DOI of Definitive Version: http://dx.doi.org/10.1145/2464576.2480816

How to Create Meaningful and Generalizable Results

Thomas Bartz-Beielstein, Martin Zaefferer, Boris Naujoks

[firstname].[lastname]@fh-koeln.de
spotseven.org
(Cologne University of Applied Sciences)

July 2013

GECCO'13 Companion, July 6-10, 2013, Amsterdam, The Netherlands.
ACM 978-1-4503-1964-5/13/07.

Agenda

Your Instructors Today

- Dr.Thomas Bartz-Beielstein is a professor for Applied Mathematics at Cologne University of Applied Sciences. He has published more than several dozen research papers, presented tutorials about tuning, and has edited several books in the field of Computational Intelligence.
- Martin Zaefferer is a research assistant at Cologne University of Applied Sciences. His research interests include computational intelligence, applications of knowledge discovery as well as simulation and model based optimization.
- Dr. Boris Naujoks is one of the leading scientists on multi-criteria decision making in Germany. He managed different projects in applying evolutionary multi objective optimization techniques in different real-world applications from airfoil design in aerospace industry to vehicle routing problems in logistics.

Questions

Q-1: How to generate test problems?
Q-2: How to generalize results?

Benchmarking: Features

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

See,e.g, [4]

Benchmarking: Current Situation

- Authors report parameter values which seem to work reasonably well
- Each algorithm will be run for some number, say ten, on each problem. Statistics are reported, e.g., mean, standard deviation
- 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?

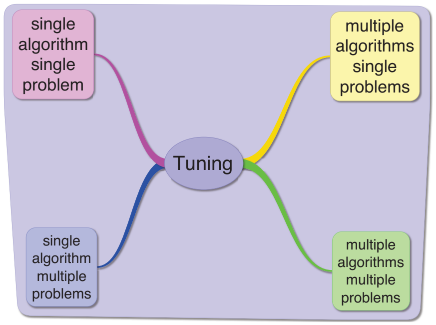
Benchmarking: Open Questions

- Algorithms are trained for this specific set of benchmark functions
  - Who defines this set of functions?
  - Fixed set of test data?
- 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 operator 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 $Y_1, \dots, Y_r$ (independent, identically distributed)
- One instance π, run the algorithm r times $\Rightarrow$ r replicates of the performance measure Y, denoted by $Y_1, \dots, Y_r$
- Samples are conditionally on the sampled instance and given the random nature of the algorithm, independent and identically distributed (i.i.d.), i.e.,

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

MASP and SAMP: Algorithm and Problem Designs

- MASP
  - Several optimization algorithms are compared on one fixed problem instance π
  - Experiment: collect sample data $Y_1, \dots, Y_R$ (independent, identically distributed)
  - Goal: comparison of algorithms on one (real-world) problem instance π
  - No generalization
- SAMP
  - Generalization!
  - Goal: Drawing conclusions about a certain *class* or *population* of instances Π
  - This is Q-1: How to generate a population of problem instances?

Test Problem Generators

- Artificial
- Natural

- Three fundamental steps for generating natural problem instances, namely
  Describing the real-world system and its data
  Feature extraction
  Instance generation

Example: Test Problem Generators

- Describing the real-world system and its data
- Classic Box and Jenkins airline data [2]
- Monthly totals of international airline passengers, 1949 to 1960
- > str(AirPassengers)

```
Time-Series [1:144] from 1949 to 1961: 112 118 132 129 121 135 148 148 136 119 ...
```


Example: Test Problem Generators

- Feature extraction based on methods from time-series analysis
- Multiplicative Holt-Winters (HW) prediction function (for time series with period length p) is

$$\hat{Y}_{t+h} = (a_t + hb_t)s_{t-p+1+(h-1) \mod p},$$

where a_t, b_t and s_t are given by

$$\begin{aligned} a_t &= \alpha(Y_t/s_{t-p}) + (1-\alpha)(a_{t-1}+b_{t-1}) \\ b_t &= \beta(a_t - a_{t-1}) + (1-\beta)b_{t-1} \\ s_t &= \gamma(Y_t/a_t) + (1-\gamma)s_{t-p} \end{aligned}$$

- The optimal values of α, β and γ are determined by minimizing the squared one-step prediction error

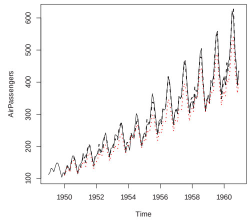
Example: Test Problem Generators

- Instance generation
- HW parameters α, β, and γ are estimated from original time-series data Y_t
- To generate new problem instances, these parameters can be slightly modified
- Based on these modified values, the model is re-fitted
- Extract the new time series. Here, we plot the original data, the Holt-Winters predictions and the modified time series.

Example: Test Problem Generators

```
> generateHW <- function(a,b,c){
+ ## Estimation
+   m <- HoltWinters(AirPassengers, seasonal = "mult")
+ ## Extraction
+   alpha0<-m$alpha
+   beta0<-m$beta
+   gamma0<-m$gamma
+ ## Modification
+   alpha1 <- alpha0*a
+   beta1 <- beta0*b
+   gamma1 <- gamma0*c
+ ## Re-estimation
+   m1 <- HoltWinters(AirPassengers, alpha=alpha1
+   , beta = beta1, gamma = gamma1)
+ ## Instance generation
+   plot(AirPassengers)
+   lines(fitted(m)[,1], col = 1, lty=2, lw=2)
+   lines(fitted(m1)[,1], lty = 3, lw =2, col = 2)
+ }
> generateHW(a=.05,b=.025,c=.5)
```

Example: Test Problem Generators



- HW problem instance generator: *solid line:* real data, *dotted line:* predictions from the Holt-Winters model, *fine dotted red line:* modified predictions

Example: Artificial Test Problem Generators

- Gallagher and Yuan present landscape test generator *Max-Set of Gaussian Landscape Generator* (GLG) [4]
- Problem instances for continuous, bound-constrained optimization problems
- Uses m weighted Gaussian functions

$$G(x) = \max_{i \in 1,2,\dots,m} (w_i g_i(x)),$$

where $g : \mathbb{R}^n \to \mathbb{R}$ denotes an n-dimensional Gaussian function

$$g(x) = \left(\frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)\Sigma^{-1}(x-\mu)^T \right) \right)^{1/n},$$

μ is an n-dimensional vector of means, and Σ is an $(n \times n)$ covariance matrix

- Mean of each Gaussian corresponds to an optimum on the landscape and the location of all optima is known
- Global optimum is the one with the largest value

Example: GLG Instance

- The following parameters can be used to specify the GLG generator
  - The number of Gaussian components m
  - The mean vector μ of each component
  - The covariance matrix Σ of each component
  - The weight of each component w_i
  - A maximum threshold $t \in [0; 1]$ can be specified for local optima and the fitness value of the global optimum G^*. Local optima are randomly generated within $[0; t \times G^*]$
- The following tuple can be used to specify an GLG generator:

$$\Pi := ([-c, c]^n, n, m, D_\mu, \{D_\Sigma\}, \{t, G^*\}), \tag{2}$$

where $c \in \mathbb{R}$ defines the boundary constraints of the search space, n the search space dimensionality, m the number of Gaussian components, D_μ the distribution used to generate the mean vectors of components, D_Σ the distribution or procedures used to generate covariances of components, $t \in [0; 1]$ the threshold for local optima, and G^* the function value of the global optimum

Example: GLG Instance

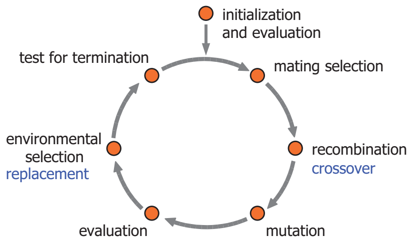
- Based on Eq. (2), we have specified the following GLG landscape generator for our experiments:

$$\Pi_1 := \left([-1; 1]^2, 2, 20, \mathcal{U}[-1; 1], \{\mathcal{U}[0.05; 0.15], \mathcal{U}[-\pi/4, \pi/4]\}, \{0.9, 10\}\right) \tag{3}$$

- Mean vector of each component is generated randomly within $[-1, 1]^2$
- Covariance matrix of each component generated with the procedure D_Σ in three steps:

  A diagonal matrix S with eigenvalues is generated
  An orthogonal matrix T is generated through $n(n-1)/2$ rotations with random angles between $[-\pi/4, \pi/4]$
  The covariance matrix generated as $T^T ST$
- The weight w_i of the component corresponding to the global optimum is set to 10 while other weights are randomly generated within $[0; 0.9]$
- Nine problem instances, $\pi_i \in \Pi_1$, $(i = 1, \dots, 9)$, see Fig. 1, generated with this parametrization

Evolution Strategy



Evolution Strategy

Parameter	Symbol	Name	Range	Value
mue	μ	Number of parent individuals	$\mathbb{N}$	5
nu	$\nu = \lambda/\mu$	Offspring-parent ratio	$\mathbb{R}_+$	2
sigmaInit	$\sigma_i^{(0)}$	Initial standard deviations	$\mathbb{R}_+$	1
nSigma	n_σ	Number of standard deviations. d denotes the problem dimension	$\{1, d\}$	1
	c_τ	Multiplier for mutation	$\mathbb{R}_+$	1
tau0			$\mathbb{R}_+$	0
tau			$\mathbb{R}_+$	1
rho	ρ	Mixing number	$\{1, \mu\}$	2
sel	κ	Maximum age	$\mathbb{R}_+$	1
mutation		Mutation	$\{1, 2\}$	2
sreco	r_σ	Recombination: strategy vars	$\{1, 2, 3, 4\}$	3
oreco	r_x	Recombination: object vars	$\{1, 2, 3, 4\}$	2

SAMP: Fixed Algorithm and Randomized Problem Designs

- SAMP-1: Algorithm and Problem Instances
- SAMP-2: Building the Model and ANOVA
- SAMP-3: Validation of the Model Assumptions
- SAMP-4: Hypothesis Testing
- SAMP-5: Confidence Intervals and Prediction

SAMP-1: Problem Instances

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

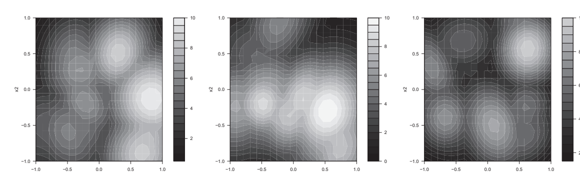


Figure : Three test problem instances from Π_1, which were generated with the GLG landscape generator as specified in Eq. 3.

SAMP-1: Algorithm and Problem Instances

- ES, run $r = 10$ times on a set of randomly generated problem instances

```
'data.frame': 90 obs. of  4 variables:
 $ y      : num  0.20749 0.26074 0.00134 0.23667 0.38032 ...
 $ yLog   : num  -1.573 -1.344 -6.614 -1.441 -0.967 ...
 $ algSeed: Factor w/ 10 levels "123","124","125",..: 1 2 3 4 5 6 7 8 9 10 ...
 $ fSeed  : Factor w/ 9 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
```


SAMP-2 Building the Model and ANOVA

- Linear statistical model

$$Y_{ij} = \mu + \tau_i + \varepsilon_{ij} \begin{cases} i = 1, \dots, q \\ j = 1, \dots, r, \end{cases} \tag{4}$$

  where μ is an overall mean and ε_{ij} is a random error term for replication j on instance i
- Note, in contrast to the fixed-effects model, τ_i is a random variable representing the effect of instance i
- The stochastic behavior of the response variable originates from both the instance and the algorithm
- This is reflected in (4), where both τ_i and ϵ_{ij} are random variables
- The model (4) is the so-called *random-effects model*, cf. [5, p. 512] or [3, p. 229].

SAMP-2: The classical ANOVA

- Similar to classical ANOVA: variability in the observations can be partitioned into a component that measures the variation between treatments and a component that measures the variation within treatments
- Based on ANOVA identity $\text{SS}_{\text{total}} = \text{SS}_{\text{treat}} + \text{SS}_{\text{err}}$, we define

$$\text{MS}_{\text{treat}} = \frac{\text{SS}_{\text{treat}}}{q-1} = \frac{r\sum_{i=1}^{q}(\bar{Y}_{i.} - \bar{Y}_{..})^2}{q-1},$$

$$\text{MS}_{\text{err}} = \frac{\text{SS}_{\text{err}}}{q(r-1)} = \frac{\sum_{i=1}^{q}\sum_{j=1}^{r}(Y_{ij} - \bar{Y}_{i.})^2}{q(r-1)}$$

- It can be shown [5] that

$$E(\text{MS}_{\text{treat}}) = \sigma^2 + r\sigma_\tau^2 \quad \text{and} \quad E(\text{MS}_{\text{err}}) = \sigma^2, \tag{5}$$

- Estimators of variance components

$$\hat{\sigma}^2 = \text{MS}_{\text{err}} \quad \text{and} \quad \hat{\sigma}_\tau^2 = \frac{\text{MS}_{\text{treat}} - \text{MS}_{\text{err}}}{r} \tag{6}$$

SAMP-2: 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}	$qr-1$			

- Expected mean squares differ

SAMP-2: ANOVA Calculations in R (1/2)

- Extract mean squared values MSA (treatment) and MSE (error) from ANOVA model
- Calculate estimators of variance components from (6): $\hat{\sigma}^2$ as the mean squared error and the second component $\hat{\sigma}_\tau^2$

```
> samp.aov <- aov(yLog ~fSeed, data=samp.df)
> (M1 <- anova(samp.aov))
Analysis of Variance Table

Response: yLog
          Df Sum Sq Mean Sq F value  Pr(>F)
fSeed      8  87.28 10.9102  2.6143 0.01346 *
Residuals 81 338.03  4.1733
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
> (MSA <- M1[1,3])
[1] 10.91023
> (MSE <- M1[2,3])
[1] 4.173264
> r <-length(unique(samp.df$algSeed)); q <- nlevels(samp.df$fSeed)
> (var.A <- (MSA - MSE)/(r))
[1] 0.6736962
> (var.E <- MSE)
[1] 4.173264
```


SAMP-2: ANOVA Calculations in R (2/2)

- Finally, the mean μ from (4) can extracted

```
> coef(samp.aov)[1]
(Intercept)
  -2.440496
```

- The p value in the ANOVA table is calculated as

```
> 1-pf(MSA/MSE,q-1,q*(r-1))
[1] 0.01346323
```

- Store ANOVA MSA for later:

```
> MSA.anova <- MSA
```


SAMP-2: ANOVA Problems?

- In some cases, the standard ANOVA, which was used in our example, produces a negative estimate of a variance component
- This can be seen in (6): If $\mathrm{MS}_{\mathrm{err}} > \mathrm{MS}_{\mathrm{treat}}$, negative values occur
- By definition, variance components are positive
- Methods, which always yield positive variance components have been developed: *restricted* (or residual, or reduced) *maximum likelihood estimators* (REML)
- The ANOVA method of variance component estimation, which is a method of moments procedure, and REML estimation may lead to different results

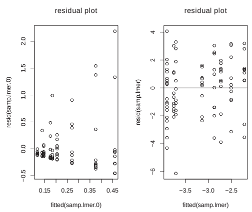
SAMP-2: Restricted Maximum Likelihood

- Based on same data: fit the random-effects model (4) using function `Rlmer` from R package `Rlmefour` [1]:

```
> library(lme4)
> samp.lmer.0 <- lmer(y~ 1 +(1|fSeed),data=samp.df)
> samp.lmer <- lmer(yLog~ 1 +(1|fSeed),data=samp.df)
> print(samp.lmer, digits = 4, corr = FALSE)

Linear mixed model fit by REML
Formula: yLog ~ 1 + (1 | fSeed)
   Data: samp.df
   AIC   BIC logLik deviance REMLdev
 397.9 405.4   -196    391.6   391.9
Random effects:
 Groups   Name        Variance Std.Dev.
 fSeed    (Intercept) 0.6737   0.82079
 Residual             4.1733   2.04286
Number of obs: 90, groups: fSeed, 9

Fixed effects:
            Estimate Std. Error t value
(Intercept)  -3.1912     0.3481  -9.166
```

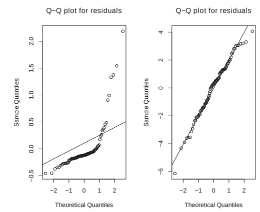
SAMP-3 Validation of the Model Assumptions

- Checking that residuals all have the same variance
- *Left:* raw data, *right:* log-transformed data



SAMP-3 Validation of the Model Assumptions

- Quantile plots (QQ plots) to validate normality assumptions
- *Left:* raw data, *right:* log-transformed data



SAMP-4 Hypothesis Testing

- Testing hypotheses about individual treatments (instances) is useless, because problem instances π_i samples from some larger population of instances Π
- We test hypotheses about the variance component σ_τ^2, i.e., the null hypothesis

$$H_0 : \sigma_\tau^2 = 0 \qquad \text{is tested versus the alternative} \qquad H_1 : \sigma_\tau^2 > 0. \tag{7}$$

- Under H_0, all treatments are identical, i.e., $r\sigma_\tau^2$ is very small
- Conclude from (5): $E(\mathrm{MS_{treat}}) = \sigma^2 + r\sigma_\tau^2$ and $E(\mathrm{MS_{err}}) = \sigma^2$ are similar
- Under the alternative, variability exists between treatments.
- Standard analysis shows: $\mathrm{SS_{err}}/\sigma^2$ is distributed as chi-square with $q(r-1)$ degrees of freedom. Under H_0, the ratio

$$F_0 = \frac{\frac{\mathrm{SS_{treat}}}{q-1}}{\frac{\mathrm{SS_{err}}}{q(r-1)}} = \frac{\mathrm{MS_{treat}}}{\mathrm{MS_{err}}} \sim F_{q-1,q(r-1)}$$

- Requirements for testing hypotheses in (4): $\tau_1, \ldots, \tau_q$ are i.i.d. $\mathcal{N}(0, \sigma_\tau^2)$, ε_{ij}, $i = 1, \ldots, q$, $j = 1, \ldots, r$, are i.i.d. $\mathcal{N}(0, \sigma^2)$, and all τ_i and ε_{ij} are independent of each other

SAMP-4 Hypothesis Testing and Decision Rules

- Considerations lead decision rule to reject H_0 at the significance level α if

$$f_0 > F(1-\alpha; q-1, q(r-1)), \tag{8}$$

where f_0 is the realization of F_0 from the observed data
- Intuitive motivation for the form of statistic F_0 can be obtained from the expected mean squares:
  - Under H_0 both $\mathrm{MS_{treat}}$ and $\mathrm{MS_{err}}$ estimate σ^2 in an unbiased way, and F_0 can be expected to be close to one
  - On the other hand, large values of F_0 give evidence against H_0

SAMP-4 Hypothesis Testing and Decision Rules in R

- Based on (5), we can determine the F statistic and the p values:

```
> VC <- VarCorr(samp.lmer)
> (sigma.tau <- as.numeric(attr(VC$fSeed,"stddev")))
[1] 0.82079
> (sigma <- as.numeric(attr(VC,"sc")))
[1] 2.042857
> q <- nlevels(samp.df$fSeed); r <- length(unique(samp.df$algSeed))
> (MSA <- sigma^2+r*sigma.tau^2)
[1] 10.91023
> (MSE <- sigma^2)
[1] 4.173264
```

  Determine p value based on (8):

```
> 1-pf(MSA/MSE,q-1,q*(r-1))
[1] 0.01346323
```

- If p value is large, the null hypothesis $H_0 : \sigma_\tau^2 = 0$ from (7) can not be rejected, i.e., this indicates that there is no instance effect
- Small p values indicate that there is an problem instance effect
- A similar conclusion was obtained from the ANOVA method of variance component estimation

SAMP-5 Confidence Intervals and Prediction

- Unbiased estimator of the overall mean μ is

$$\sum_{i=1}^{q}\sum_{j=1}^{r}\frac{y_{ij}}{qr}$$

- Its estimated standard error is given by $\mathrm{se}(\hat{\mu}) = \sqrt{\mathrm{MS}_{\mathrm{treat}}/qr}$ and

$$\frac{\bar{Y}_{\cdot\cdot} - \mu}{\sqrt{\mathrm{MS}_{\mathrm{treat}}/qr}} \sim t_{q(r-1)}$$

- Hence, [3, p. 232] show that confidence limits for μ can be derived as

$$\bar{y}_{\cdot\cdot} \pm t(1-\alpha/2; q(r-1))\sqrt{\mathrm{MS}_{\mathrm{treat}}/qr} \tag{9}$$

SAMP-5 Confidence Intervals and Prediction in R (MLE)

- Prediction of the algorithm's performance on a new instance
- Based on (9), the 95% confidence interval can be calculated as follows.

```
> s <- sqrt(MSA/(q*r))
> Y.. <- mean(samp.df$yLog)
> qsr <- qt(1-0.025,q*(r-1))
> c( Y.. - qsr * s, Y.. + qsr * s)
[1] -3.883996 -2.498484
```

- Using the ANOVA results from above, i.e., MSA.anova, we obtain the same confidence interval
- Similar procedures for combinations of fixed and random effects: *mixed models*, see [3]

Hands-on example in R

- Experimental Framework: R-package SPOT (Sequential Parameter Optimization Toolbox)
- Tuned Algorithm: Evolution Strategy ES
- ES objective function: Gaussian Landscape

- Install SPOT from within R:

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

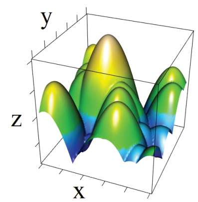
- Load SPOT:

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

Please note: SPOT Version larger than 1.0.4045 is used

Test Function Instance Generator

- Gaussian Landscape Generator GLG
- Based on code by Yuan and Gallagher 2006 [4]
- R implementation in SPOT



Documentation / Help on GLG in SPOT:

```
> ?spotGlgCreate
```


Test Function Instance Generator

- Parameters
  - ngauss: Number of Gaussian components
  - dim: Dimension of the search space
  - lower: Lower boundary
  - upper: Upper boundary
  - maxval: Maximum value (global optimum)
  - ratio: Local optima reach up to *ratio* × *maxval*

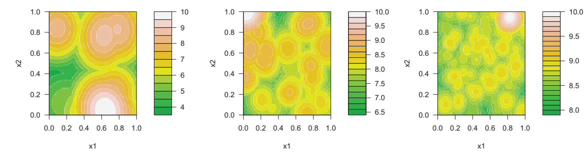


Figure : Landscapes with ngauss set to 20, 200, and 2000

Test Function Instance Generator

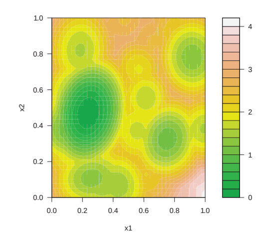
- Generate landscape

```
> require(SPOT)
> set.seed(1)
> #set problem definition
> dim=2
> ngauss=200
> lower <- rep(0,dim)
> upper <- rep(1,dim)
> maxval = 10
> ratio = 0.9
> seedGLG = 123
> #create target function
> fn <- spotGlgCreate(dimension=dim,nGaussian=ngauss,lower=lower,
+  upper=upper, globalvalue=maxval,ratio=ratio,seedGLG)
```


Test Function Instance Generator

- Plot landscape

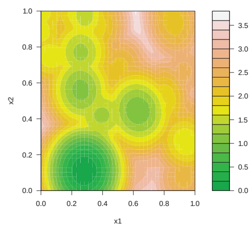
```
> fun <- function(x) return(maxval-fn(x)) #SPOT does minimization.
> spotSurfContour(fun,lo=lower,up=upper,40)
```



Test Function Instance Generator

- Plot an other instance

```
> seedGLG = 1234
> fn <- spotGlgCreate(dimension=dim,nGaussian=ngauss,lower=lower,
+  upper=upper, globalvalue=maxval,ratio=ratio,seedGLG)
> fun <- function(x) return(maxval-fn(x)) #SPOT does minimization.
> spotSurfContour(fun,lo=lower,up=upper,40)
```



Test Function Instance Generator

- Concept of instance generation
  - Parameters kept fixed
  - Different landscapes generated per seed
  - Parameter set –> Problem class
  - Each seed –> Problem instance

Tuned Algorithm: Evolution Strategy

- As already introduced
- See also help:

```
> ?spotOptimEs
```

- Test case SAMP
  - One fixed parameter setting
- Test case MAMP
  - Four recombination operators

Running the ES

- Run ES on one problem instance

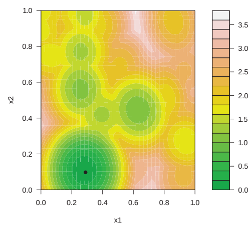
```
> seedES=1
> res <- spotOptimEs(par= rep(NA,dim), fn = fun, lower= lower, upper= upper,
+  control=list(maxit=100,seed=seedES,mue=5,nu=2))
> print(res)
```

```
$par
[1] 0.28933626 0.09753753

$value
[1] 0.006726317

$convergence
[1] 0

$counts
[1] 100
```

Running the ES

- Plot result

```
> spotSurfContour(fun,lo=lower,up=upper,40,points1=matrix(res$par,,2))
```



SAMP: APD File I

File: glges01.apd *(Please note: This file has to be in your R working directory.)*

```
#ES settings
control <- list()
control$maxit <- 100
control$sigmaInit <- 1.0
control$nSigma <- 1
control$tau0 <- 0.0
control$tau <- 1.0
control$stratReco <- 3
control$objReco <- 2
control$kappa <- -1
control$mue <- 5
control$nu <- 2
control$sigmaRestart <- 0
control$prescanmult <- 1
control$mutation <- 2
control$rho <- "bi"
```


SAMP: APD File II

```
control$maxGen <- Inf

#GLG settings
dim=2
lb <- rep(-1,dim)
ub <- rep(1,dim)
ngauss <- 20
maxval <- 10
ratio <- 0.9
npinst <- 9
glgSeed <- 0
```


SAMP: Preparing Experiment

SPOT configuration list

```
> configuration <-list(
+    alg.func="spotAlgStartEsGlg"
+    ,alg.roi=spotROI(c(2,1),c(2,1),varnames=c("NU","TAU"))
+    ,alg.seed = 123
+    ,init.design.func = "spotCreateDesignLhs"
+    ,init.design.size = 1
+    ,init.design.repeats  = 10
+    ,io.verbosity=1
+    ,io.apdFileName = "glges01.apd"
+    ,io.resFileName = "glges01.res"
+    ,io.desFileName = "glges01.des"
+    ,io.bstFileName = "glges01.bst"
+    ,spot.seed = 1234
+    ,spot.fileMode=T
+    ,report.func = "spotReportSAMP")
```

SAMP: Running the Experiment

- First, create the (very simplistic) experimental design

```
> result<-spot(spotConfig=configuration,spotTask="init")
```

```
spot.R::spot started
```

- This will create the design to be evaluated in glges01.des:

NU	TAU	CONFIG	REPEATS	STEP	SEED
2	1	1	10	0	123

- This design can be evaluated:

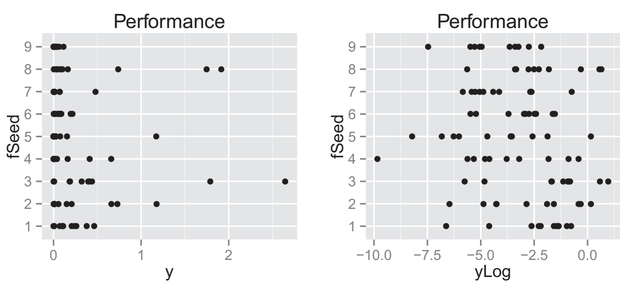
```
> result<-spot(spotConfig=result,spotTask="run")
```


SAMP: Reporting results 1/4

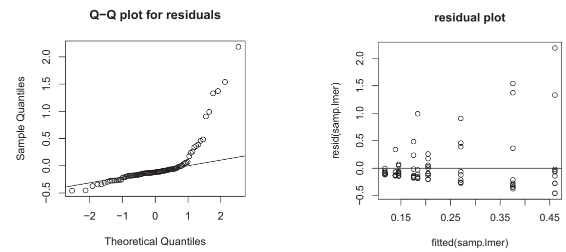
```
> result<-spot(spotConfig=result,spotTask="rep")
```

```
[...]
Linear mixed model fit by REML
Formula: yLog ~ 1 + (1 | fSeed)
   Data: samp.df
   AIC   BIC logLik deviance REMLdev
 397.9 405.4   -196    391.6   391.9
Random effects:
 Groups   Name        Variance Std.Dev.
 fSeed    (Intercept) 0.6737   0.82079
 Residual             4.1733   2.04286
Number of obs: 90, groups: fSeed, 9

Fixed effects:
            Estimate Std. Error t value
(Intercept)  -3.1912     0.3481  -9.166
[1] "P-value log.: 0.013463233651567"
[1] "P-value: 0.0293342642244862"
[1] "Confidence Interval log.: -3.88399600237052 to -2.49848421628946"
[1] "Confidence Interval: 0.0921162444145894 to 0.368410711264746"
```

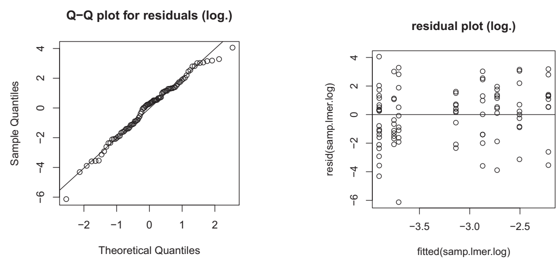

SAMP: Reporting results 2/4



SAMP: Reporting results 3/4



SAMP: Reporting results 4/4



MAMP: Preparing Experiment

- The same APD file is used.
- SPOT configuration list:

```
> configuration <-list(
+    alg.func="spotAlgStartEsGlg"
+    ,alg.roi=spotROI(1,4,varnames="OBJRECO",type="FACTOR")
+    ,alg.seed = 123
+    ,init.design.func = "spotCreateDesignFactors"
+    ,init.design.size = 4
+    ,init.design.repeats  = 10
+    ,io.verbosity=1
+    ,io.apdFileName = "glges01.apd"
+    ,io.resFileName = "glges02.res"
+    ,io.desFileName = "glges02.des"
+    ,io.bstFileName = "glges02.bst"
+    ,spot.seed = 1234
+    ,spot.fileMode=T
+    ,report.func = "spotReportMAMP")
```


MAMP: Running the Experiment

- First, create the experimental design

```
> result<-spot(spotConfig=configuration,spotTask="init")
```

```
spot.R::spot started
```

- This will create the design to be evaluated in glges02.des:

OBJRECO	CONFIG	REPEATS	STEP	SEED
1	1	10	0	123
2	2	10	0	123
3	3	10	0	123
4	4	10	0	123

- This design can be evaluated:

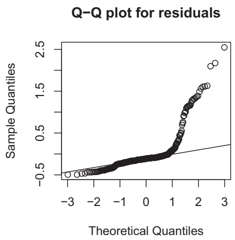
```
> result<-spot(spotConfig=result,spotTask="run")
```

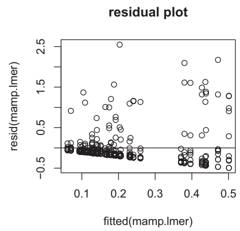

MAMP: Reporting results 1/5

```
> result<-spot(spotConfig=result,spotTask="rep")
```

```
[...]
Linear mixed model fit by REML
Formula: frml
   Data: mamp.df
  AIC  BIC logLik deviance REMLdev
 1664 1691 -824.8     1644    1650
Random effects:
 Groups        Name        Variance   Std.Dev.
 fSeed:OBJRECO (Intercept) 3.8414e-09 6.1979e-05
 fSeed         (Intercept) 4.5459e-01 6.7423e-01
 Residual                  5.4986e+00 2.3449e+00
Number of obs: 360, groups: fSeed:OBJRECO, 36; fSeed, 9

Fixed effects:
            Estimate Std. Error t value
(Intercept)  -3.7512     0.2565 -14.627
OBJRECO1      0.1341     0.2141   0.626
OBJRECO2      0.5599     0.2141   2.616
OBJRECO3      0.0519     0.2141   0.242
```

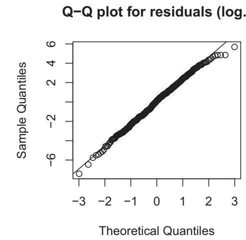
MAMP: Reporting results 2/5

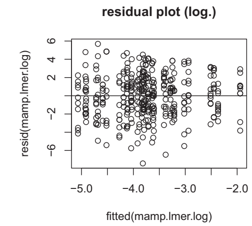




- Residuals not well distributed

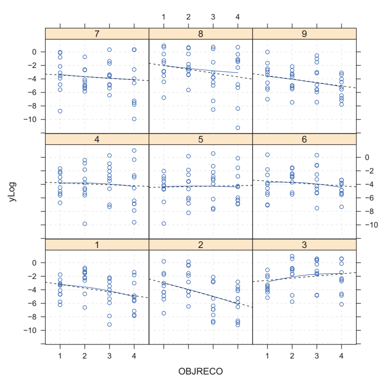
MAMP: Reporting results 3/5



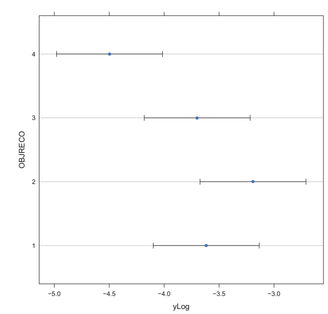


- Better residual distribution for log.-transformed case

MAMP: Reporting results 4/5



MAMP: Reporting results 5/5



- Significant difference between 2-4

Some Remarks

- Just a demonstration
- Analysis needs more instances
- Actual Purpose: real world problems
- Work in progress
- To be improved / future work in SPOT:
  - Ease of use
  - More demos and examples
  - Better reports
  - Visualization
  - Tuning: Predicting/Exploiting models

Overview

- Basics in *Multicriteria Optimization*MCO (short)
- Preliminaries (algorithm + indicator)
- Concept transfer
- Results
- Scientific chances

Basics in MCO

Multicriteria optimization

- Minimize

$$f : \mathbb{R}^n \longrightarrow \mathbb{R}^m, \quad f(x) = (f_1(x), \dots, f_m(x))$$

  - w.r.t.

$$\begin{array}{l} l(p) \leq x_p \leq u(p), \quad p = 1, \dots n \\ g_j(x) \leq 0, \quad j = 1, \dots, r \\ h_k(x) = 0, \quad k = 1, \dots, s \end{array}$$

Concept of Pareto dominance

- Solution x dominates solution y

$$\begin{array}{llll} x <_p y & :\Leftrightarrow & \forall i: \quad f_i(x) \leq f_i(y) & (i = 1, \dots m) \\ & & \exists j: \quad f_j(x) < f_j(y) & (j = 1, \dots m) \end{array}$$

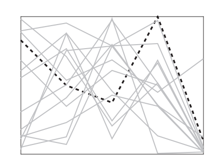
Basics in MCO

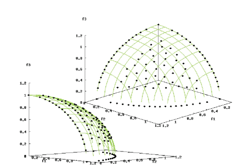
Pareto Dominance

- **Pareto set**: Set of all non-dominated solutions in search space

$$\{x \mid \nexists z : \quad z <_p x\}$$

- **Pareto front**: Image of Pareto set in objective space
- Different Pareto front visualizations:





Preliminaries: How to compare results

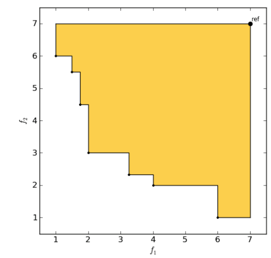
- Different approaches
  - Distance based
  - Spread based
  - Combining both: hypervolume
- Hypervolume
  - Size of space covered by Pareto front
  - w.r.t to reference point
    (parameter of the method)

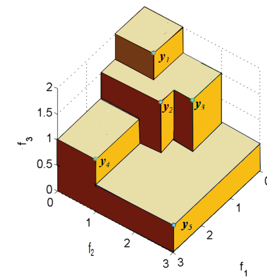
$$\Lambda\left(\bigcup_{a \in A^*} \{y' \mid a \prec y' \prec y_{\text{ref}}\}\right)$$

with

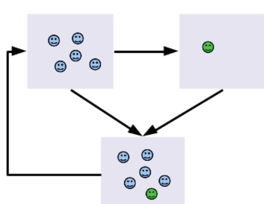
- current Pareto front A^*, reference point y_{ref}

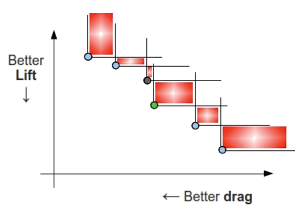
Preliminaries: Hypervolume





Preliminaries: SMS-EMOA





- $(\mu + 1)$ hypervolume selection
  - 1 solution generated by variation
  - solution with least hypervolume contribution omitted
    (secondary ranking criterion, first: non-dominated sorting)

Concept transfer: Problem instances

In practice

- Generate instances of real world problem:
  - Natural instances: part of the problem
  - Artificial instances: Model and randomize each objective
    (see: approach using Holt-Winters above)

In theory

- Gaussian Landscape Generator (see Eq. 3)
  - New instance for each objective?
  - New parametrization of one instance for each objective?
  - (Just?) new realization using same parametrization (one instance)?

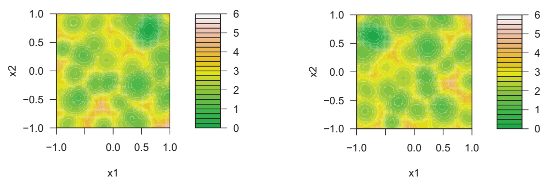
⇒ Complex, difficult: all alternatives have pros and cons

Concept transfer: Problem instances

- Our approach

$$f: \quad f(x) = (f_1(x), \alpha \circ f_1(x))$$

with α rotating the given function by a predefined angle
- Based on suggestion by O. Mersmann
- Scaleable!
  - Rotate by 0 degree: single-objective case
  - Rotate by 180 degree: full symmetric case



Concept transfer: Problem instance example

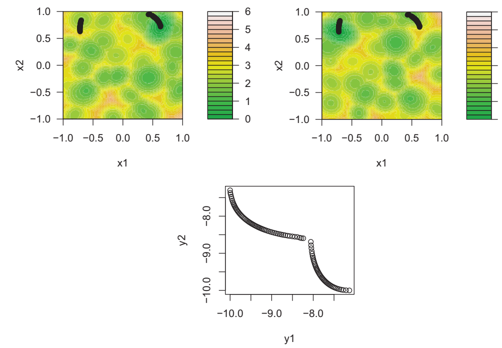


Figure : Pareto front and set with 90 degree rotation.

Concept transfer: Problem instance example

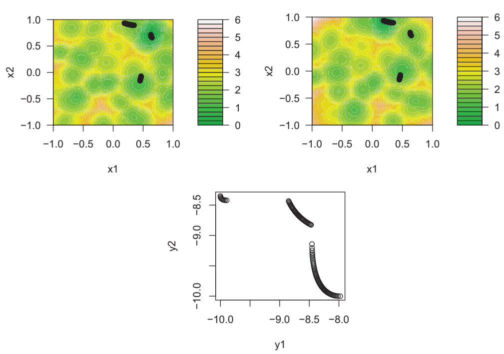


Figure : Pareto front and set with 30 degree rotation.

Concept transfer: Performance indicator

How to compare for different problems?
- different problems yield different hypervolume values
- bias on final results

Our approach
- mean performance of random search
  - generate 1000 random points
  - calculate hypervolume of resulting Pareto front
  - repeat for 100 times
  - ⇒ mean value of 100 hypervolumes considered
- calculate difference between SMS-EMOA result and mean
- consider differences as normalized hypervolume
  - if positive: results are better than for randomized approach
  - if negative: ...... (no good)

MCO SAMP: APD file I

File: smsemoaglg01.apd *(Please note: This file has to be in your R working directory)*

```
#SMS−EMOA settings
control = list()
control$mu = 100
control$maxeval = 1000

#GLG settings
dim = 2
lb = rep(−1,dim)
ub = rep(1,dim)
ngauss= 200
maxval = 10
ratio = 0.9
alpha = pi/6 #30 deg
```


MCO SAMP: APD file II

```
#instances
npinst = 9 #number of random instances
glgSeed = 0 #starting seed for random problem instances
repeats = 100 #repeats for random search

# do not change the following
evals = control$maxeval
```


MCO SAMP: Preparing Experiment

Parametrization

- Design space dimension: 2
- Number of considered instances: 9
- Rotation angle for 2nd objective: 30 degrees
- Number of repetitions per run: 10
- Number of evaluations per run: 1000
- Population size: 100

MCO SAMP: Preparing Experiment

First, create the problem instances

```
> apdfile="smsemoaglg01.apd"
> source(apdfile,local=TRUE)
> seeds=glgSeed:(glgSeed+npinst)
> instances=list()
> for(i in 1:npinst){
+  tmpSeed= glgSeed:(glgSeed+npinst)
+  instances[[i]] <- spotGlgCreateRotSearched(dim,alpha,nGaussian=ngauss,
+  lower=lb, upper=ub, globalvalue=maxval,
+  ratio=ratio,seeds[i],repeats,evals)
+ }
```

MCO SAMP: Preparing Experiment

Second, generate SPOT configuration list

```
> configuration=list(
+    alg.func="spotAlgStartSmsEmoaGlg"
+    ,alg.roi=spotROI(100,100,varnames="mu")
+    ,alg.seed = 12345
+    ,init.design.func = "spotCreateDesignLhs"
+    ,init.design.size = 1
+    ,init.design.repeats  = 10
+    ,io.verbosity=1
+    ,io.apdFileName = apdfile
+    ,io.resFileName = "smsemoaglg01.res"
+    ,io.bstFileName = "smsemoaglg01.bst"
+    ,io.desFileName = "smsemoaglg01.des"
+    ,spot.seed = 125
+    ,spot.fileMode=T
+    ,problem.instances=instances
+    ,report.func = "spotReportSAMP")
```


MCO SAMP: Running the Experiment

- First, create the (very simplistic) experimental design

```
> result<-spot(spotConfig=configuration,spotTask="init")
```

- This will create the design to be evaluated in smsemoaglg01.des:

mu CONFIG REPEATS STEP SEED
100 1 10 0 12345

- This design can be evaluated:

```
> result<-spot(spotConfig=result,spotTask="run")
```

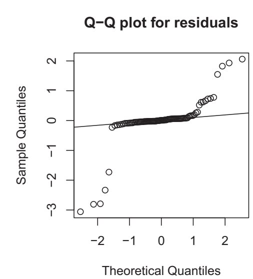

MCO SAMP: Reporting results 1/4

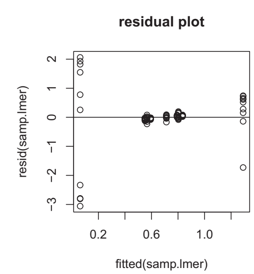
```
> result<-spot(spotConfig=result,spotTask="rep")
[...]
[1] "Summary of the mixed model: "
Linear mixed model fit by REML
Formula: y ~ 1 + (1 | fSeed)
   Data: samp.df
   AIC   BIC logLik deviance REMLdev
 229.1 236.6 -111.5    221.1   223.1
Random effects:
 Groups   Name        Variance Std.Dev.
 fSeed    (Intercept) 0.14862  0.38552
 Residual             0.61081  0.78154
Number of obs: 90, groups: fSeed, 9

Fixed effects:
            Estimate Std. Error t value
(Intercept)   0.6896     0.1526   4.518
[...]
[1] "P-value log.: 0.000640827756405948"
[1] "P-value: 0.00189089008095467"
[1] "Confidence Interval log.: 1.4066864498981 to 1.62119139357494"
[1] "Confidence Interval: 0.385926054509692 to 0.993354615587717"
```

- p value is small, thus the null hypotheses is rejected

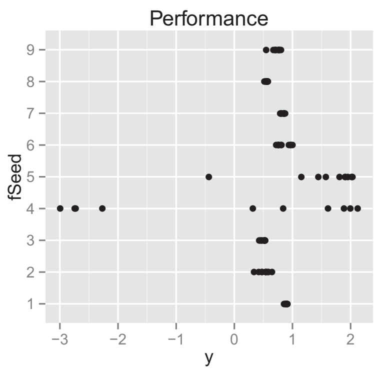
MCO SAMP: Reporting results 2/4

- Distribution of residuals indicates bad model fit
- Log.-transformation not suitable (plots look the same)





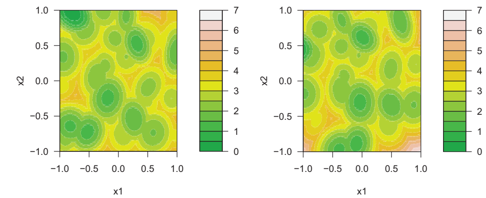
MCO SAMP: Reporting results 3/4



MCO SAMP: Reporting results 4/4

- Landscapes for instance 4:

```
> fun1 <- function(x) return(instances[[4]](x)[,1])
> fun2 <- function(x) return(instances[[4]](x)[,2])
> spotSurfContour(fun1,lb,ub,levels=seq(from=0,to=7,by=0.5))
> spotSurfContour(fun2,lb,ub,levels=seq(from=0,to=7,by=0.5))
```



- rotation moved global optimum of f_1 outside the search space
- in some runs, hypervolume of randomized fronts not achieved
- negative values

MCO MAMP: Preparing Experiment

- Approach like in single-objective case
- Parametrization from SAMP case.
- The same APD file is used.
- Consider population size as factor
- SPOT configuration list:

```
> configuration=list(
+    alg.func="spotAlgStartSmsEmoaGlg"
+    ,alg.roi=spotROI(10,100,varnames="mu",type="INT")
+    ,alg.seed = 12345
+    ,auto.loop.steps = Inf
+    ,auto.loop.nevals = 1
+    ,init.design.func = "spotCreateDesignLhd"
+    ,init.design.size = 5
+    ,init.design.repeats  = 10
+    ,io.verbosity=1
+    ,io.apdFileName = apdfile
+    ,io.resFileName = "smsemoaglg02.res"
+    ,io.bstFileName = "smsemoaglg02.bst"
+    ,io.desFileName = "smsemoaglg02.des"
+    ,spot.seed = 125
+    ,spot.fileMode=T
+    ,problem.instances=instances
+    ,report.func = "spotReportMAMP")
```


MCO MAMP: Running the Experiment

- First, create the experimental design

```
> result<-spot(spotConfig=configuration,spotTask="init")
```

- This will create the design to be evaluated in smsemoaglg02.des:

mu	CONFIG	REPEATS	STEP	SEED
75	1	10	0	12345
14	2	10	0	12345
99	3	10	0	12345
35	4	10	0	12345
53	5	10	0	12345

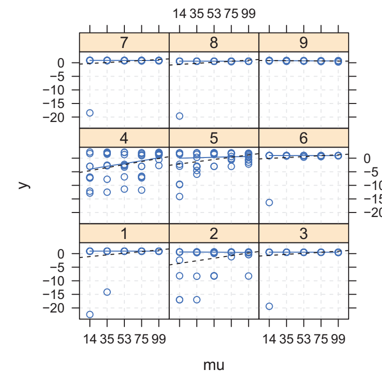
- This design can be evaluated:

```
> result<-spot(spotConfig=result,spotTask="run")
```

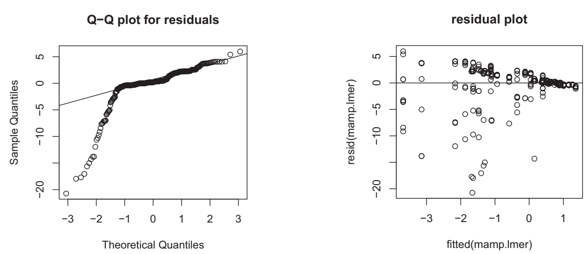
MCO MAMP: Reporting results 1/3

```
> result<-spot(spotConfig=result,spotTask="rep")
[...]
[1] "Summary of the mixed model produced by lmer: "
Linear mixed model fit by REML
Formula: frml
   Data: mamp.df
  AIC  BIC logLik deviance REMLdev
 2369 2402  -1177     2350    2353
Random effects:
 Groups   Name        Variance   Std.Dev.
 fSeed:mu (Intercept) 1.5730e-20 1.2542e-10
 fSeed    (Intercept) 9.2323e-01 9.6085e-01
 Residual             1.0614e+01 3.2580e+00
Number of obs: 450, groups: fSeed:mu, 45; fSeed, 9

Fixed effects:
            Estimate Std. Error t value
(Intercept) -0.25962    0.35514  -0.731
mu1         -1.72623    0.30717  -5.620
mu2          0.08755    0.30717   0.285
mu3          0.32005    0.30717   1.042
mu4          0.48296    0.30717   1.572
[...]
```


MCO MAMP: Reporting results 2/3



MCO MAMP: Reporting results 3/3



MCO Summary, Outlook

- Summary
  - Concept can be transferred to MCO/EMO functions
  - Meaningful results are received
  - Important step in problem understanding
  - Many directions to proceed detected
- Proof of concept
  - Adaptation necessary
    - In theory: problem instance generation
    - In general: indicator
- Problems
  - Rotating optima out of bounds
  - Negative effect on modeling

MCO Summary, Outlook

Potential research directions-1

With respect to proposed concept

- What about results for different rotation angles?
- What about a different concept for MCO problem generation (two others proposed)?
- Alternative ways for comparisons?
  - Hypervolume used in different way?
  - Completely different approach, not invoking hypervolume?

MCO Summary, Outlook

Potential research directions-2

In general

- How do growing angles in fitness function rotation influence the Pareto sets
- When do these separate?
- Influence on Pareto front?
- When does this split . . . and "how"?
- Problem instance generator offers great way to "play" with different functions and investigate Pareto sets, corresponding Pareto fronts, and the mapping in between

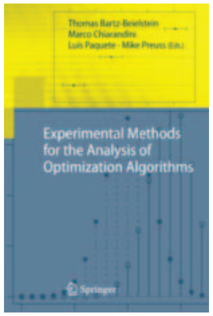
Summary

Q-1: How to generate test problems?
- Randomization!
- Objective
- Systematic approach
- Related to standard ANOVA

Q-2: How to generalize results?
- Randomization!
- Artificial problems and natural problems treated in the same framework
- Confidence intervals (predictable algorithm behavior)

- Updates and additional material can be downloaded from `spotseven.org`

References



Acknowledgments

- This work has been supported by the Federal Ministry of Education and Research (BMBF) under the grants FIWA (AIF FKZ 17N1009), CIMO (FKZ 17002X11) and MCIOP (FKZ 17N0311).

Acknowledgments

[1] Douglas M. Bates.
lme4: Mixed-effects modeling with R.
2010.

[2] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel.
Time Series Analysis, Forecasting and Control.
Holden-Day, 1976.

[3] 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.

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

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

Bartz-Beielstein, Zaefferer, Naujoks (Cologne) Tutorial: meaningful and generalizable results July 2013 97 / 97