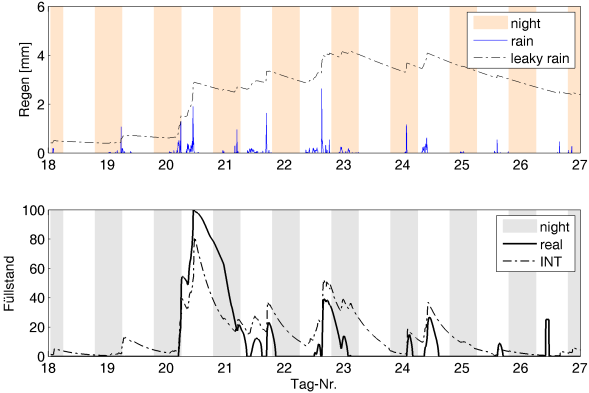
Case study: Prediction of fill levels in stormwater tanks

- Based on rain measurements and soil conditions
- Data
    - 150.000 data ...
    -
    - ...

Case study: Prediction of fill levels

- Goal:
  - Minimize prediction error for 108 days
  - Objective function
  - Fiction of optimization, see [**?**]
  - MSE

Case study: Prediction of fill levels



- Problem: Standard and CI-based modeling methods show larger prediction errors when trained on rain data with strong intermittent and bursting behaviour

Case study: Prediction of fill levels

- 6 Methods (many more available):
  - Neural Networks (NN)
  - Echo State Networks (ESN)
  - Nonlinear AutoRegressive models with eXogenous inputs (NARX)
  - Finite Impulse Response filter (FIR)
  - Differential equations (ODE)
  - Integral equations (INT)
- Details: [**?**]

Case study: Prediction of fill levels

- Each method has some parameters (here: 2 – 13)
- Problem design vs. algorithm design
- Parameter and factor

  Neural Networks (NN): not considered
  Echo State Networks (ESN): not considered
  Nonlinear AutoRegressive models with eXogenous inputs (NARX): 2, i.e., neurons and delay states
  Finite Impulse Response filter (FIR): 5, i.e., evaporation, delay, scaling, decay, length
  Differential equations (ODE): 6
  Integral equations (INT): 13
- Details: [**?**]

Case study: Prediction of fill levels

Table: Factors of the INT-Model. The ODE-Model uses a subset of 6 factors (shaded light gray): $\alpha, \beta, \tau_{\text{rain}}, \Delta, \alpha_L, \beta_L$.

Parameter	Symbol	manuell	Best SPO	Bereich SPO
Abklingkonstante Füllstand (Filter *g*)	α	0.0054	0.00845722	$[0, 0.02]$
Abklingkonstante Filter *h*	α_H	0.0135	0.309797	{0 ... 1}
Abklingkonstante 'leaky rain'	α_L	0.0015	0.000883692	{0 ... 0.0022}
Einkopplung Regen in Füllstand	β	7.0	6.33486	{0 ... 10}
Einkopplung Regen in 'leaky rain'	β_L	0.375	0.638762	{0 ... 2}
Einkopplung *K*-Term in Füllstand	h_0	0.5	6.87478	{0 ... 10}
Schwelle für 'leaky rain'	Δ	2.2	7.46989	{0 ... 10}
Flankensteilheit aller Filter	κ	1	1.17136	{0 ... 200}
Zeitverzögerung Füllstand zu Regen	τ_{rain}	12	3.82426	{0 ... 20}
Startzeitpunkt Filter *h*	τ_{in3}	0	0.618184	{0 ... 5}
Endzeitpunkt Filter *h*	τ_{out3}	80	54.0925	{0 ... 500}
Endzeitpunkt Filter *g*	τ_{out}	80	323.975	{0 ... 500}
RMSE		12.723	9.48588	

Case study: Prediction of fill levels in stormwater tanks

- SPO in a nutshell
  - I. Pre-experimental planning
  - II. Screening
  - III. Modeling and optimization

Case study: Prediction of fill levels

Step I: Pre-experimental planning

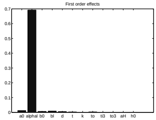
- Test runs, no planning possible
- No optimality conditions applicable
- Detect ROI intervals
- Intervals should courageously be chosen
- Treatment of infeasible factor settings (penalty)

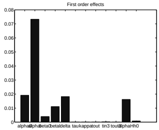
Case study: Prediction of fill levels

Step II: Screening

- Short run time
- Sparse design
- Consider extreme values
- Detect outliers that destroy the SPO meta-model

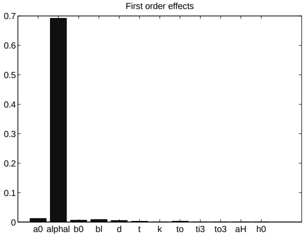
- Unbalanced factor effects indicate not correctly specified ROI



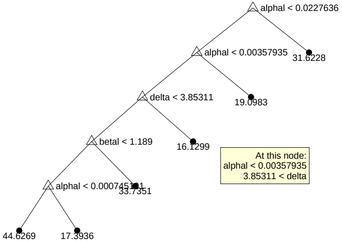


Case study: Prediction of fill levels
Step II: Screening

- Not correctly secified ROIs



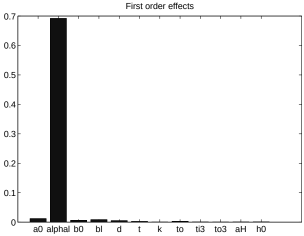
- Regression tree



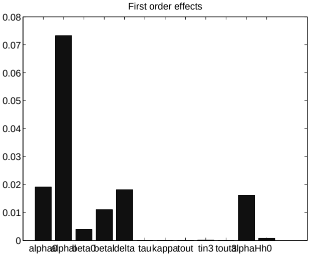
Case study: Prediction of fill levels

Step II: Screening

- Before



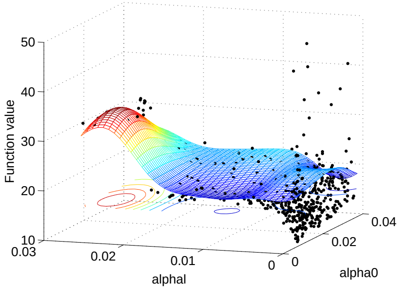
- After



Case study: Prediction of fill levels

Step III: Modeling and Optimization

- Reduced parameter set (INT: from 13 to 6)
- Complex design



Case study: Prediction of fill levels

Result

Table: Comparison. RSME

Method	randomized design	manually chosen	SPO
FIR	25.42	25.57	20.10
NARX	85.22	75.80	38.15
ODE	39.25	13.60	9.99
INT	31.75	12.72	9.49

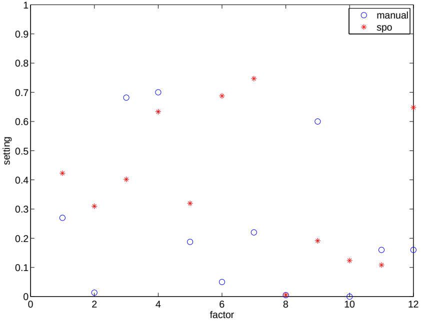
Case study: Prediction of fill levels in stormwater tanks

- Comparison of different prediction methods
- SPO to find in a comparable manner the best parameters for each method
- Standard and CI-based modeling methods show larger prediction errors when trained on rain data with strong intermittent and bursting behaviour
- Models developed specific to the problem show a smaller prediction error
- SPO is applicable to diverse forecasting methods and automates the time-consuming parameter tuning
- Best manual result achieved before was improved with SPO by 30%
- SPO analyses in a consistent manner the parameter influence and allows a purposeful simplification and/or refinement of the model design

Case study: Prediction of fill levels in stormwater tanks

Results

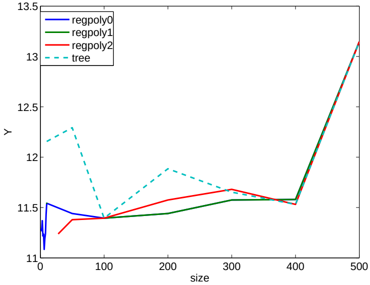
- Ranges
- No bias, no systematic error



Case study: Prediction of fill levels

Results

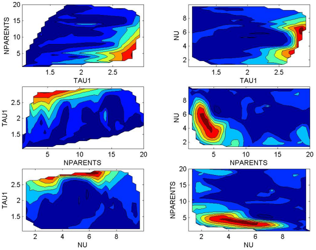
- Design considerations
- How many design points are necessary?
- Initial design size?

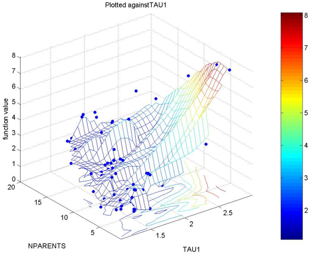


SPO and EDA

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

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





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

1

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

  Keep One, Send One (KOSO) to my right neighbor

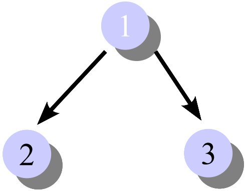
  Balanced strategy KOSO*: Send to neighbor with lower load only
- Is KOSO* better than KOSO?

Empirical Analysis: Algorithms for Scheduling Problems

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

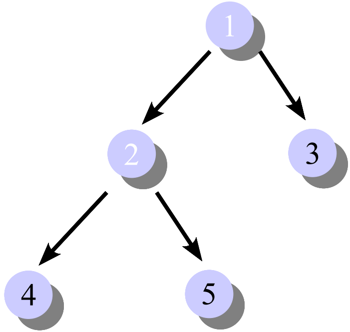
  Keep One, Send One (KOSO) to my right neighbor

  Balanced strategy KOSO*: Send to neighbor with lower load only
- Is KOSO* better than KOSO?



Empirical Analysis: Algorithms for Scheduling Problems

- 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?

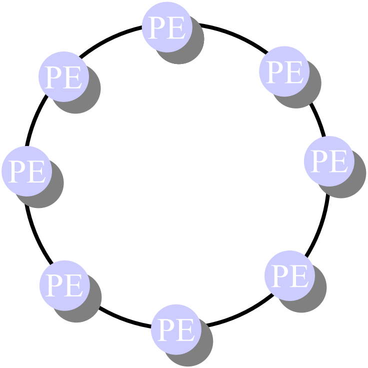


Empirical Analysis: Algorithms for Scheduling Problems

- 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

- 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

There Is a Problem With the Experiment

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

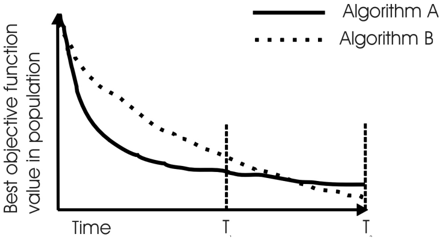
- Current approach: Either do not mention it, or redo everything
- If redoing is easy, nothing is lost
- If it is not, we must either:
  - Let people know about it, explaining why it probably does not change results
  - Or do validation on a smaller subset: How large is the difference (e.g. statistically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

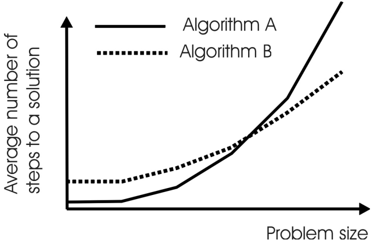
"Traditional" Measuring in EC

Simple Measures

- MBF: mean best fitness
- AES: average evaluations to solution
- SR: success rates, SR(t) $\Rightarrow$ 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] for different expected lengths, unsuccessful runs are stopped at FE_{max}:
$$SP2 = \frac{1 - p_s}{p_s} FE_{max} + \mathbb{E}(T_A^s) \tag{2}$$

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

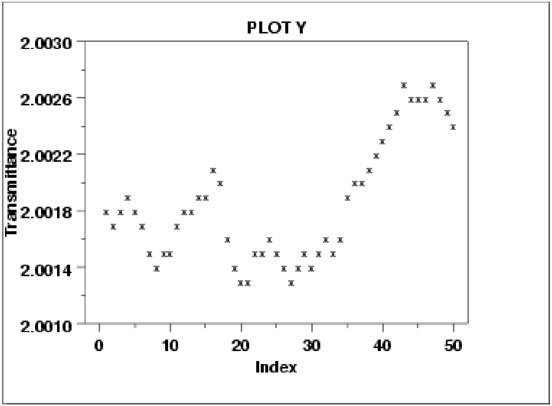
In real-world optimization, 10^4 evaluations is a lot, sometimes only 10^3 or less is possible:

- We are relieved from choosing termination criteria
- Substitute models may help (Algorithm based validation)
- We encourage more research on short runs

Selecting a performance measure is a *very* important step

Diagrams Instead of Tables

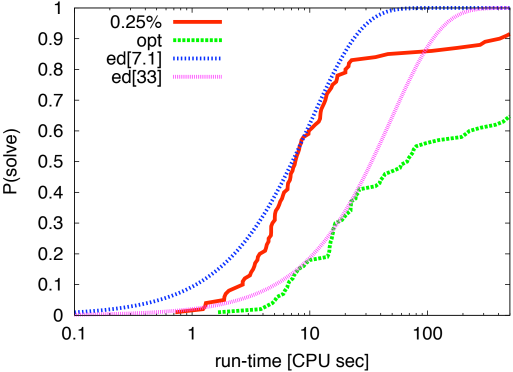
Would You Have Seen This From a Table?



Sequence plot

Visual Comparison With a Task Set

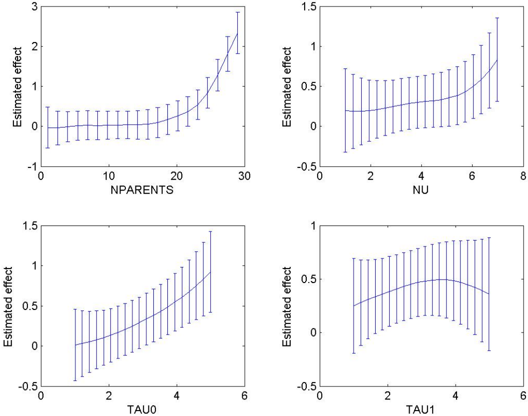
Run-length distributions



(courtesy of Thomas Stuetzle)

(Single) Effect Plots

Useful, but not Perfect

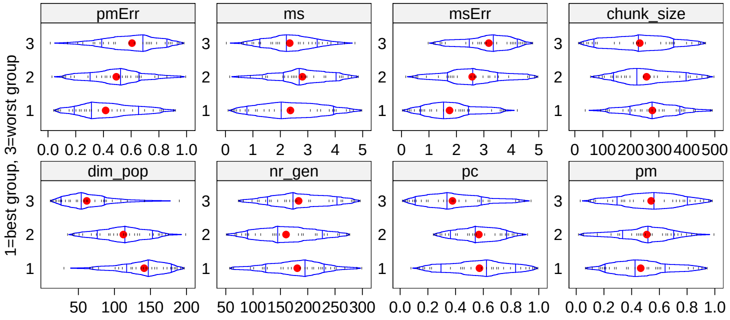


- Large variances originate from averaging
- The τ_0 and especially τ_1 plots show different behavior on extreme values (see error bars), probably distinct (averaged) effects/interactions

One-Parameter Effect Investigation

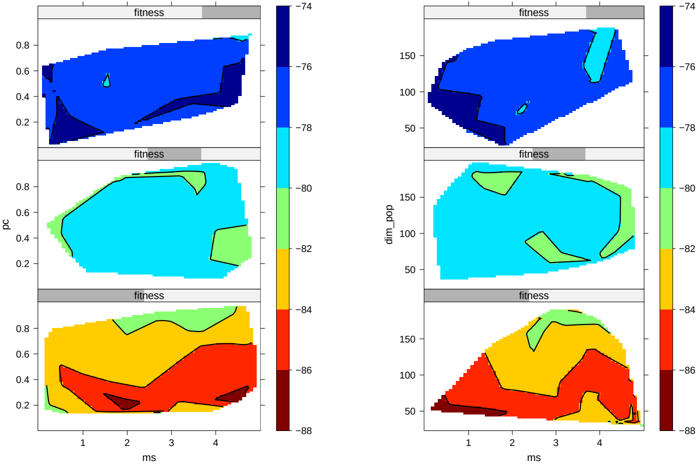
Effect Split Plots: Effect Strengths

- 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

$\Rightarrow$ 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?
  $\Rightarrow$ *Rules of thumb, finally theory*

[1] no free lunch theorem

The Art of Comparison
Efficiency vs. Adaptability

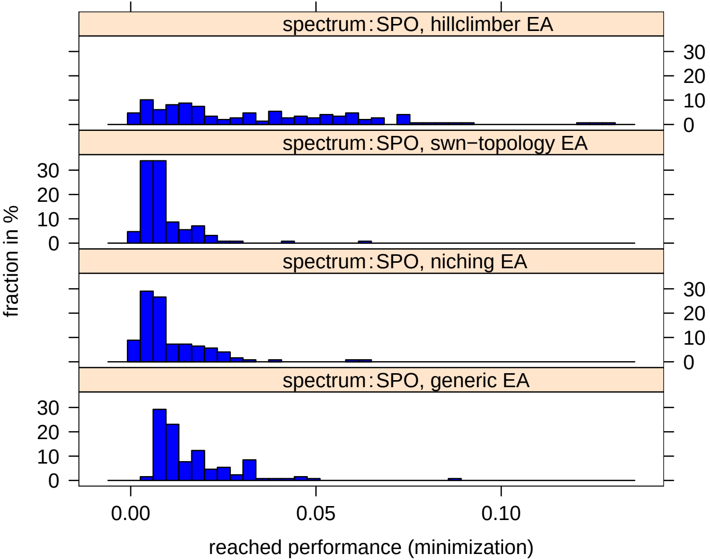
Most existing experimental studies focus on the efficiency of optimization algorithms, but:

- 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)

Possible Alternatives?

Parameterless EAs:

- Easy to apply, but what about performance and robustness?
- Where did the parameters go?

Usually a mix of:

- Default values, sacrificing top performance for good robustness
- Heuristic rules, applicable to *many* but not *all* situations; probably not working well for completely new applications
- (Self-)Adaptation techniques, these cannot learn too many parameter values at once, and not necessarily reduce the number of parameters

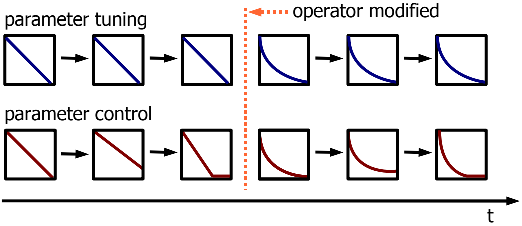
$\Rightarrow$ We can reduce number of parameters, but usually at the cost of either performance or robustness

Parameter Control or Parameter Tuning?

The time factor:

- Parameter control: during algorithm run
- Parameter tuning: before an algorithm is run

But: Recurring tasks, restarts, or adaptation (to a problem) blur this distinction

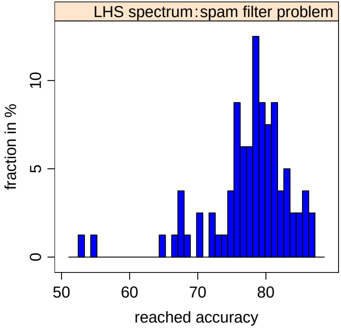


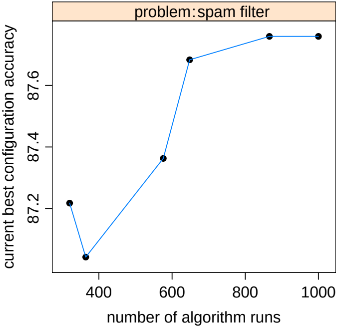
And: How to find meta-parameter values for parameter control?
⇒ Parameter control *and* parameter tuning

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

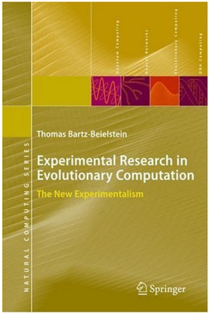
- Testing against mean values is statistically not meaningful
- But giving lots of data is not always possible (page limit)
- Many online sources (e.g. ACM JEA) allow for storing data

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*!

Updates



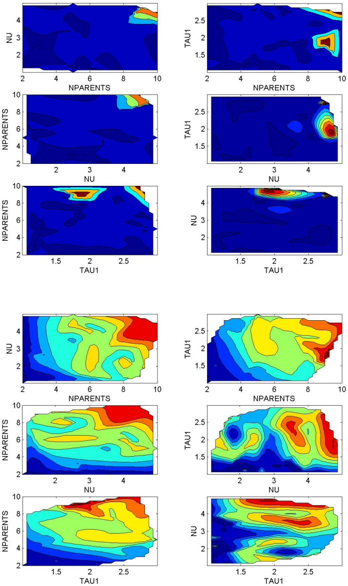
- Please check `http://www.gm.fh-koeln.de/~bartz/experimentalresearch/ExperimentalResearch.html` for updates, software, etc.
- To appear 2009: Empirical Methods for the Analysis of Optimization Algorithms
- See also Kleijnen, Saltelli et al.

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
- ...

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 ${}_p^{(1)}$
  - ES with, say $\mu = 2$, performs better than ES with *mu* > 2 if compared on problem design ${}_p^{(2)}$
  - ...
  - ES with, say $\mu = 2$, performs better than ES with *mu* > 2 if compared on problem design ${}_p^{(n)}$



SPO Core: Default Method

Heuristic for Stochastically Disturbed Function Values

- Start with latin hypercube sampling (LHS) design: Maximum spread of starting points, small number of evaluations
- Sequential enhancement, guided by DACE model
- Expected improvement: Compromise between optimization (min Y) and model exactness (min MSE)
- Budget-concept: Best search points are re-evaluated
- Fairness: Evaluate new candidates as often as the best one

Table: Current best search points recorded by SPO, initial LHS

$\frac{\lambda}{\mu}$	τ_0	*restart threshold*	*#eval best*	*config ID*	*result*	*std. deviation*
10.075	0.4180	22	4	42	0.0034	0.0058
5.675	0.7562	2	4	72	0.0042	0.0035
10.625	0.0796	5	4	57	0.0042	0.0054
4.905	0.1394	10	4	86	0.0047	0.0068
3.585	0.0398	13	4	81	0.0048	0.0056
3.145	0.0200	8	4	3	0.0050	0.0056
2.595	0.7960	4	4	83	0.0065	0.0048
2.375	1.8905	7	4	64	0.0113	0.0115

SPO Core: Default Method

Heuristic for Stochastically Disturbed Function Values

- Start with latin hypercube sampling (LHS) design: Maximum spread of starting points, small number of evaluations
- Sequential enhancement, guided by DACE model
- Expected improvement: Compromise between optimization (min Y) and model exactness (min MSE)
- Budget-concept: Best search points are re-evaluated
- Fairness: Evaluate new candidates as often as the best one

Table: Current best search points recorded by SPO, step 7

$\frac{\lambda}{\mu}$	τ_0	*restart threshold*	*#eval best*	*config ID*	*result*	*std. deviation*
5.675	0.7562	2	4	72	0.0042	0.0035
10.625	0.0796	5	4	57	0.0042	0.0054
4.905	0.1394	10	4	86	0.0047	0.0068
3.585	0.0398	13	4	81	0.0048	0.0056
3.145	0.0200	8	4	3	0.0050	0.0056
2.595	0.7960	4	4	83	0.0065	0.0048
3.866	0.0564	4	8	106	0.0096	0.0065
2.375	1.8905	7	4	64	0.0113	0.0115
...	...	...	...	...	...	...
10.075	0.4180	22	8	42	0.0177	0.0181

SPO Core: Default Method

Heuristic for Stochastically Disturbed Function Values

- Start with latin hypercube sampling (LHS) design: Maximum spread of starting points, small number of evaluations
- Sequential enhancement, guided by DACE model
- Expected improvement: Compromise between optimization (min Y) and model exactness (min MSE)
- Budget-concept: Best search points are re-evaluated
- Fairness: Evaluate new candidates as often as the best one

Table: Current best search points recorded by SPO, step 12

$\frac{\lambda}{\mu}$	τ_0	*restart threshold*	*#eval best*	*config ID*	*result*	*std. deviation*
10.625	0.0796	5	10	57	0.0024	0.0038
5.675	0.7562	2	5	72	0.0042	0.0031
4.905	0.1394	10	4	86	0.0047	0.0068
3.585	0.0398	13	4	81	0.0048	0.0056
3.145	0.0200	8	4	3	0.0050	0.0056
11.620	0.0205	2	10	111	0.0055	0.0052
2.595	0.7960	4	4	83	0.0065	0.0048
3.866	0.0564	4	8	106	0.0096	0.0065

SPO Core: Default Method

Heuristic for Stochastically Disturbed Function Values

- Start with latin hypercube sampling (LHS) design: Maximum spread of starting points, small number of evaluations
- Sequential enhancement, guided by DACE model
- Expected improvement: Compromise between optimization (min Y) and model exactness (min MSE)
- Budget-concept: Best search points are re-evaluated
- Fairness: Evaluate new candidates as often as the best one

Table: Current best search points recorded by SPO, step 17

$\frac{\lambda}{\mu}$	τ_0	*restart threshold*	*#eval best*	*config ID*	*result*	*std. deviation*
10.625	0.0796	5	20	57	0.0023	0.0034
4.881	0.0118	8	20	116	0.0028	0.0029
5.675	0.7562	2	5	72	0.0042	0.0031
4.905	0.1394	10	4	86	0.0047	0.0068
3.585	0.0398	13	4	81	0.0048	0.0056
3.145	0.0200	8	4	3	0.0050	0.0056
11.620	0.0205	2	10	111	0.0055	0.0052
7.953	0.0213	2	10	114	0.0065	0.0055

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.
Springer, Berlin, Heidelberg, New York, 2006.

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.

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