Experimental Research in Evolutionary Computation

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
University of Dortmund

Saturday, 7 July 2007

GECCO´07, July 7-11, 2007, London, England, United Kingdom.

ACM 978-1-59593-697-4/07/0007.

Overview

Scientific Goals?

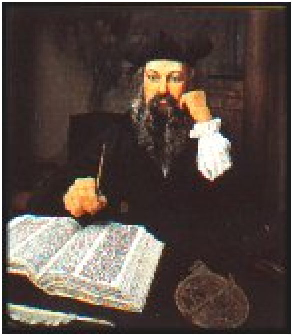


Figure: Nostradamus

- Why is astronomy considered scientific—and astrology not?
- And what about experimental research in EC?

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

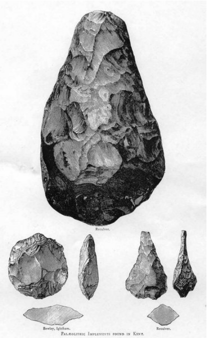
(RG-3) *Conjecture.* Good: demonstrate performance. Better: explain and understand performance

(RG-4) *Quality.* Robustness (includes insensitivity to exogenous factors, minimization of the variability) [Mon01]

Goals in Evolutionary Computation

- Given: Hard real world optimization problems, e.g., chemical engineering, airfoil optimization, bioinformatics
- Many theoretical results are too abstract, do not match with reality
- Real programs, not algorithms
- Develop problem specific algorithms, experimentation is necessary
- Experimentation requires statistics

A Totally Subjective History of Experimentation in Evolutionary Computation



- Palaeolithic
- Yesterday
- Today
- Tomorrow

Stone Age: Experimentation Based on Mean Values

- First phase (foundation and development, before 1980)
- Comparison based on mean values, no statistics
- Development of standard benchmark sets (sphere function etc.)
- Today: Everybody knows that mean values are not sufficient

Stone Age Example: Comparison Based on Mean Values

Example (PSO swarm size)

- Experimental setup:
  - 4 test functions: Sphere, Rosenbrock, Rastrigin, Griewangk
  - Initialization: asymmetrically
  - Termination: maximum number of generations
  - PSO parameter: default
- Results: Table form, e.g.,

Table: Mean fitness values for the Rosenbrock function

Population	Dimension	Generation	Fitness
20	10	1000	96,1725
20	20	1500	214,6764

- Conclusion: “Under all the testing cases, the PSO always converges very quickly”

Yesterday: Mean Values and Simple Statistics

- Second phase (move to mainstream, 1980-2000)
- Statistical methods introduced, mean values, standard deviations, tutorials
- *t* test, *p* value, . . .
- Comparisons mainly on standard benchmark sets
- Questionable assumptions (NFL)

Yesterday: Mean Values and Simple Statistics

Example (GAs are better than other algorithms (on average))

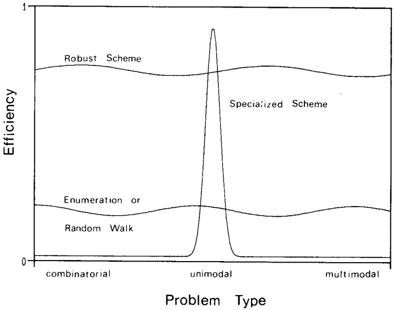


Figure: [Gol89]

Theorem (NFL)

There is no algorithm that is better than another over all possible instances of optimization problems

Today: Based on Correct Statistics

- Third phase (Correct statistics, since 2000)
  - Statistical tools for EC
  - Conferences, tutorials, workshops, e.g., Workshop On Empirical Methods for the Analysis of Algorithms (EMAA) (`http://www.imada.sdu.dk/~marco/EMAA`)
  - New disciplines such as algorithm engineering
- But: There are three kinds of lies: lies, damned lies, and statistics (Mark Twain or Benjamin Disraeli), why should we care?
- Because it is the only tool we can rely on (at the moment,i.e., 2007)

Today: Based on Correct Statistics

Example (Good practice)

Table 3: Results of the algorithms with population of 20

Test functions	SGA mean best (std. dev.)	FDGA: OGA mean best (std. dev.)	FDGA: MGA mean best (std. dev.)	FDGA: IGA mean best (std. dev.)	t-value between SGA to the best FDGA	Best algorithm
f_1	8.0607e+000 1.5537e+000	8.5689e+000 1.6671e+000	8.6545e+000 1.5069e+000	8.2722e+000 1.5728e+000	-6.76 +	SGA
f_2	7.8010e-001 4.5832e+000	4.2479e+000 1.3211e+000	3.5444e+000 2.0873e+000	3.5093e+000 1.4978e+000	-4.00 +	SGA
f_3	6.4957e+000 1.8003e+000	9.2728e+000 1.8037e+000	8.6608e+000 1.8614e+000	8.6378e+000 1.9866e+000	-5.65 +	SGA
f_4	1.3506e+002 3.3349e+002	9.2200e+002 2.8070e+002	8.2073e+002 2.5999e+002	8.2272e+002 2.4858e+002	-11.69 +	SGA
f_5	2.7476e-002 3.0828e-002	6.8234e-002 5.4773e-002	8.2052e-002 5.2042e-002	6.2478e-002 5.5991e-002	-3.87 +	SGA
f_6	2.0791e-003 9.1846e-004	2.7050e-005 3.5287e-006	2.5915e-005 3.3219e-006	2.5830e-005 2.7375e-006	15.81 +	FDGA
f_7	2.0791e-003 9.1846e-004	4.3338e-011 7.5496e-012	4.0195e-011 8.0494e-012	4.0062e-011 8.3297e-012	1.91 +	FDGA
f_8	7.1211e+001 7.1211e+001	5.0154e+001 4.1123e+001	5.1774e+001 3.7574e+001	4.0649e+001 4.1068e+001	3.13 +	FDGA
f_9	1.4856e-001 6.2373e-002	5.1283e-002 4.1936e-003	4.6518e-002 1.6727e-002	4.6506e-002 1.2852e-002	11.33 +	FDGA
f_{10}	9.2123e-002	7.2324e-002	6.4803e-002	6.4846e-002	2.94 +	FDGA

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

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

Today: Based on Correct Statistics

- We claim: Fundamental ideas from statistics are misunderstood!
- For example: What is the p value?

Definition (p value)

The p value is the probability that the null hypothesis is true

Today: Based on Correct Statistics

- We claim: Fundamental ideas from statistics are misunderstood!
- For example: What is the p value?

Definition (p value)

The p value is the probability that the null hypothesis is true. No!

Today: Based on Correct Statistics

- We claim: Fundamental ideas from statistics are misunderstood!
- For example: What is the *p* value?

Definition (*p* value)

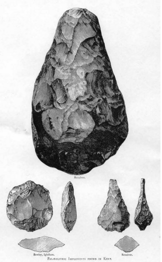
The *p* value is $p = P\{$ result from test statistic, or greater $\mid$ null model is true $\}$

- $\Rightarrow$ The *p* value is not related to any probability whether the null hypothesis is true or false

Tomorrow: Correct Statistics and Correct Conclusions

- Problems of today:
  Adequate statistical methods, but wrong scientific conclusions
- Generally: Statistical tools to decide whether *a* is better than *b* are necessary
- Tomorrow:
  - Consider scientific meaning
  - Severe testing as a basic concept
  - First Symposium on Philosophy, History, and Methodology of Error, June 2006

Comparison



- Does my algorithm A perform better than the best known B?
- Typical procedure:

  Select test problem (instance) P
  Run algorithm A, say $n = 10$ times
  Obtain 10 fitness values: $x_{A,i}$
  Run algorithm B, say $n = 10$ times
  Obtain 10 fitness values: $x_{B,i}$
- Stone age analysis: Compare mean values
- Write paper: 1/2 algorithm, 1/3 results, 1/6 interpretation
- Gain merits

Merits

GECCO

HOME

ADDITIONAL INFORMATION
CALL FOR LATE BREAKING PAPERS
COMMITTEES
COMPETITIONS
DISCOUNTS FOR ISGEC MEMBERS
EVOLUTIONARY COMPUTATION IN INDUSTRY
FREE TUTORIALS PLANNED
GRADUATE STUDENT WORKSHOP
HOTEL AND LOCAL ARRANGEMENTS
KEYNOTE SPEAKERS

BEST PAPER AWARDS FOR GECCO 1997

My algorithm A is better than algorithm B - 100 comparisons

Marcel Mean

My algorithm A is better than algorithm B - 10 comparisons

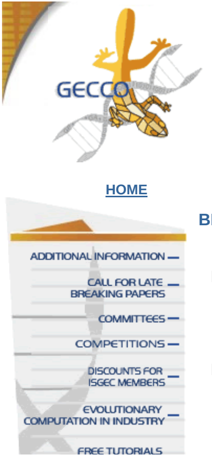
Marco Median

- Best Paper Nominees
- Competition Results
- Human Competitive Award winners
- Best student paper and best student presentation awards

Comparison today

- Does my algorithm A perform better than the best known B?
- Typical procedure:
  - Select several instances of a test problem P_k
  - Run algorithm A, say $n = 10$ times on every P_k
  - Obtain 10 fitness values: $x_{A,k,i}$
  - Run algorithm B, say $n = 10$ times on every P_k
  - Obtain 10 fitness values: $x_{B,i,k}$
- Analysis today comparable to other sciences: Detailed statistical analysis, hypotheses testing
- Write paper: 1/3 algorithm, 1/3 results, 1/3 statistical analysis
- Gain merits

Merits



BEST PAPER AWARDS FOR GECCO 2007

My algorithm A is better than algorithm B
- significant results: p<0.01

Pierre Values

My algorithm A is better than algorithm B
- significant results: p<0.1

Stan Significant

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 difference meaningful?
- Standard statistic: *p*-value
- Problems related to the *p*-value

High-Quality Statistics

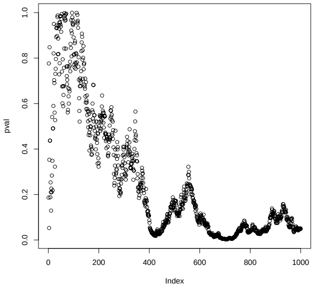
- Fundamental to all comparisons - even to high-level procedures
- The basic procedure reads:

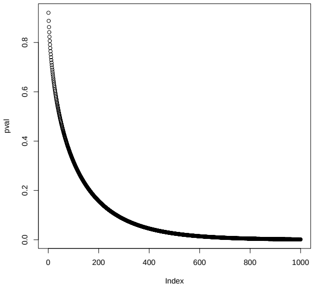
  Select test problem (instance) P
  Run algorithm A, say n times
  Obtain n fitness values: $x_{A,i}$
  Run algorithm B, say n times
  Obtain n fitness values: $x_{B,i}$

R-demo

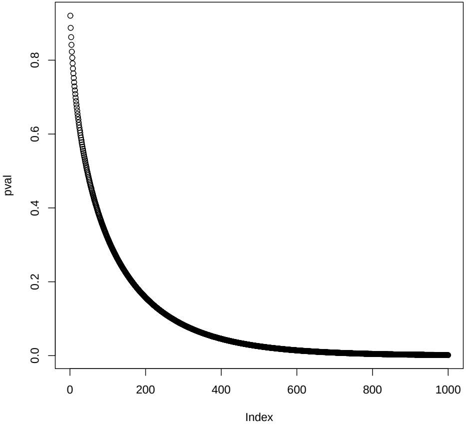
- ```
  > n=100
  > run.algorithm1(n)
    [1]  99.53952  99.86982 101.65871...
  > run.algorithm2(n)
    [1]  99.43952  99.76982 101.55871...
  ```
- Now we have generated a plethora of important data - what is the next step?
- Select a test (statistic), e.g., the mean
- Set up a hypothesis, e.g., there 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: $d <= 0$ 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.
  ```

Large *n* problem





Scientific?



Seeking for Objectivity

- [MS06] propose two conditions for a severe test:

  A statistical hypothesis H passed a severe test T with data x_0 *if*

  (S-1) x_0 *agrees with H, and*

  (S-2) with very high probability, test T would have produced a result that accords less well with H than x_0 *does, if H were false*

- They define a severity function which has three arguments: a test, an outcome, and an inference or claim:

  $$\text{SEV}(\text{Test } T, \text{ outcome } x, \text{ claim } H).$$

- For reasons of simplicity, we will use $\text{SEV}(\mu; \mu_1)$

Severity interpretation of acceptance

- Evaluating severity of test $T(\alpha)$ in the case of a statistically insignificant result (H_0 is accepted):

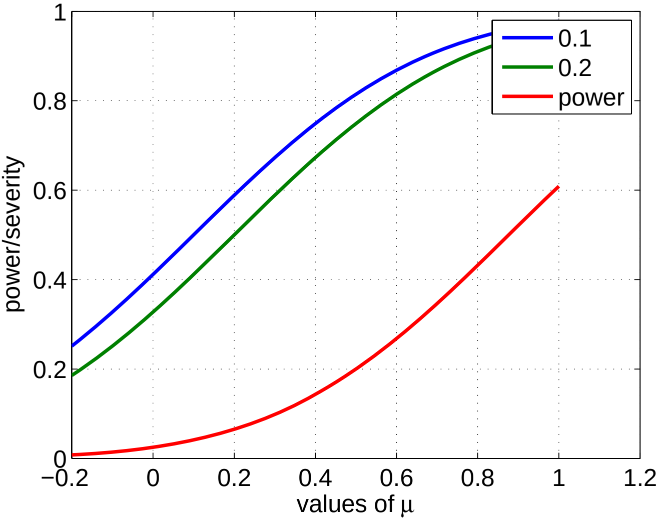
$$\text{SEV}(\mu \leq \mu_1) = P(d(X) > d(x_0); \mu \leq \mu_1 \text{ false }) = P(d(X) > d(x_0); \mu > \mu_1)$$

- Setting severity and power in contrast with each other:

$$\text{POW}(T(\alpha); \mu_1) = P(d(X) > c_\alpha; \mu = \mu_1),$$
$$\text{SEV}(T(\alpha); \mu_1) = P(d(X) > d(x_0); \mu = \mu_1).$$

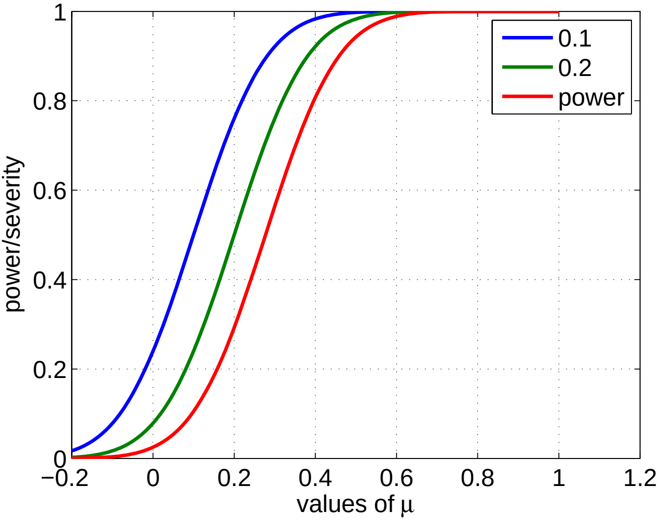
- Severe testing as a concept for post-data inference

Severity ($n = 5$)



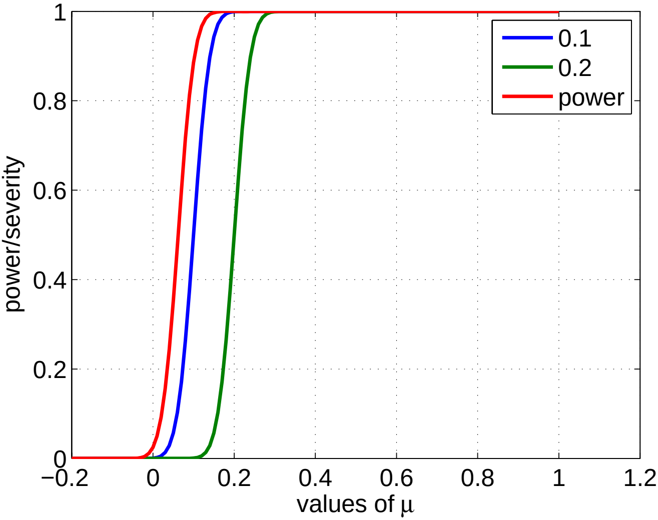
- $n = 5$

Severity ($n = 50$)



- $n = 50$

Severity ($n = 1000$)



- $n = 1000$

Summary

- p value is not sufficient
- Tools to derive meta-statistical rules
- Other tools needed

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

  What exactly is the algorithm good for?

[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

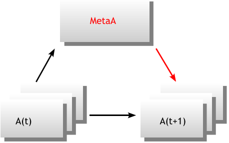
Interesting, previously neglected aspects:

- 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.)

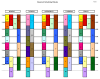
Similarities and Differences to Existing Approaches

- Agriculture, industry: Design of Experiments (DoE)

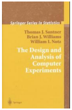

- Evolutionary algorithms: Meta-algorithms

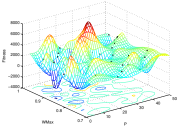


- Algorithm engineering: Rosenberg Study (ANOVA)



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





Empirical Analysis: Algorithms for Scheduling Problems

1

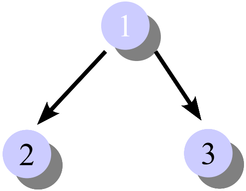
- Problem:
  - Jobs build binary tree
  - Parallel computer with ring topology
- 2 algorithms:

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

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

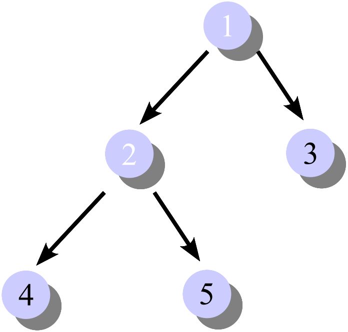
Empirical Analysis: Algorithms for Scheduling Problems

- 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?

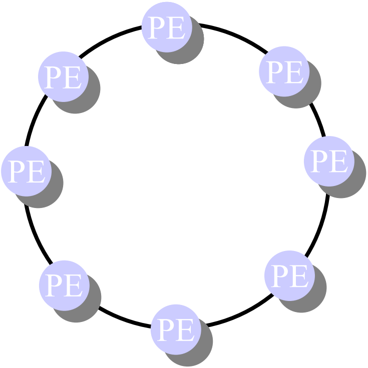


Empirical Analysis: Algorithms for Scheduling Problems

- 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

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

Algorithm Designs

Example (Algorithm design)

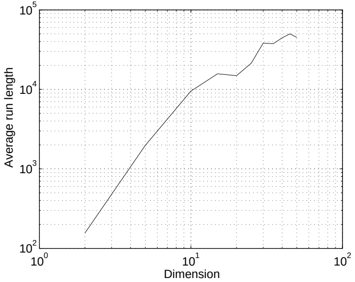
Particle swarm optimization. Set of exogenous strategy parameters

- Swarm size s
- Cognitive parameter c_1
- Social parameter c_2
- Starting value of the inertia weight w_{max}
- Final value of the inertia weight w_{scale}
- Percentage of iterations for which w_{max} is reduced
- Maximum value of the step size v_{max}

Problem Designs

Example (Problem design)

Sphere function $\sum_{i=1}^{d} x_i^2$ and a set of d-dimensional starting points, performance measure, termination criterion



- Tuning (efficiency):
  - Given one problem instance $\Rightarrow$ determine improved algorithm parameters
- Robustness (effectivity):
  - Given one algorithm $\Rightarrow$ test several problem instances

□

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)

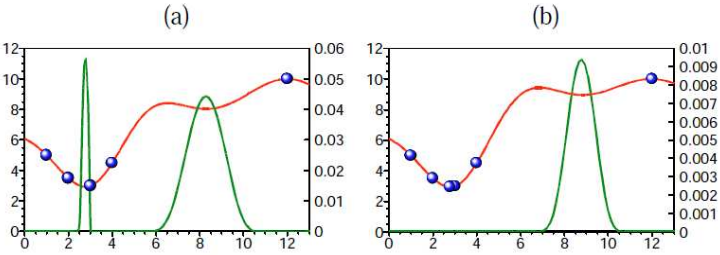


Figure: Axis labels left: function value, right: expected improvement. Source: [JSW98]

(a) Expected improvement: 5 sample points

(b) Another sample point $x = 2.8$ was added

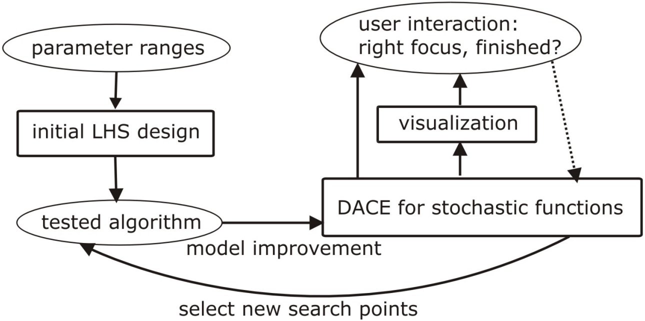
Heuristic for Stochastically Disturbed Function Values

- 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 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_{scale}	w_{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

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

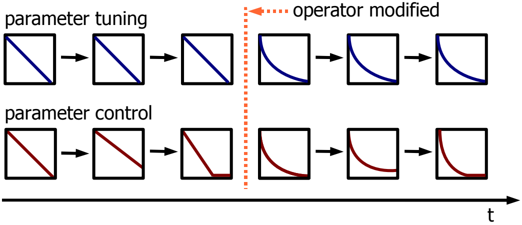
$\Rightarrow$ We can reduce number of parameters, but usually at the cost of either performance or robustness

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

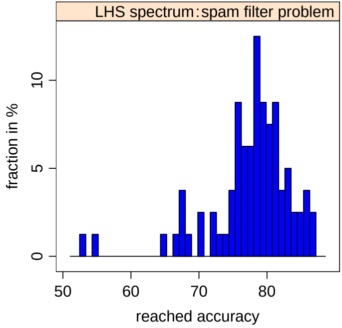


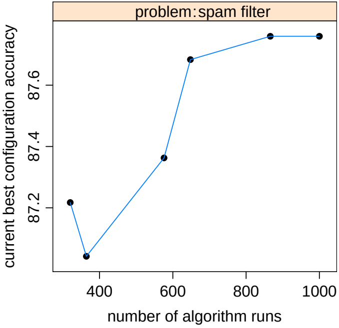
And: How to find meta-parameter values for parameter control?
⇒ Parameter control *and* parameter tuning

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:

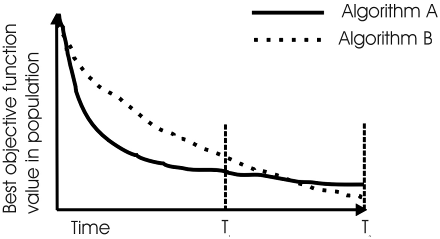
- How much amount to put into tuning (fixed budget, until stagnation)?
- Where shall we be on the spectrum when we compare?
- Can we compare spectra ($\Rightarrow$ adaptability)?

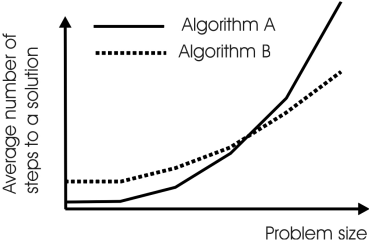
"Traditional" Measuring in EC

Simple Measures

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

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





(figures provided by Gusz Eiben)

Aggregated Measures

Especially Useful for Restart Strategies

Success Performances:

- SP1 [HK04] for equal expected lengths of successful and unsuccessful runs $\mathbb{E}(T^s) = \mathbb{E}(T^{us})$:
$$SP1 = \frac{\mathbb{E}(T_A^s)}{p_s} \tag{1}$$
- SP2 [AH05] 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

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

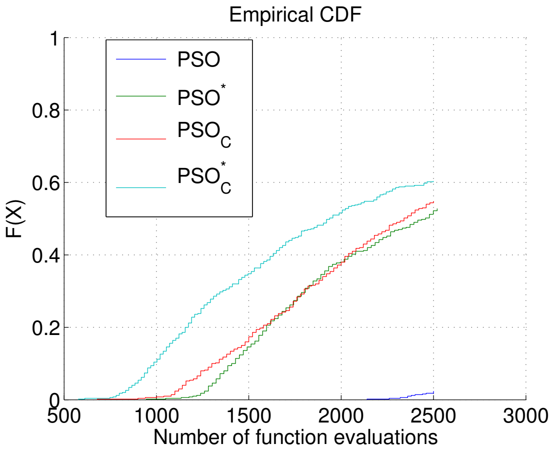
Suggested Report Structure

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 12-step SPO experiments

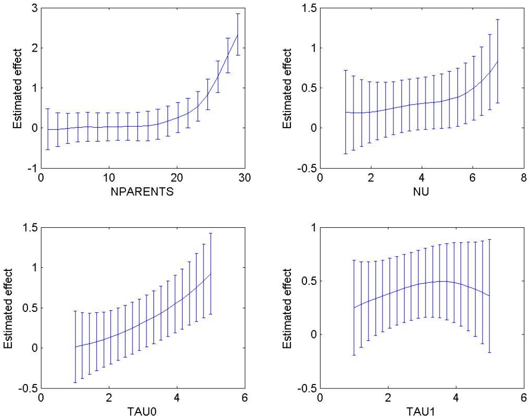
Objective Interpretation of the Results

Comparison. Run-length distribution



(Single) Effect Plots

Useful, but not Perfect

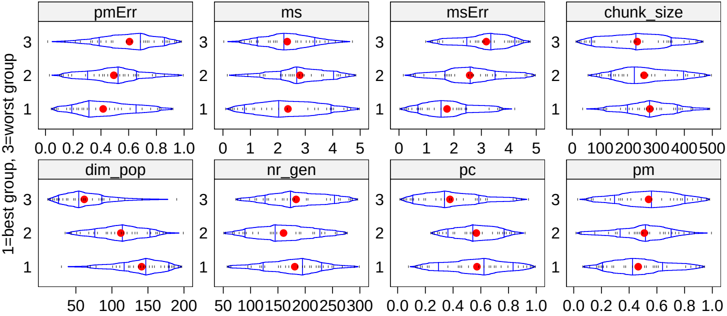


- Large variances originate from averaging
- The τ_0 and especially τ_1 plots show different behavior on extreme values (see error bars), probably distinct (averaged) effects/interactions

One-Parameter Effect Investigation

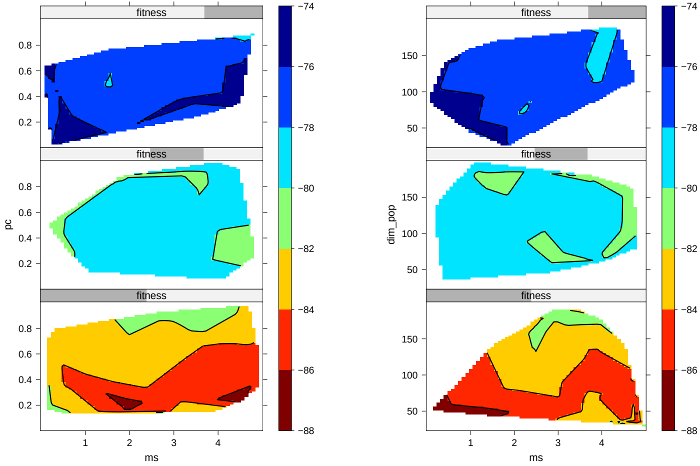
Effect Split Plots: Effect Strengths

- 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



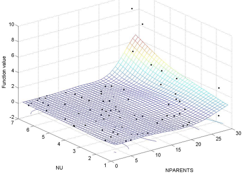
Two-Parameter Effect Investigation

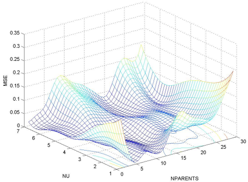
Interaction Split Plots: Detect Leveled Effects

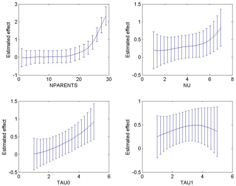


SPO in Action

- 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)

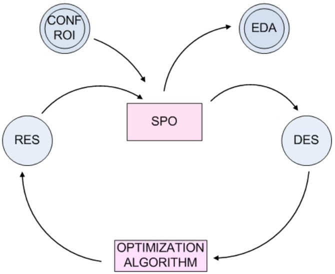
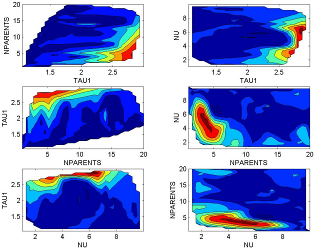


Figure: SPO Interfaces

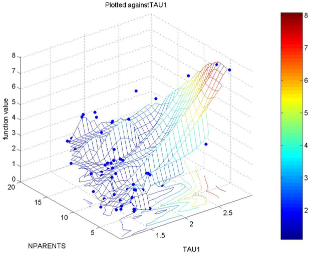
SPO live demo

SPO and EDA

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

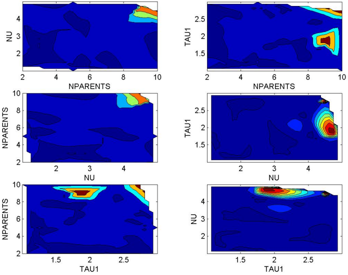


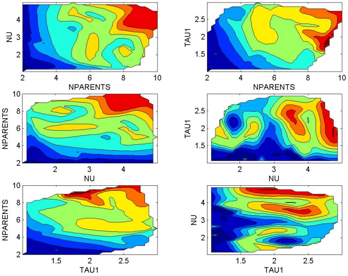
- Box plots
- Trellis plots
- Design plots
- ...



How to Perform an Experimental Analysis

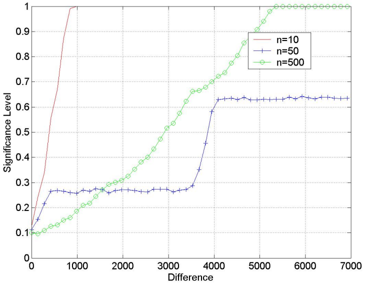
- Scientific claim: “ES with small populations perform better than ES with larger ones on the sphere.”
- Statistical hypotheses:
    - ES with, say $\mu = 2$, performs better than ES with *mu* > 2 if compared on problem design ${}_p^{(1)}$
    - ES with, say $\mu = 2$, performs better than ES with *mu*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





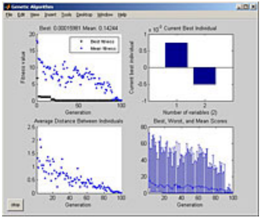
SPO Open Questions

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



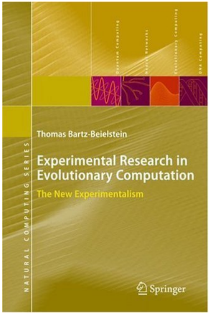
SPOT Community

- Provide SPOT interfaces for important optimization algorithms
- Simple and open specification
- Currently available for the following products:



Program	Language	
Evolution Strategy	JAVA, MATLAB	`http://www.springer.com/3-540-32026-1`
Genetic Algorithm and Direct Search Toolbox	MATLAB	`http://www.mathworks.com/products/gads`
Particle Swarm Optimization Toolbox	MATLAB	`http://psotoolbox.sourceforge.net`

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.

Kit Yan Chan, Emin Aydin, and Terry Fogarty.
An empirical study on the performance of factorial design based crossover on parametrical problems.
In *Proceedings of the 2004 IEEE Congress on Evolutionary Computation*, pages 620–627, Portland, Oregon, 20-23 June 2004. IEEE Press.

David E. Goldberg.
Genetic Algorithms in Search, Optimization, and Machine Learning.
Addison-Wesley, Reading MA, 1989.

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.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.

D. G. Mayo and A. Spanos.
Severe testing as a basic concept in a neyman–pearson philosophy of induction.
British Journal for the Philosophy of Science, 57:323–357, 2006.

T. J. Santner, B. J. Williams, and W. I. Notz.
The Design and Analysis of Computer Experiments.
Springer, Berlin, Heidelberg, New York, 2003.