Research Topics in Sequential Parameter Optimization

T. Bartz-Beielstein

Fachhochschule Köln

May, 10th 2012

Table of Contents

Optimization via Simulation

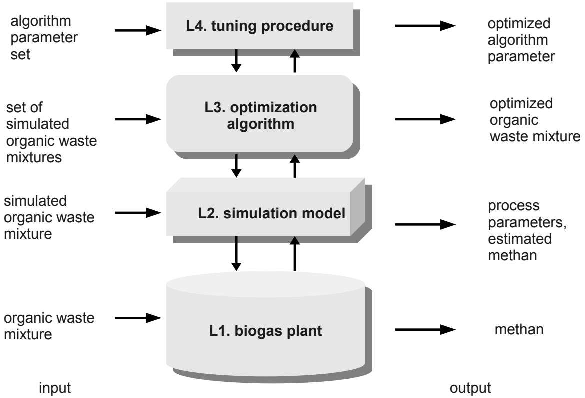
(L1) The real-world system, e.g., a biogas plant

(L2) The related simulation model. The objective function f is defined at this layer. In optimization via simulation, problem parameters are defined at this layer

(L3) The optimization algorithm A. It requires the specification of algorithm parameters, say $\vec{p}^i \in \vec{P}$, where $\vec{P}$ denotes the set of parameter vectors

(L4) The experiments and the tuning procedure

Optimization via Simulation



Sequential Parameter Optimization

- Sequential approach: most powerful optimization technique
- SPOT package can be downloaded from the comprehensive R archive network at `http://CRAN.R-project.org/package=SPOT`
- SPOT is one possible implementation of the *sequential parameter optimization* (SPO) framework introduced in [1]
  - Several implementations developed world wide (SPOT+)
- For a detailed documentation of the functions from the SPOT package, the reader is referred to the package help manuals

Initial Problem Statement

- Developmented started in 2001 at TU Dortmund
- Performance of modern search heuristics such as *evolution strategies* (ES), *differential evolution* (DE), or *simulated annealing* (SANN) relies crucially on their parameterizations—or, statistically speaking, on their factor settings
- *Algorithm design*: factors that influence the behavior (performance) of an algorithm, e.g., population size in ES
- *Problem design*: factors from the optimization (simulation) problem, search space dimension

Tuning

- Interesting goal of SPO: to detect the importance of certain parts (subroutines such as recombination in ES) by systematically varying the factor settings of the algorithm design
- This goal related to improving the algorithm's *efficiency*
- Will be referred to in the following as *algorithm tuning*
- The experimenter is seeking for an improved parameter setting for one problem instance

Robustness

- Varying problem instances, e.g., search space dimensions or starting points of the algorithm, are associated with *effectivity* or the algorithm's robustness
- Experimenter is interested in one parameter setting of the algorithm with which the algorithm performs sufficiently good on several problem instances
- GECCO Tutorial provides an overview, preprint can be downloaded from `http://www.gm.fh-koeln.de/~bartz/Papers.d/Bart12f.pdf`

SPO and DoE

- SPO may lead to a better understanding of the algorithm
- SPO combines several techniques from classical and modern statistics, namely *design of experiments* (DoE) and *design and analysis of computer experiments* (DACE) [1]
- Basic ideas from SPO rely heavily on Kleijnen's work on statistical techniques in simulation [7, 8]
- Book "Experimental methods for the analysis of optimization algorithms" `http://www.springer.com/978-3-642-02537-2` devoted to recent developments in the field of experimental research [3]

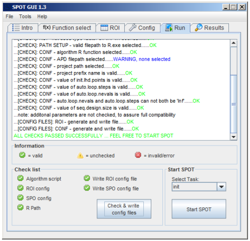
Sequential Parameter Optimization SPO

Use statistical techniques and methods from design of experiment to solve optimization problems.

1. Take initial samples from design space and evaluate on target function/algorithm
2. Build surrogate model (Linear, Tree-based, Kriging, ...) based on known evaluations
3. Determine promising new solutions with model
4. Evaluate new solutions
5. If termination criterion not reached: go to **2.**
6. Summarize Results / Create Report

SPO Toolbox (SPOT)

- Currently maintained and developed as an R-Package
- Interfaces to several other R-packages
- Provides Demos and Documentation
- Graphical User Interface
- Alternative version is available for matlab



SPOT: Installation, Help, Demos

- Install from CRAN:

```
> install.packages("SPOT")
```

- Load package to Workspace:

```
> require("SPOT")
```

- Get help on some spot functions

```
> ?spot
> ?spotOptim
```

- Get a list of SPOT demos

```
> demo(package="SPOT")
```

- Run a SPOT demo

```
> demo("spotDemo18ForresterOptim",ask=F)
```

- Start the GUI

```
> spotGui()
```

Applications: Algorithms Tuned by SPOT

- Several types of evolution strategies
- Time series prediction and anomaly detection
- Classification
- Symbolic Regression
- Simulated Annealing
- For more applications see [2]

Simulated Annealing SANN

- Randomized optimization algorithm
- Two parameters: 1) starting temperature TEMP 2) number of function evaluations at each temperature TMAX

```
> fn <- function (x) {
+  x1 <- x[1] ;  x2 <- x[2]
+  y <- (x2 - 5.1/(4 * pi^2) * (x1^2) + 5/pi * x1 - 6)^2 +
+   10 * (1 - 1/(8 * pi)) * cos(x1) + 10
+  return(y)
+ }
> temp=10; tmax=10
> result<-optim(par=c(-2,3),fn,method="SANN", control = list(temp=temp, tmax=tm
> result$value
```

```
[1] 0.3981833
```

```
> result$par
```

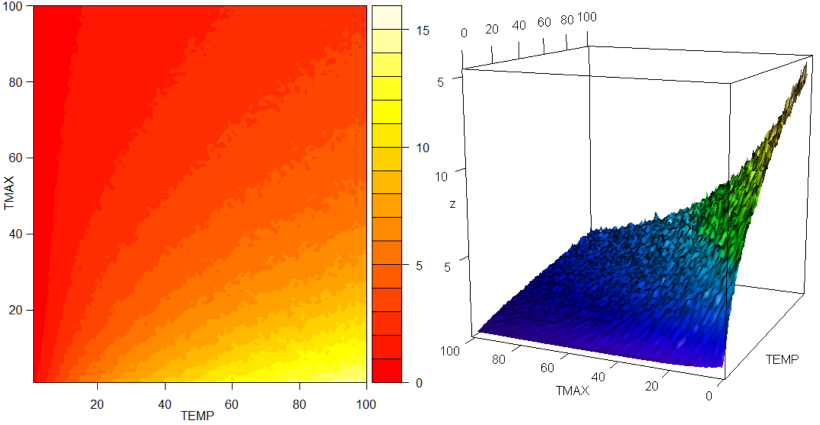
```
[1] -3.144066 12.264619
```

SANN Sweep

- Since this is a simple test problem: Complete sweep
- Understand underlying fitness shape
- 1000 repeats for each setting (takes rather long)

```
> target <- function(x,y,x0,fn,maxit){
+  zz<-matrix(0,length(x))
+  repeats=1000
+  for(i in 1:repeats){
+  set.seed(i)
+  zz =zz + apply(cbind(x,y),1,testalgorithm,x0=x0,fn=fn,maxit=maxit)
+  }
+  return(zz/repeats)
+ }
> x <- seq(1, 100, length.out = 100)
> y <- x
> z <- outer(x, y, target,x0=x0,fn=fn,maxit=maxit)
> filled.contour(x, y, z, color.palette=heat.colors,xlab="temp",ylab="tmax")
> pal <- topo.colors(100)
> require(rgl)
> persp3d(x,y,z,col=pal[cut(z,100)],xlab="TEMP",ylab="TMAX")
```

Plots from Sweep



Tuning SANN in the SPOT Framework: Problem Definition

- Target function: Branin-Function (2-D function with three global minima)

```
> require(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
> testalgorithm <- function(pars,x0,fn,maxit){
+  temp<-pars[1]
+  tmax<-pars[2]
+  y <- optim(x0, fn, method="SANN",
+  control=list(maxit=maxit,
+  temp=temp, tmax=tmax))
+  return(y$value)
+ }
```

Tuning SANN: Configure SPOT

- ROI: Region of interest, in which parameters are tuned
- Surrogate: Kriging based on Forrester et. al. [6]
- Settings are minimalistic (uses a lot of default values)

```
> roi<-spotROI(c(1,1),c(100,100),type=c("INT","INT"))
> config<-list(alg.func=testalgorithm,
+  alg.roi=roi,
+  init.design.size=20,
+  seq.predictionModel.func="spotPredictForrester",
+  seq.predictionOpt.func="spotPredictOptMulti",
+  seq.predictionOpt.method="cmaes",
+  seq.predictionOpt.budget=1000,
+  report.func="spotReportSens",
+  spot.fileMode=T,
+  io.verbosity=3,
+  auto.loop.nevals=100)
```

Tuning SANN: Run SPOT Demo

- Pass configuration to SPOT
- Pass additional parameters to SPOT, needed by target function

```
> # Warning: run takes several minutes
> # (execute from the script sannDemo2012Rome.R)
> # setwd("C:/Users/bartz/Documents/workspace/SvnDdmo.d/trunk/DdmoSlides.d
> res<-spot(spotConfig=config,x0=x0,fn=fn,maxit=maxit)
```

Tuning SANN: Run SPOT

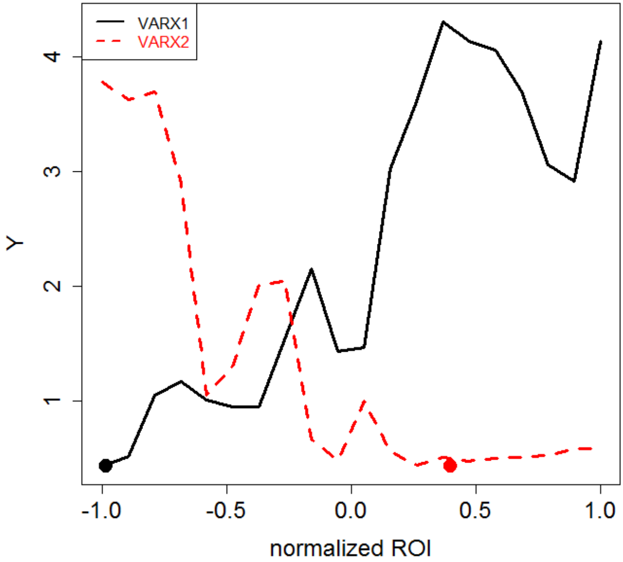
- Output from the following R code:

```
> res<-spot(spotConfig=config,x0=x0,fn=fn,maxit=maxit)

Sensitivity plot for this ROI:
      lower upper type     BEST
VARX1     1   100  INT  1.84744
VARX2     1   100  INT 70.36899

Best solution found with 103 evaluations:
           Y   VARX1    VARX2 COUNT CONFIG
245 0.409769 1.84744 70.36899     5     24

Standard deviation of best solution:
0.409769033349987 +- 0.0112398099327513
```


VARX1
VARX2
Y
4
3
2
1
-1.0
-0.5
0.0
0.5
1.0
normalized ROI

Tuning SANN: Raw Results

- Result file, logged information separated by space

```
Function XDIM YDIM STEP SEED CONFIG VARX1 VARX2 Y
UserSuppliedFunction 2 1 0 1234 1 23 62 0.518342556082896
UserSuppliedFunction 2 1 0 1235 1 23 62 0.402079045134601
UserSuppliedFunction 2 1 0 1234 2 62 33 3.50021485407806
```

- Results in R command line

```
str(res$alg.currentResult)
```

```
'data.frame':   103 obs. of  9 variables:
 $ Function: Factor w/ 1 level "UserSuppliedFunction": 1 1 1
 $ XDIM    : num  2 2 2 2 2 2 2 2 2 2 ...
 $ YDIM    : int  1 1 1 1 1 1 1 1 1 1 ...
 $ STEP    : int  0 0 0 0 0 0 0 0 0 0 ...
 $ SEED    : num  1234 1235 1234 1235 1234 ...
 $ CONFIG  : int  1 1 2 2 3 3 4 4 5 5 ...
 $ VARX1   : num  23 23 62 62 38 38 70 70 8 8 ...
 $ VARX2   : num  62 62 33 33 26 26 16 16 90 90 ...
 $ Y       : num  0.518 0.402 3.5 16.653 4.774 ...
```

Tuning SANN: Other Report Functions

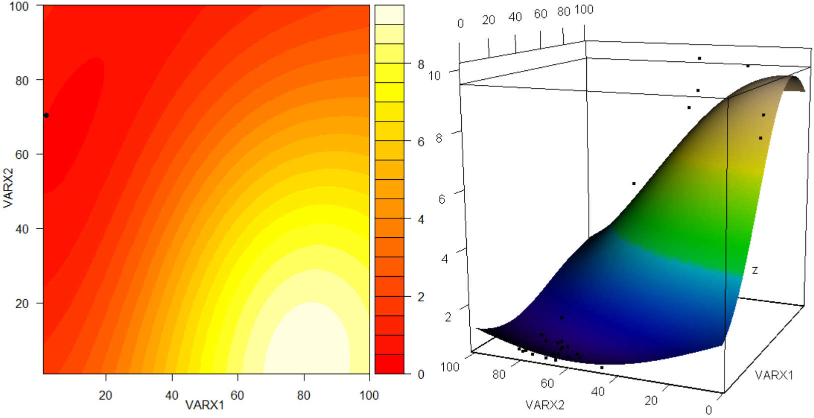
- Other reports/graphics can be created
- spotReportContour for a contour plot

```
> spot(spotConfig=append(list(
+  report.func="spotReportContour",
+  report.interactive=F),
+  res),
+  spotTask="rep")
```

- spotReport3d for 3d plot

```
> spot(spotConfig=append(list(
+  report.func="spotReport3d",
+  report.interactive=F),
+  res),
+  spotTask="rep")
```

Plots from SPOT



SPOT: Existing Features

- Single and multi criteria optimization
- Automated tuning, or manual steps
- Modular concept: Use different combinations of models / methods
- Available surrogate models: Linear, Tree, Kriging, Support Vector Machine, Random Forest, ...
- Tuning real valued parameters as well as factors (i.e. with tree-based models)
- User can use custom models
- Different means of budget allocation
- Logging and Report generation

Research Topic: Noise Handling

- SPOT provides tools for improving the confidence during the search. First approaches increase the number of repeats. An early SPOT implementation proceeded as follows [4]:

  > *At each step, two new designs are generated and the best is re-evaluated. This is similar to the selection procedure in $(1+2)$-Evolution Strategies. The number of repeat runs, k, of the algorithm designs is increased (doubled), if a design has performed best twice or more. A starting value of $k = 2$ was chosen*

- This simple approach did not use any information about the variance

Selection under Noise: Approaches

- Simple: use an identical number of replications for each design
  - Inefficient, because some designs might have low variance, others high variance
- Efficient: Probability of correct selection (PCS)
  - Definition: Correct selection can be defined as *correctly selecting the true best design*
- Problem variations: Subset selection
  - subset which contains the best
  - subset which contains the m best
  - ...

Selection under Noise: Approaches

- State of the art:
  - allocate a larger portion of the computing budget to design that are critical to the process of identifying the best design
  - Use both: (1) sample means and (2) variances
- Assumptions and Derivation of the OCBA equations (see Appendix)

Research Topic: Noise Handling Using OCBA

- Lasarczyk was the first who combined SPOT and OCBA [9]
- OCBA was developed to ensure a high *probability of correct selection* (PCS)
- To maximize PCS, a larger portion of the available budget is allocated to those designs that are critical to the process of identifying the best candidates
- OCBA uses sample means and variances in the budget allocation procedure in order to maximize PCS

[OCBA's Central Idea]

- A number of simulation replications, say T
- Allocated to m competing design points with means $\overline{Y}_1, \overline{Y}_2, \ldots, \overline{Y}_m$ and finite variances $\sigma_1^2, \sigma_2^2, \ldots, \sigma_m^2$, respectively
- N_i is the number of replications allocated to design i
- $\delta_{b,j} = \overline{Y}_b - \overline{Y}_i$ denotes the difference of the i-th and b-th mean with $\overline{Y}_b \leq \min_{i \neq b} \overline{Y}_i$.

[OCBA's Central Idea]

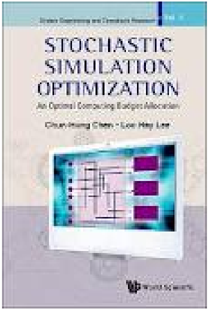
- *Approximate Probability of Correct Selection* asymptotically maximized when

$$\frac{N_i}{N_j} = \left(\frac{\sigma_i/\delta_{b,i}}{\sigma_j/\delta_{b,j}}\right)^2, \quad i,j \in \{1,2,\ldots,m\}, \text{ and } i \neq j \neq b,$$

$$N_b = \sigma_b \sqrt{\sum_{i=1, i\neq b} \frac{N_i^2}{\sigma_i^2}}, \tag{1}$$

OCBA Conclusion

- As can be seen from (1), the allocated computing budget is proportional to variance and inversely proportional to the difference from the best design



SPOT + OCBA: Development

- The OCBA implementation in our study is based on Lasarczyk's work [9]
- New design points which were proposed by the meta model are evaluated several times, e.g., twice [1]
- During each SPOT step, a certain budget (here: `spot.ocba` = 3 is allocated to the candidate solutions to ensure a high PCS for the best design point
- Chen and Lee present a comprehensive coverage of the OCBA methodology [5]

[1] This value can be modified using the `init.design.repeats` variable in SPOT's config file

The OCBA Interface in SPOT

- Spreads the budget in an optimal way for the different design points, considering a minimization problem
- `spotOcba <- function(samp.mean, samp.var, samp.count, budget.add, iz=NA, verbose=0)`
- param
  - samp.mean vector of mean values, length nd
  - samp.var vector of variances, length nd
  - samp.count vector of repeats performed already, length nd
  - budget.add additional number of repeats, distributed among the nd design points
  - iz indifference zone
  - verbose verbosity, 0 is no printing

An OCBA Example in SPOT

```
> samp.mean <- numeric();  samp.var <-  numeric();
> samp.count <- rep(20,10); budget <- 100
> for (m in 1:10) {
+  sample <- rnorm(20, mean=m, sd=6);
+  samp.mean <- c(samp.mean,mean(sample));
+  samp.var <- c(samp.var,var(sample));
+ }
> spotOcba(samp.mean, samp.var, samp.count, budget)
```

```
 [1] 60 40  0  0  0  0  0  0  0  0
```

Current Research

- Extend report functions
- Implementation of ensembles of surrogate models
- Interaction between simulation and surrogate models
- Improve multi criteria optimization
- Adaptive ROI
- New test problems or applications
- Noise handling
- ..., see `www.spotseven.de`
- Discussion ⇒ Cooperation?
  - R source code from the slides

Selection under Noise: Problem Formulation

- Goal: $\min_\theta J(\theta)$, where θ p dimensional vector of (decision) variables
- Real-world application: $J(\theta)$ cannot be determined directly
- $J(\theta) = E[L(\theta, \omega)]$, ω comprises uncertainty
- $L(\theta, \omega)$ available via simulation
- Optimization goal (search):

$$\min \overline{J}(\theta) = \frac{1}{N} \sum_{j=1}^{N} L(\theta, \omega)$$

Critical Designs and OCBA

- General: First step: generate n_0 simulation replications
- Notation summary
    - J_i: true mean of design i
    - σ_i^2: variance of design i
    - $L(\Theta_i, \omega_{ij})$: performance estimate obtained from the output of the jth simulation replication for design i
    - N_i: number of repeats for design i
    - $\overline{J}_i(\theta)$: sample mean of design i, i.e., $(\frac{1}{N_i}\sum_{j=1}^{N_i} L(\theta_i, \omega_{ij})$
    - t: true best design, i.e., $t = \text{argmin}_i J_i$
    - $\delta_{t,i}$: distance ith and best, i.e., $\delta_{t,i} = J_t - J_i$
    - $\sigma_{t,i}^2$: sum of variances, i.e., $\frac{\sigma_t^2}{N_t} + \frac{\sigma_i^2}{N_i}$

OCBA Assumptions

- Simulation output is independent identical distributed and has normal distribution with mean J_i and variance σ_i^2
- $L(\theta_i, \omega_{ij}) \sim N(J_i, \sigma_i^2)$ with known J_i, σ_i^2 and t
- OCBA goal: Find a budget allocation that maximizes PCS, where CS is defined as picking the best design based on sample means from simulation output
- PCS defined as the probability that the true best design has the smallest sample mean and therefore can be selected correctly:

$$PCS = P\{CS\} = P\{\bar{J}_t < \bar{J}_i, i \neq t\}$$

OCBA Assumptions

- Considering normality assumptions $\Rightarrow \overline{J}_i \sim N(J_i, \frac{\sigma_i^2}{N_i})$
- OCBA restated: How should N_i be increased that PCS is maximized $\Rightarrow$ Choose $N_1, N_2, \ldots, N_k$ such that PCS is maximized subject to the budget T (limited)
- $$\max_{N_1,\ldots,N_k} PCS \text{ s.t. } \sum^k N_i = T \qquad (2)$$

OCBA Derivation

- No close-form expression for PCS ⇒ approximation via Bonferroni inequality

$$PCS = P\{\bigcap_{i \neq t} \left(\bar{J}_t - \bar{J}_i < 0\right)\}$$

$$\geq 1 - \sum_{i \neq t}^{k} P\{\bar{J}_t > \bar{J}_i\} =: APCS$$

OCBA Derivation

- Approximation for Eq. 2:

$$\max_{N_1,\ldots,N_k} 1 - \sum_{i \neq t}^{k} P\{\bar{J}_t > \bar{J}_i\} \text{ s.t. } \sum^{k} N_i = T$$

$\Rightarrow$

- Lagrange
- assumption: $N_t >> N_i$
- $T \to \infty$

$\Rightarrow$ Theorem:

OCBA Derivation

- Given a total number of simulation replications T to be allocated to k competing designs with with means $\overline{Y}_1, \overline{Y}_2, \ldots, \overline{Y}_k$ and finite variances $\sigma_1^2, \sigma_2^2, \ldots, \sigma_k^2$, respectively. The *Approximate Probability of Correct Selection* can be asymptotically maximized when

$$\frac{N_i}{N_j} = \left(\frac{\sigma_i / \delta_{b,i}}{\sigma_j / \delta_{b,j}} \right)^2, \quad i, j \in \{1, 2, \ldots, k\}, \text{ and } i \neq j \neq b, \tag{3}$$

$$N_b = \sigma_b \sqrt{\sum_{i=1, i \neq b} \frac{N_i^2}{\sigma_i^2}},$$

Thomas Bartz-Beielstein.
Experimental Research in Evolutionary Computation—The New Experimentalism.
Natural Computing Series. Springer, Berlin, Heidelberg, New York, 2006.

Thomas Bartz-Beielstein.
Sequential parameter optimization—an annotated bibliography.
CIOP Technical Report 04/10, Research Center CIOP (Computational Intelligence, Optimization and Data Mining), Cologne University of Applied Science, Faculty of Computer Science and Engineering Science, April 2010.

Thomas Bartz-Beielstein, Marco Chiarandini, Luis Paquete, and Mike Preuss, editors.

Experimental Methods for the Analysis of Optimization Algorithms.
Springer, Berlin, Heidelberg, New York, 2010.

Thomas Bartz-Beielstein, Konstantinos E. Parsopoulos, and Michael N. Vrahatis.
Design and analysis of optimization algorithms using computational statistics.
Applied Numerical Analysis and Computational Mathematics (ANACM), 1(2):413–433, 2004.

Chun-Hung Chen and Loo Hay Lee.
Stochastic simulation optimization.
World Scientific, 2011.

Alexander Forrester, Andras Sobester, and Andy Keane.
Engineering Design via Surrogate Modelling.

Wiley, 2008.

J. P. C. Kleijnen.
Statistical Tools for Simulation Practitioners.
Marcel Dekker, New York NY, 1987.

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

Christian W. G. Lasarczyk.
Genetische Programmierung einer algorithmischen Chemie.

PhD thesis, Technische Universität Dortmund, 2007.