Design of Experiments and Sequential Parameter Optimization

Thomas Bartz-Beielstein

Cologne University of Applied Sciences

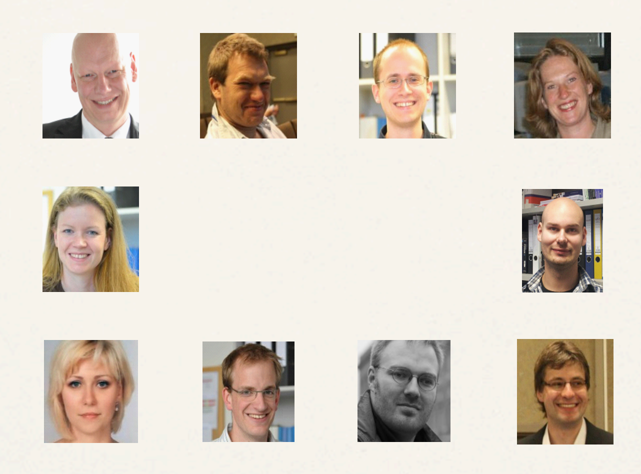
spotseven.org

3rd Brazilian-German Frontiers of Science and Technology Symposium

November 8-12, 2012, Brasilia, Brazil

Agenda

The SPOTSeven Team



- www.spotseven.org

What is this Thing called Science?

- Astrology versus Astronomy

Algorithm Invention and Introduction

- How do we generate scientific results?

How to Publish Paper in Evolutionary Computing

- An extraordinary number of algorithms exist in Evolutionary Computation
- Establishing new algorithms is highly competitive
- Paper, which introduces the new operators, is published
- Benchmarks
- Appropriate parameter settings must be determined

Scientific Ingredient 1: Benchmarking—General Rules

- Validity
- Reproducibility
- Comparability
- Commons rules:
  - Use statistics
  - Documentation
  - Comparisons
- On-going discussion

Computational Intelligence: State-of-the-Art Methoden und Benchmarkprobleme

Frank Hoffmann[1], Ralf Mikut[2], Andreas Kroll[3], Markus Reischl[2], Oliver Nelles[4], Horst Schulte[5], Torsten Bertram[1]

[1]Technische Universität Dortmund, Lehrstuhl für Regelungssystemtechnik
E-Mail: {frank.hoffmann}{torsten.bertram}@tu-dortmund.de
[2]Karlsruher Institut für Technologie, Institut für Angewandte Informatik
E-Mail: {ralf.mikut}{markus.reischl}@kit.edu
[3]Universität Kassel, FB Maschinenbau, FG Mess- und Regelungstechnik
E-Mail: andreas.kroll@mrt.uni-kassel.de
[4]Universität Siegen, Mess- und Regelungstechnik - Mechatronik Department Maschinenbau
E-Mail: oliver.nelles@uni-siegen.de
[5] HTW Berlin, FB Ingenieurwissenschaften I, FG Regelungstechnik und Systemdynamik
E-Mail: schulte@htw-berlin.de

Zusammenfassung: Dieser Beitrag gibt einen Überblick über den Stand der Technik in der Computational Intelligence für Methoden zur Klassifikation, zum Text Mining, zur nichtlinearen Regression, nichtlinearen Systemidentifikation und Regelung. Im Fokus steht eine systematische, wissenschaftlichen Ansprüchen genügende Vorgehensweise bei der vergleichenden Bewertung und Analyse alternativer Ansätze. Die einzelnen Abschnitte geben praktikable Hinweise auf vorhandene, möglichst frei verfügbare Implementierungen, Benchmarkdatensätze und -probleme als Hilfestellung für den Methodenvergleich zukünftiger Publikationen innerhalb des CI-Workshops.

1 Einführung

Die Methodik und Vorgehensweise bei der Bewertung, dem Vergleich und systematischen Analyse neuartiger Methoden der Mustererkennung und Funktionsapproximation hat auf vergangenen Computational Intelligence Workshops zu Kritik und Diskussionen geführt. In einigen Beiträgen fehlte

Scientific Ingredient 2: Features of Test Function Generators

- Difficult to solve using simple methods such as hill climbers
- Nonlinear, non separable, non symmetric
- Scalable with respect to problem dimensionality
- Scalable with respect to evaluation time
- Tunable by a small number of user parameters

See,e.g, [2]

The Current Situation

- Combining the two scientific ingredients
- Authors report parameter values which seem to work reasonably well
- Test problem suite setup: usually a small number of specified test problems. Each algorithm will be run for some number, say ten, on each problem. Statistics are reported, e.g., mean, standard deviation
- What is the problem of this approach?

One Problem (only?)

- Paper does not tell the whole story, because:
- Preliminary trials, which were performed to determine these parameter settings, are performed, but not reported
- Chance of 5% that your new algorithm performs better than the best algorithm
- Run your new (in fact worse) algorithm 100 times
- Report only positive results
- Even if not intended, might happen unconsciously

Ben Michael Goldacre's TED talk

http://embed.ted.com/talks/ben_goldacre_what_doctors_don_t_know_about_the_drugs_they_prescribe.html

Benchmarking—A First Solution?

- One expert compares his new algorithm with establishes approaches. Subjective (unfair?) comparison
- Many experts compare their algorithms on several, standardized data. Objective (fair) comparison
- Use accepted data bases, e.g., UCI
- Divide data into train, validation, and test data
- What is the problem of this approach?

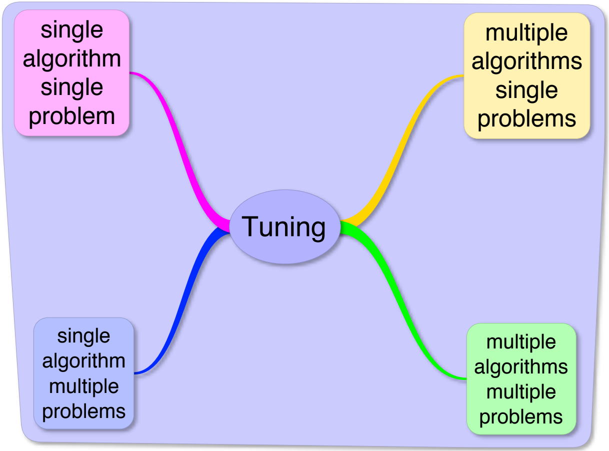
Benchmarking—Only A First Solution

- Algorithms are trained for this specific set of benchmark functions
  - Who defines this set of functions?
- In practice, I do not need an algorithm which performs good on a set of test problems (which was developed by some experts)
- Really wanted:
  - An algorithm, which performs very good on my set of real-word test problems
  - Not only demonstrating
  - Understanding!
- Let's have a short look at the problem

A Taxonomy of Algorithm and Problem Designs

- Classify parameters
- Parameters may be *qualitative*, like for the presence or not of an recombination operater or *numerical*, like for parameters that assume real values
- Our interest: understanding the contribution of these components
- Statistically speaking: parameters are called *factors*
- The interest is in the effects of the specific *levels* chosen for these factors

Problems and Algorithms



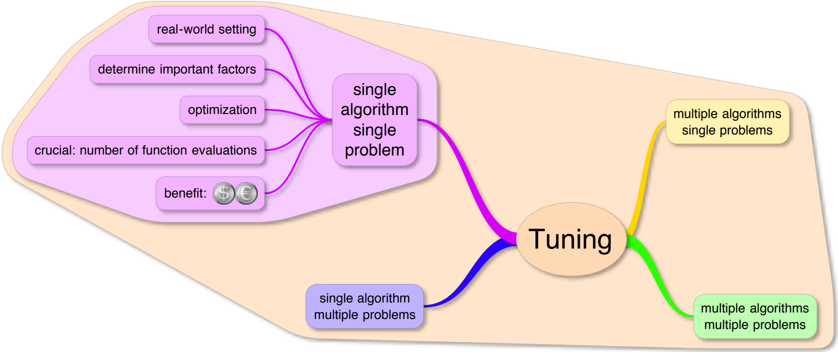
- How to perform comparisons?
- Adequate statistics and models?

SASP: Algorithm and Problem Designs

- Basic design: assess the performance of an *optimization algorithm* on a single problem instance π
- Randomized optimization algorithms $\Rightarrow$ performance Y on one instance is a random variable
- Experiment: On an instance π algorithm is run r times $\Rightarrow$ collect sample data $Y_1, \ldots, Y_r$ (independent, identically distributed)
- One instance π, run the algorithm r times $\Rightarrow$ r replicates of the performance measure Y, denoted by $Y_1, \ldots, Y_r$
- Samples are conditionally on the sampled instance and given the random nature of the algorithm, independent and identically distributed (i.i.d.), i.e.,

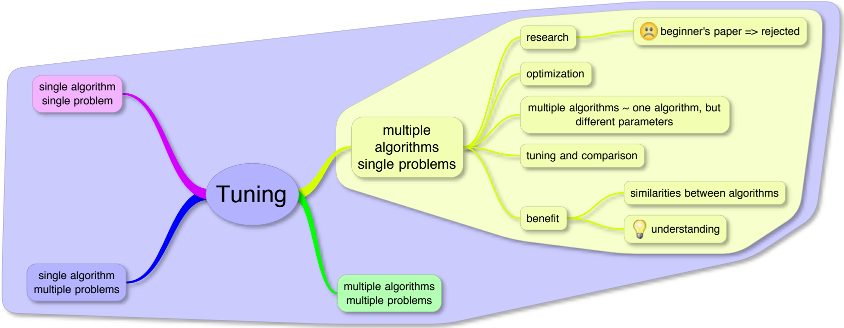
$$p(y_1, \ldots, y_r | \pi) = \prod_{j=1}^{r} p(y_j | \pi). \tag{1}$$

SASP – Single Algorithm, Single Problem



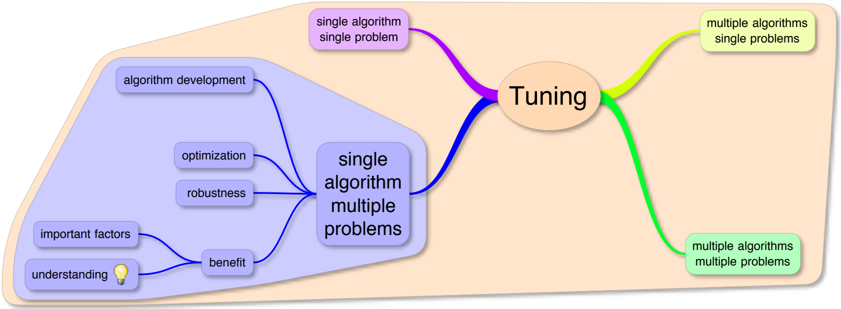
MASP: Algorithm and Problem Designs

- Several optimization algorithms are compared on one fixed problem instance π
- Experiment: collect sample data $Y_1, \ldots, Y_R$ (independent, identically distributed)
- Goal: comparison of algorithms on one (real-world) problem instance π



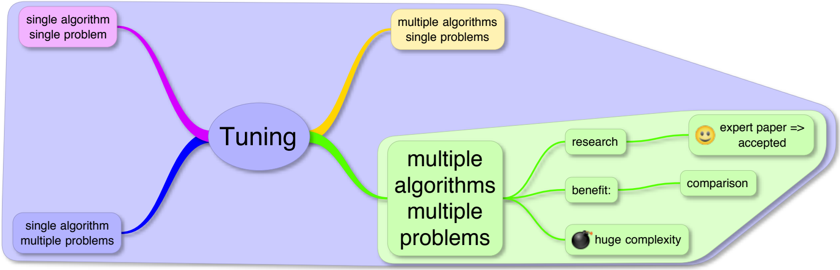
SAMP: Algorithm and Problem Designs

- Goal: Drawing conclusions about a certain *class* or *population* of instances Π



MAMP: Fixed Algorithm and Problem Designs

- Typically:
  - Take a few, fixed instances for the problem at hand
  - Collect the results of some runs of the algorithms on these instances
- Statistically, instances are also *levels of a factor*
- Instances treated as *blocks*
- All algorithms are run on each single instance
- Results are therefore *grouped* per instance



MAMP: Randomized Problem Designs

- Sometimes, several hundred (or even more) problem instances to be tested $\Rightarrow$ interest not just on the performance of the algorithms on a few specific instances, but rather on the generalization of the results to the entire population of instances
- Procedure: instances are chosen at random from a large set of possible instances of the problem
- Statistically, instances are also *levels of a factor*
- However, factor is of a different nature from the fixed algorithmic factors described above
- Levels are chosen at random and the interest is not in these specific levels but in the population from which they are sampled
- $\Rightarrow$ levels and factors are random
- This leads naturally to a mixed model [1]

Comparison of Two Simulated Annealing Parameter Settings

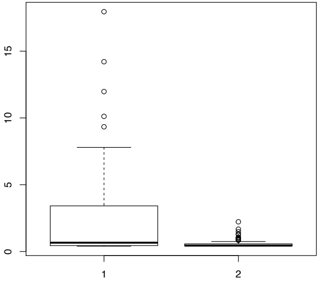
- Fixed-Effects Design: one *algorithm* is evaluated on one *instance* π (fixed), i.e., SASP

```
> set.seed(123)
> library(SPOT)
> fn <- spotBraninFunction #test function to be optimized by SANN
> x0 <- c(-2,3) #starting point that SANN uses when optimizing Branin
> maxit <- 100 #number of evaluations of Branin allowed for SANN
> temp <- 10
> tmax <- 10
> n <- 100
> y <- rep(1,n)
> y0<-sapply(y, function(x) x<-optim(par=x0, fn=fn, method="SANN"
+                                    , control=list(maxit=maxit,
+                                                   temp=temp, tmax=tmax))$value)
> temp <- 4
> tmax <- 62
> y <- rep(1,n)
> y1<-sapply(y, function(x) x<-optim(par=x0, fn=fn, method="SANN"
+                                    , control=list(maxit=maxit,
+                                                   temp=temp, tmax=tmax))$value)
```

Comparison: Simple EDA Using Boxplots

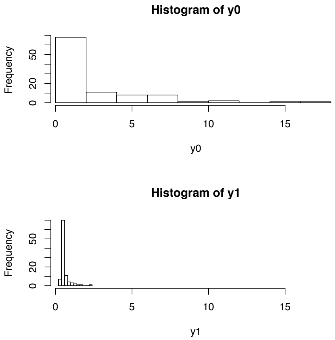
```
> summary(y0)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.3984  0.4444  0.6587  2.2770  3.4020 17.9600
> summary(y1)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.3985  0.4150  0.4439  0.5609  0.5736  2.2250

> boxplot(y0,y1)
```



Comparison: Simple EDA Using Histograms

```
> par(mfrow=c(2,1))
> hist(y0,xlim = c( min(y0,y1), max(y0,y1)))
> hist(y1,xlim = c( min(y0,y1), max(y0,y1)))
> par(mfrow=c(1,1))
```

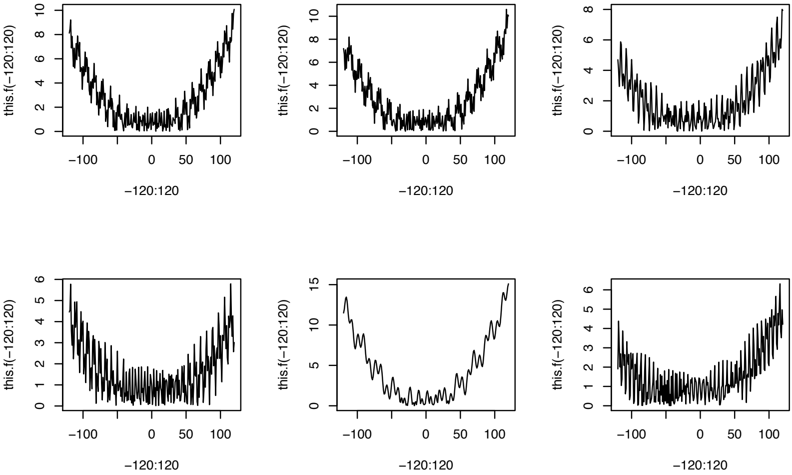


Comparison

- What have we learned from this comparison?
  - Reducing `temp` from 10 to 4 while increasing `tmax` from 10 to 62 improves the performance of simulated annealing
  - We have demonstrated that one setting is better
- Is this interesting for other researcher?
- Can this result be generalized?

Introducing Mixed Models: Problem Instances

- Nine problem instances, which were randomly drawn from an infinite number of instances: `fSeed`



Introducing Mixed Models: Algorithm Settings

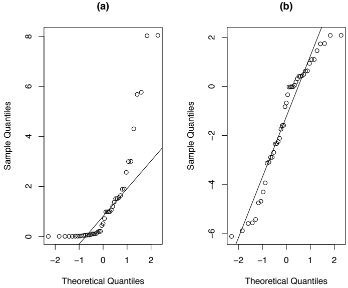
- Evolution Strategy with and without mutation: `mut`
- Five repeats of the ES: `algSeed`

```
> df <- read.table("expBart12a1.csv", header=T)
> df$mut <- factor(df$mut)
> df$fSeed <- factor(df$fSeed)
> df$algSeed <- factor(df$algSeed)
> df<-cbind(df, yLog = log(df$y))
> str(df)
```

```
'data.frame':       90 obs. of  5 variables:
 $ y      : num  6.22 5.4 4.56 6.12 4.65 ...
 $ mut    : Factor w/ 2 levels "1","2": 1 1 1 1 1 2 2 2 2 2 ...
 $ fSeed  : Factor w/ 9 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ algSeed: Factor w/ 5 levels "1","2","3","4",..: 1 2 3 4 5 1 2 3 4 5 ...
 $ yLog   : num  1.83 1.69 1.52 1.81 1.54 ...
```

Introducing Mixed Models: Assumptions

- Test the validity of the model assumptions by generating normal quantile plots (QQ plots)



The Easiness of Mixed Models: The classical ANOVA

Table: ANOVA table for a one-factor fixed and random effects models

Source of Variation	Sum of Squares	Degrees of freedom	Mean Square	EMS Fixed	EMS Random
Treatment	SS_{treat}	$q-1$	MS_{treat}	$\sigma^2 + r\frac{\sum_{i=1}^{q}\tau_i^2}{q-1}$	$\sigma^2 + r\sigma_\tau^2$
Error	SS_{err}	$q(r-1)$	MS_{err}	σ^2	σ^2
Total	SS_{total}	$qr-1$			

```
> obj1 <- aov(yLog ~fSeed, data=df)
> summary(obj1)

            Df Sum Sq Mean Sq F value Pr(>F)
fSeed        8   53.6   6.705   1.412  0.204
Residuals   81  384.6   4.748
```

The Easiness of Mixed Models: Some Statistics

- Technical stuff: how to access information in R

```
> (M1 <- anova(obj1))

Analysis of Variance Table

Response: yLog
          Df Sum Sq Mean Sq F value Pr(>F)
fSeed      8  53.64  6.7049  1.4122  0.204
Residuals 81 384.59  4.7480

> (MSA <- M1[1,3])

[1] 6.704854

> (MSE <- M1[2,3])

[1] 4.74797

> r <-length(unique(df$algSeed))
> q <- nlevels(df$fSeed)
> (var.A <- (MSA - MSE)/(r))

[1] 0.3913767

> (var.E <- MSE)

[1] 4.74797
```

Results: What have we Learned?

- Relatively high variance

```
> var.A + var.E
```

```
[1] 5.139347
```

- Large *p* value indicates that variance is caused by mutation operator, not by problem instances

```
>  1-pf(MSA/MSE,q-1,q*(r-1))
```

```
[1] 0.2249
```

- Confidence interval provide information of the algorithm performance on this set of problem instances (prediction for new problem instances)

```
> s <- sqrt(MSA/(q*r))
> Y.. <- mean(df$yLog)
> qsr <- qt(1-0.025,r)
> c( exp(Y.. - qsr * s), exp(Y.. + qsr * s))
```

```
[1] 0.3529964 2.5681782
```

Results: What have we Learned?

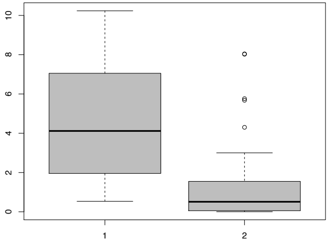
- Mutation improves Algorithm performance (smaller values are better)

```
> summary(df$y[df$mut==1])

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.5352  1.9560  4.1170  4.6380  7.0540 10.2300

> summary(df$y[df$mut==2])

    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
0.002223 0.055440 0.511000 1.354000 1.550000 8.042000
```

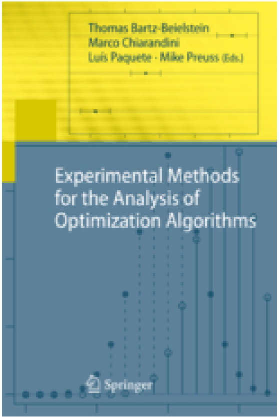


Outlook

JOURNAL OF NEGATIVE RESULTS IN BIOMEDICINE

- Journal of Negative Results in BioMedicine strongly promotes and invites the publication of clinical trials that fall short of demonstrating an improvement over current treatments
- The aim of the journal is to provide scientists and physicians with responsible and balanced information in order to improve experimental designs and clinical decisions
- Do we need a similar infrastructure in computer science?
  - Register and announce experiments
  - Login
  - Download test data
  - Perform experiments
  - Report
- Only a phantasy? Let's start a discussion...

Suggested Reading



- http://www.spotseven.org

- Experimental Methods for the Analysis of Optimization Algorithms
- See also Kleijnen [3], Saltelli et al.

Acknowledgments

- This work has been supported by the Federal Ministry of Education and Research (BMBF) under the grants MCIOP (FKZ 17N0311) and CIMO (FKZ 17002X11)

[1] Marco Chiarandini and Yuri Goegebeur.
Mixed models for the analysis of optimization algorithms.
In Thomas Bartz-Beielstein, Marco Chiarandini, Luís Paquete, and Mike Preuss, editors, *Experimental Methods for the Analysis of Optimization Algorithms*, pages 225–264. Springer, Germany, 2010.
Preliminary version available as *Tech. Rep.* DMF-2009-07-001 at the The Danish Mathematical Society.

[2] M. Gallagher and B. Yuan.
A general-purpose tunable landscape generator.
IEEE transactions on evolutionary computation, 10(5):590–603, 2006.

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