

Experimental Research in Evolutionary Computation

Thomas Bartz-Beielstein¹ Mike Preuss²

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

²Department of Computer Science
Technical University of Dortmund

Sunday, 13 July 2008

Copyright is held by the author/owner(s).

GECCO'08, July 7-11, 2008, Atlanta, Georgia, USA.

ACM 978-1-60558-131-6/08/07.

Bartz-Beielstein, Preuss (Cologne, Dortmund)

Experimental Research

Sunday, 13 July 2008 1 / 65

intro goals

Goals in Evolutionary Computation

- (RG-1) *Investigation*. Specifying optimization problems, analyzing algorithms. Important parameters; what should be optimized?
- (RG-2) *Comparison*. Comparing the performance of heuristics
- (RG-3) *Conjecture*. Good: demonstrate performance. Better: explain and understand performance
- (RG-4) *Quality*. Robustness (includes insensitivity to exogenous factors, minimization of the variability) [Mon01]

Overview

- 1 Introduction
 - Goals
 - History
 - Drawing Conclusions
 - Tomorrow
- 2 How to set up an experimental study
 - Objective of the experimental analysis
 - Measures
 - Example: Evolution strategy
- 3 SPO Toolbox (SPOT)
 - Demo
- 4 Rethinking Experimentation
- Beyond the NFL
 - Parametrized Algorithms
 - Parameter Tuning
- 5 Sequential Parameter Optimization
 - Similar Approaches
 - Example: Rosenberg Study
 - Basics
 - Overview
 - Models
 - Heuristic
- 6 Tools for Practical Experimentation
 - Performance Measuring
 - Reporting Experiments
 - Visualization

Bartz-Beielstein, Preuss (Cologne, Dortmund)

Experimental Research

Sunday, 13 July 2008 2 / 65

intro goals

What happend so far?

- Presentations at MIC, HM, PPSN, CEC, GECCO, ...
- More than a dozen talks, workshops and tutorials
- A few tens publications
- Next steps?

Bartz-Beielstein, Preuss (Cologne, Dortmund)

Experimental Research

Sunday, 13 July 2008 3 / 65

Bartz-Beielstein, Preuss (Cologne, Dortmund)

Experimental Research

Sunday, 13 July 2008 4 / 65

Next steps

- Learning from history
- Some ideas presented today



- 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

A Totally Subjective History of Experimentation in Evolutionary Computation

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}
- Problems of today: Adequate statistical methods, but wrong scientific conclusions
- 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

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
- Fundamental problem in every experimental analysis: **Is the observed value, e.g., difference, meaningful?**
- Standard statistic: p -value
- 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

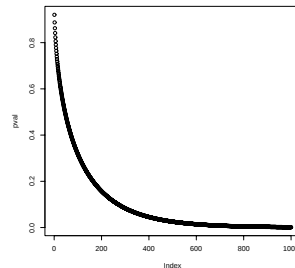
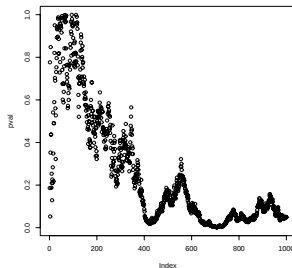
- Minimization problem
- For reasons of simplicity: Assume known standard deviation  $\sigma = 1$
- Compare difference in means:

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

- Formulate hypotheses:  
 $H_0: d \leq 0$  there is no difference in means vs.  
 $H_1: d > 0$  there is a difference ( $B$  is better than  $A$ )

## R-demo. Analysis

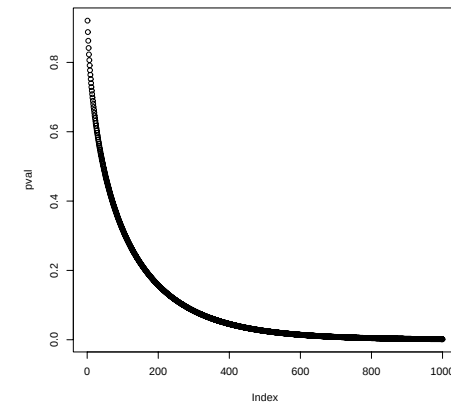
- `> n=5`  
`> run.comparison(n)`  
`[1] 0.8230633`
- Hmm, that does not look very nice. Maybe I should perform more comparisons, say  $n = 10$
- `> n=10`  
`> run.comparison(n)`  
`[1] 0.7518296`
- Hmm, looks only slightly better. Maybe I should perform more comparisons, say  $n = 100$
- `> n=100`  
`> 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.

Large  $n$  problem

## Scientific?



Figure: Nostradamus



## Tomorrow: Correct Statistics and Correct Conclusions

- Tomorrow:
  - 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



## Why are we performing experiments?

- Why are we interested in improving the algorithm's performance?
  - Because it does not find any feasible solution or
  - Because it has to be competitive to the best known algorithm
- How do we define importance or significance?
- Many statistics available
- Each measure will produce its own ranking
- Planning of experiments

## 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
- First classification:
  - algorithm
  - problem

## Classification

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

- $\vec{X} = (X_1, X_2, \dots, X_r)$  denote  $r$  factors from the algorithm design and
- $Y$  denotes some output (i.e., best function value from 1000 generations)
- Problem design remains unchanged

## Measures for factor effects

- Overview

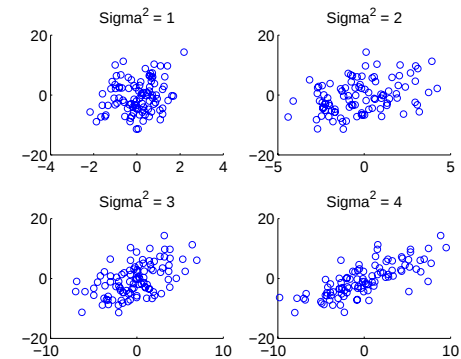
Statistics: Variance ( $V$ 's)  
 Calculus: Derivation ( $\partial$ 's)  
 DoE: Regression coefficients ( $\beta$ 's)  
 DACE: Coefficients ( $\theta$ 's)  
 Graphics: Visualizations

## Measures: Variance

### Example (Toy problem)

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

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



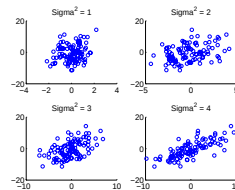
## Measures: Variance

## Example (Toy problem)

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

- Effect should produce shape or pattern
- Effect of factor

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



## Measures: Variance

## Example (Summary: Toy problem)

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

## Measures: Derivation based

- Evaluate the function at a set of different points in the problem domain
- Define the *effect* of the  $i$ th factor as incremental ratio

$$\frac{f(X_1, X_2, \dots, X_i + \Delta, \dots, X_r) - f(X_1, \dots, X_r)}{\Delta}$$

## Measures: regression based

- Regression based measures
- Relate the *effect* of the  $i$ th factor to its regression coefficient

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

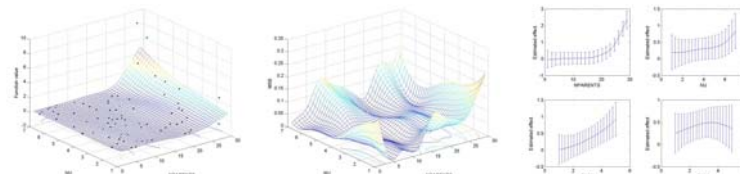
- Related: Kriging based measures

## Screening Evolution Strategies

- Detect important factors
- CMA-ES [HO01]
- Screening uses tools from SPOT

## SPO in Action

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



- Software can be downloaded from <http://www.gm.fh-koeln.de/~bartz/experimentalresearch/spot.zip>

## SPO Installation

- Create a new directory, e.g., `g:\myspot`
- Unzip SPO toolbox
- Unzip MATLAB DACE toolbox:  
`http://www2.imm.dtu.dk/~hbn/dace/`
- Unzip CM-ES package from Nikolas Hansen's WWW-page.
- Start MATLAB
- Add `g:\myspot` to MATLAB path
- Run `spotdriver('demo1000')`

## 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: demo1000.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: demo1000.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: demo1000.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: demo1000.res

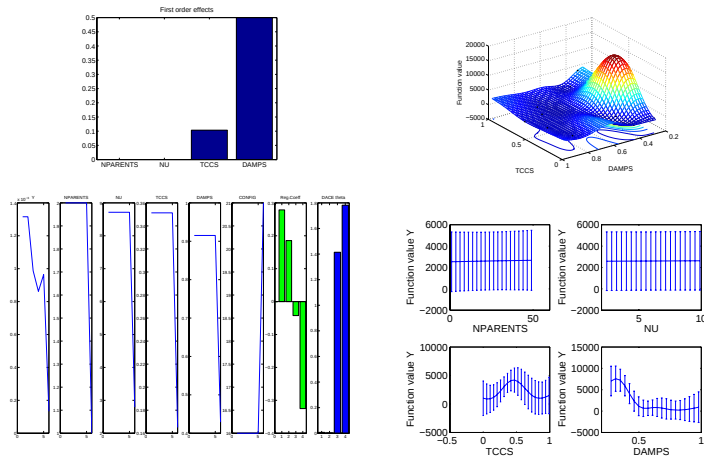
## Summary: SPO Interfaces

- 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

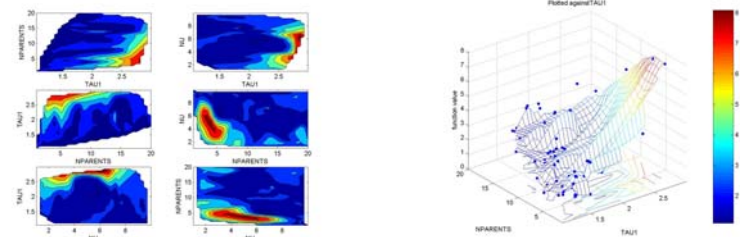
## SPO live demo: some impressions



## 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
  - Tree-based models
- Designs?
  - Space filling
  - Factorial
  - Combinations
- Statistical tools
- Significance
- Standards
- TBD
  - Provide SPOT interfaces for important optimization algorithms
  - Tools to derive meta-statistical rules
  - Other tools needed, because  $p$  value is not sufficient

The Art of Comparison  
*Orientation*

The NFL<sup>1</sup> 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?

<sup>1</sup>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.)

## 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  
⇒ 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

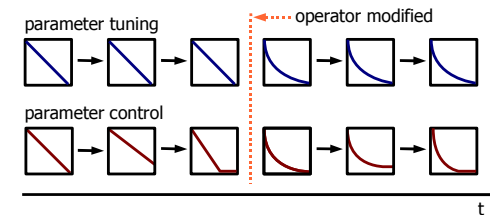
⇒ We can reduce number of parameters, but usually at the cost of either performance or robustness

## 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



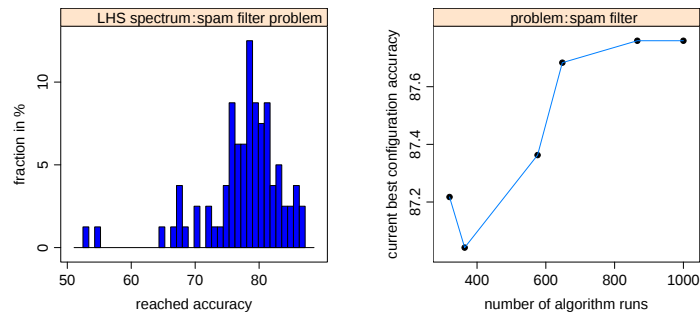
And: How to find meta-parameter values for parameter control?

⇒ 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:

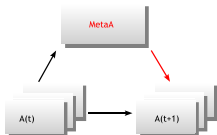
- How much amount to put into tuning (fixed budget, until stagnation)?
- Where shall we be on the spectrum when we compare?
- Can we compare spectra ( $\Rightarrow$  adaptability)?

## Similarities and Differences to Existing Approaches

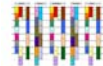
- Agriculture, industry: Design of Experiments (DoE)



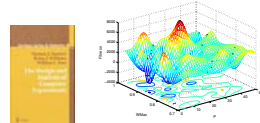
- Evolutionary algorithms: Meta-algorithms



- Algorithm engineering: Rosenberg Study (ANOVA)



- Statistics: Design and Analysis of Computer Experiments (DACE)



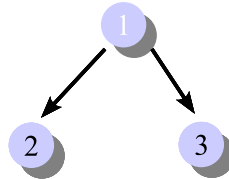
## Empirical Analysis: Algorithms for Scheduling Problems

- 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?

1

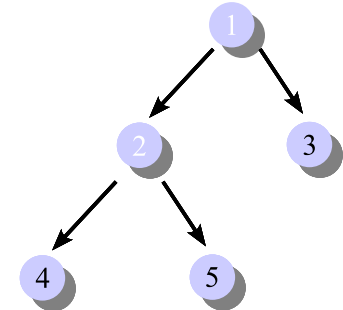
## Empirical Analysis: Algorithms for Scheduling Problems

- 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?



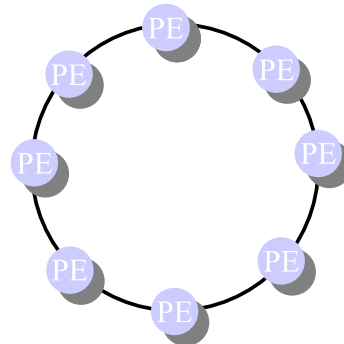
## Empirical Analysis: Algorithms for Scheduling Problems

- 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

## Designs

- Sequential Parameter Optimization based on
  - Design of Experiments (DOE)
  - Design and Analysis of Computer Experiments (DACE)
- Optimization run = experiment
- Parameters = design variables or factors
- Endogenous factors: modified during the algorithm run
- Exogenous factors: kept constant during the algorithm run
  - Problem specific
  - Algorithm specific

## SPO Overview

- 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, Regression trees, etc.).  
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: severity etc.

## Statistical Model Building and Prediction

*Design and Analysis of Computer Experiments (DACE)*

- Response  $Y$ : Regression model and random process
- Model:

$$Y(x) = \sum_h \beta_h f_h(x) + Z(x)$$

- $Z(\cdot)$  correlated random variable
- Stochastic process.
- **DACE stochastic process model**
- Until now: DACE for **deterministic** functions, e.g. [SWN03]
- New: DACE for **stochastic** functions

## Expected Model Improvement

Design and Analysis of Computer Experiments (DACE)

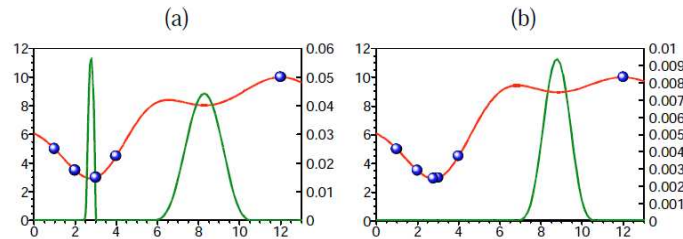


Figure: Axis labels left: function value, right: expected improvement. Source: [JSW98]

- (a) Expected improvement: 5 sample points
- (b) Another sample point  $x = 2.8$  was added

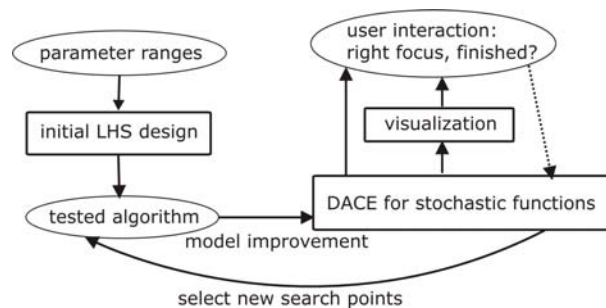
## Heuristic for Stochastically Disturbed Function Values

- 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 ( $\min \text{MSE}$ )
- Budget-concept: Best search points are re-evaluated
- Fairness: Evaluate new candidates as often as the best one

Table: SPO. Algorithm design of the best search points

| $Y$   | $s$ | $c_1$ | $c_2$ | $W_{\max}$ | $W_{\text{scale}}$ | $W_{\text{iter}}$ | $v_{\max}$ | Conf. | $n$ |
|-------|-----|-------|-------|------------|--------------------|-------------------|------------|-------|-----|
| 0.055 | 32  | 1.8   | 2.1   | 0.8        | 0.4                | 0.5               | 9.6        | 41    | 2   |
| 0.063 | 24  | 1.4   | 2.5   | 0.9        | 0.4                | 0.7               | 481.9      | 67    | 4   |
| 0.061 | 32  | 1.8   | 2.1   | 0.8        | 0.4                | 0.5               | 9.6        | 41    | 4   |
| 0.058 | 32  | 1.8   | 2.1   | 0.8        | 0.4                | 0.5               | 9.6        | 41    | 8   |

## Data Flow and User Interaction



- User provides parameter ranges and tested algorithm
- Results from an LHS design are used to build model
- Model is improved incrementally with new search points
- User decides if parameter/model quality is sufficient to stop

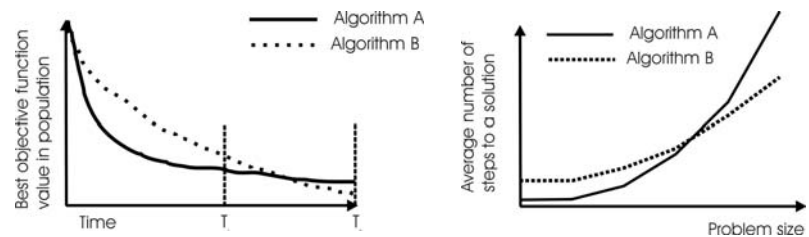
□

## “Traditional” Measuring in EC

Simple Measures

- MBF: mean best fitness
- AES: average evaluations to solution
- SR: success rates,  $\text{SR}(t) \Rightarrow$  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  $\mathbb{E}(T^s) = \mathbb{E}(T^{us})$ :

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

- SP2 [AH05] for different expected lengths, unsuccessful runs are stopped at  $FE_{max}$ :

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

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

## Current “State of the Art”

Around 40 years of empirical tradition in EC, but:

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

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

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

⇒ We suggest a 7-part reporting scheme

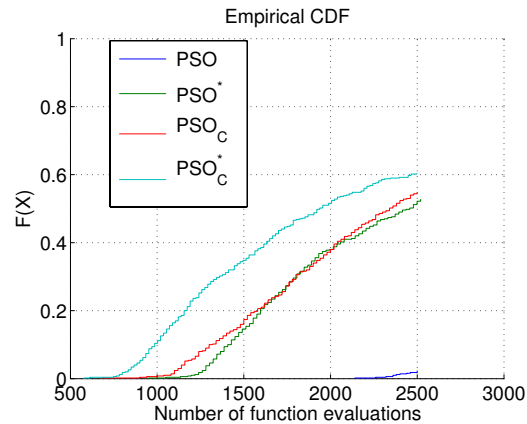
## Suggested Report Structure

- ER-1: **Research Question** the matter dealt with
- ER-2: **Pre-experimental planning** first—possibly explorative—program runs, leading to task and setup
- ER-3: **Task** main question and scientific and derived statistical hypotheses to test
- ER-4: **Setup** problem and algorithm designs, sufficient to replicate an experiment
- ER-5: **Results/Visualization** raw or produced (filtered) data and basic visualizations
- ER-6: **Observations** exceptions from the expected, or unusual patterns noticed, plus additional visualizations, no subjective assessment
- ER-7: **Discussion** test results and necessarily subjective interpretations for data and especially observations

This scheme is well suited to report 12-step SPO experiments

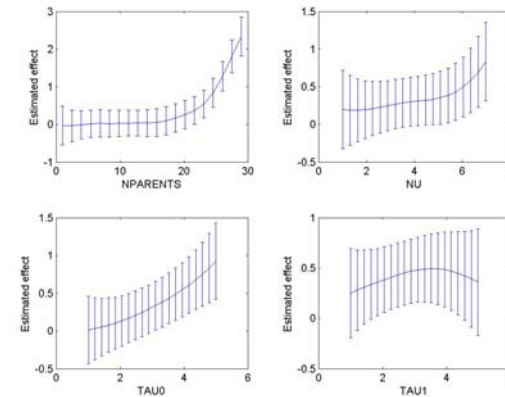
## Objective Interpretation of the Results

*Comparison. Run-length distribution*



## (Single) Effect Plots

*Useful, but not Perfect*

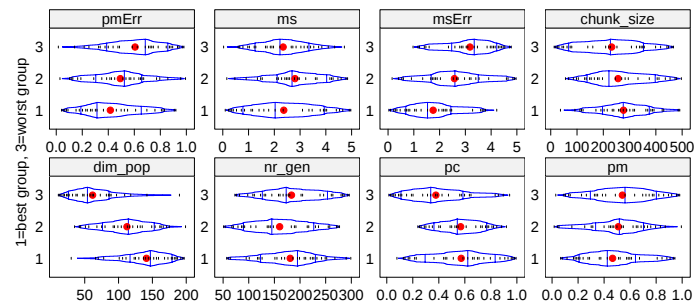


- Large variances originate from averaging
- The  $\tau_0$  and especially  $\tau_1$  plots show different behavior on extreme values (see error bars), probably distinct (averaged) effects/interactions

## One-Parameter Effect Investigation

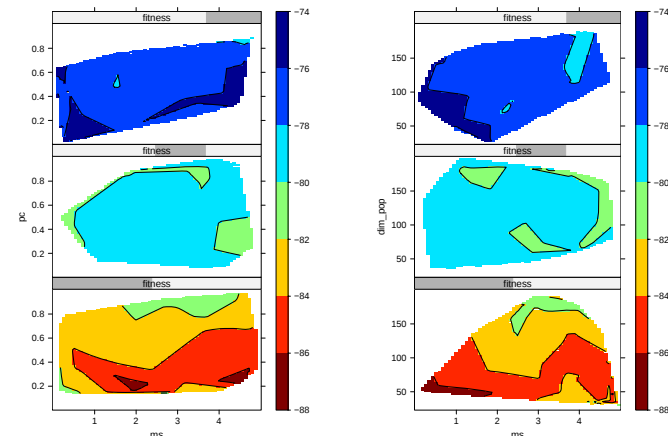
*Effect Split Plots: Effect Strengths*

- 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 Levelled Effects*



## Updates



- Please check  
<http://www.gm.fh-koeln.de/~bartz/experimentalresearch/ExperimentalResearch.html>  
 for updates, software, etc.

## 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
- ...



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.



N. Hansen and A. Ostermeier.

Completely derandomized self-adaptation in evolution strategies.

*Evolutionary Computation*, 9(2):159–195, 2001.



D.R. Jones, M. Schonlau, and W.J. Welch.

Efficient global optimization of expensive black-box functions.

*Journal of Global Optimization*, 13:455–492, 1998.



D. C. Montgomery.

*Design and Analysis of Experiments*.

Wiley, New York NY, 5th edition, 2001.



T. J. Santner, B. J. Williams, and W. I. Notz.

*The Design and Analysis of Computer Experiments*.

Springer, Berlin, Heidelberg, New York, 2003.