Empirical Methods for the Analysis of Optimization Heuristics

Marco Chiarandini

Department of Mathematics and Computer Science
University of Southern Denmark, Odense, Denmark
www.imada.sdu.dk/~marco
www.imada.sdu.dk/~marco/COMISEF08

October 16, 2008
COMISEF Workshop

Outline

Outline

Outline

Capital Asset Pricing Model (CAPM)

Tool for pricing an individual asset i

Individual security's reward-to-risk ratio $= \beta_i \cdot$ Market's securities reward-to-risk ratio

$$\big(E(R_i) - R_f\big) = \beta_i \cdot \big(E(R_m) - R_f\big)$$

β_i sensitivity of the asset returns to market returns

Under normality assumption and least squares method:

$$\beta_i = \frac{\mathrm{Cov}(R_i, R_m)}{\mathrm{Var}(R_m)}$$

Alternatively:

$$R_{it} - R_{ft} = \beta_0 + \beta_1 \cdot (R_{mt} - R_{ft})$$

Use more robust techniques than least squares to determine β_0 and β_1
[Winker, Lyra, Sharpe, 2008]

Least Median of Squares

$$Y_t = \beta_0 + \beta_1 X_t + \epsilon_t$$

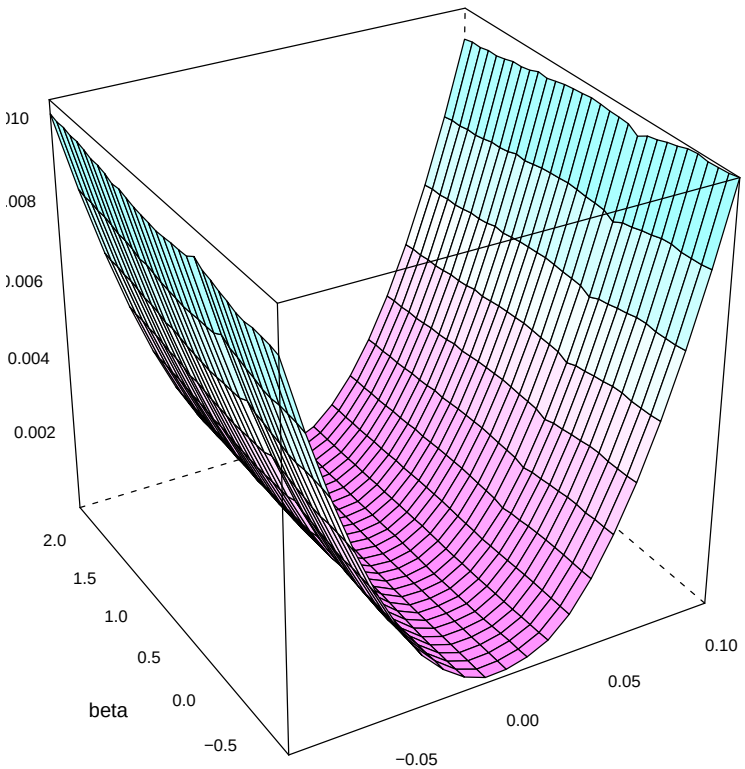
$$\epsilon_t^2 = \left(Y_t - \beta_0 - \beta_1 X_t \right)^2$$

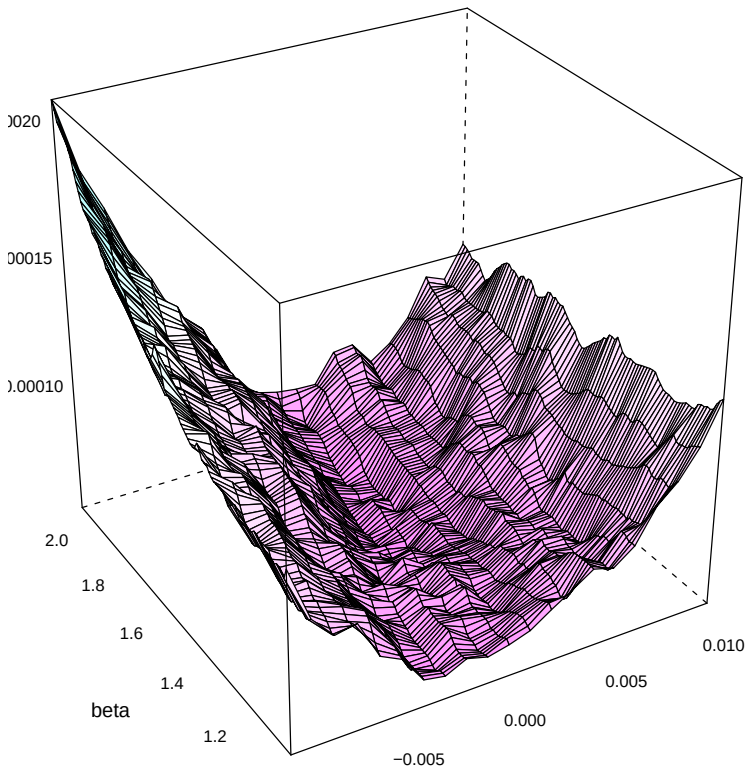
least squares method:

$$\min \sum_{t=1}^{n} \epsilon_t^2$$

least median of squares method:

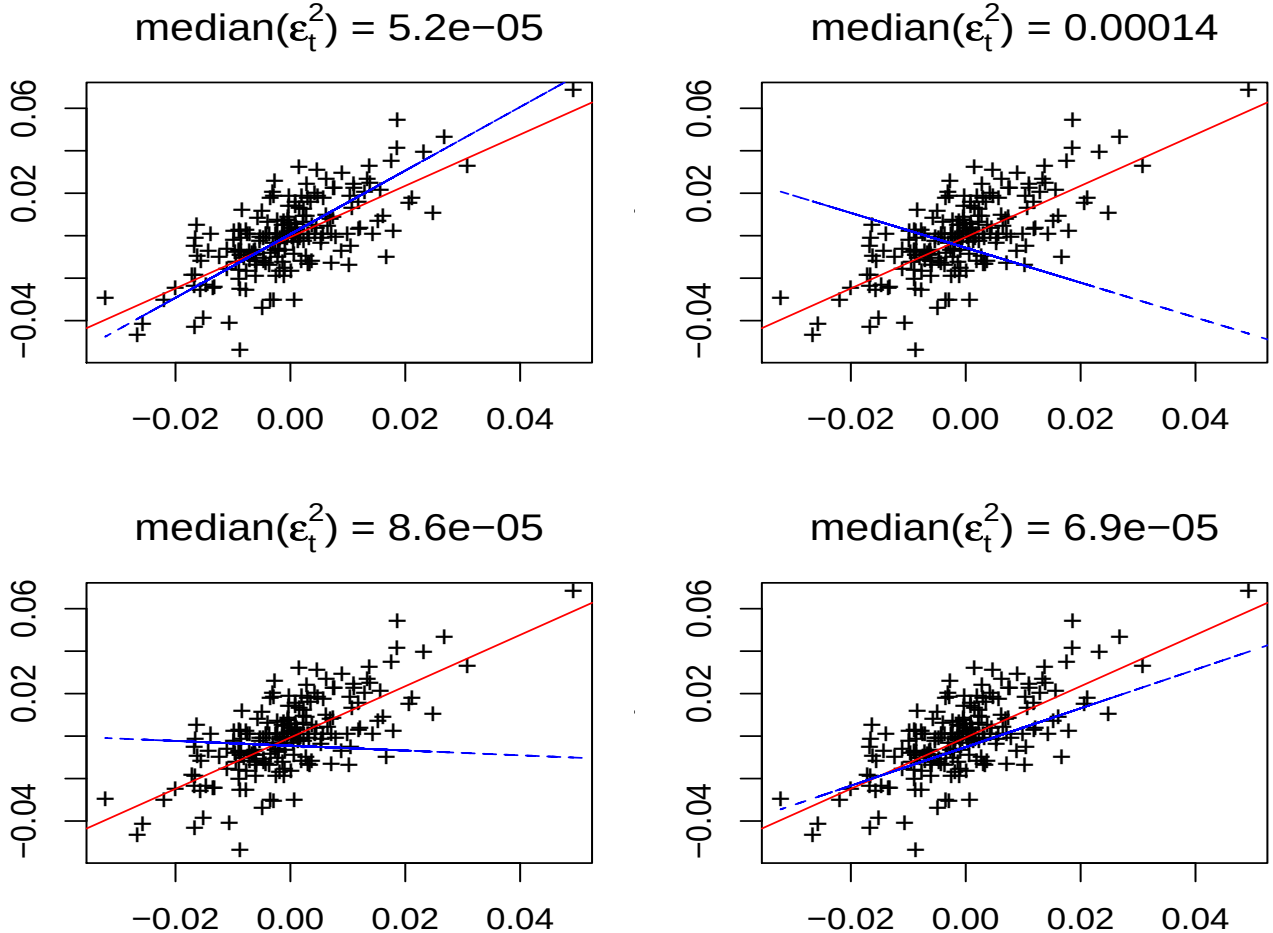
$$\min \left\{ \text{median} \left[\epsilon_t^2 \right] \right\}$$





Optimize non-differentiable, nonlinear and multimodal cost functions.
No analytical methods ➡ optimization heuristics

Four solutions corresponding to four different local optima (red line: least squares; blue line: least median of squares)



Outline

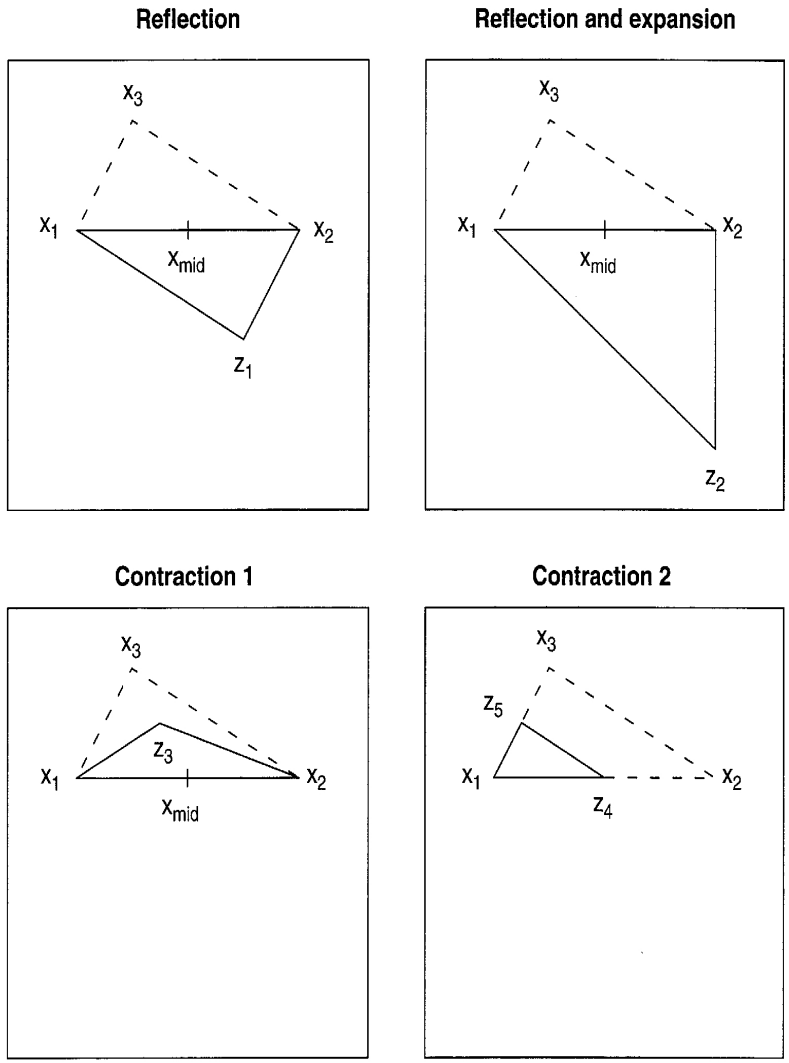
Optimization Heuristics

- Nelder-Mead
- Simulated Annealing
- Differential Evolution

Nelder-Mead

Nelder-Mead simplex method [Nelder and Mead, 1965]:

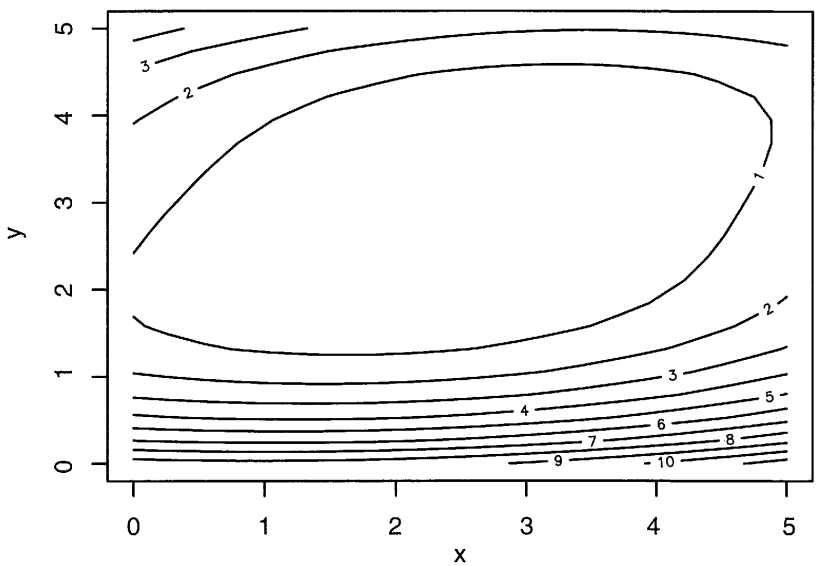
- start from $x_1, \ldots, x_{p+1}$ such that the simplex has nonzero volume
- points are ordered $f(x_1) \leq \ldots \leq f(x_{p+1})$
- At each iteration replace x_{p+1} with a better point among proposed z_i, $i = 1, \ldots, p+3$ constructed as shown

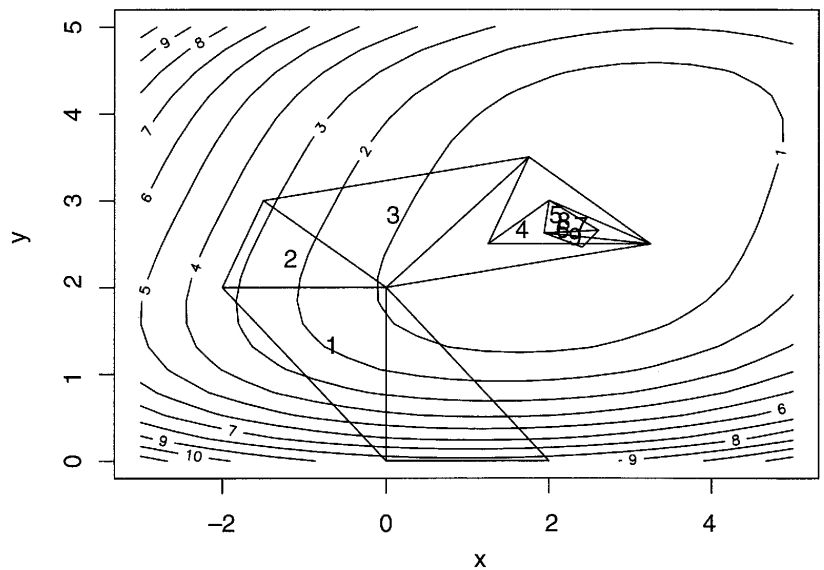


Nelder-Mead

Nelder-Mead simplex method [Nelder and Mead, 1965]:

Example:



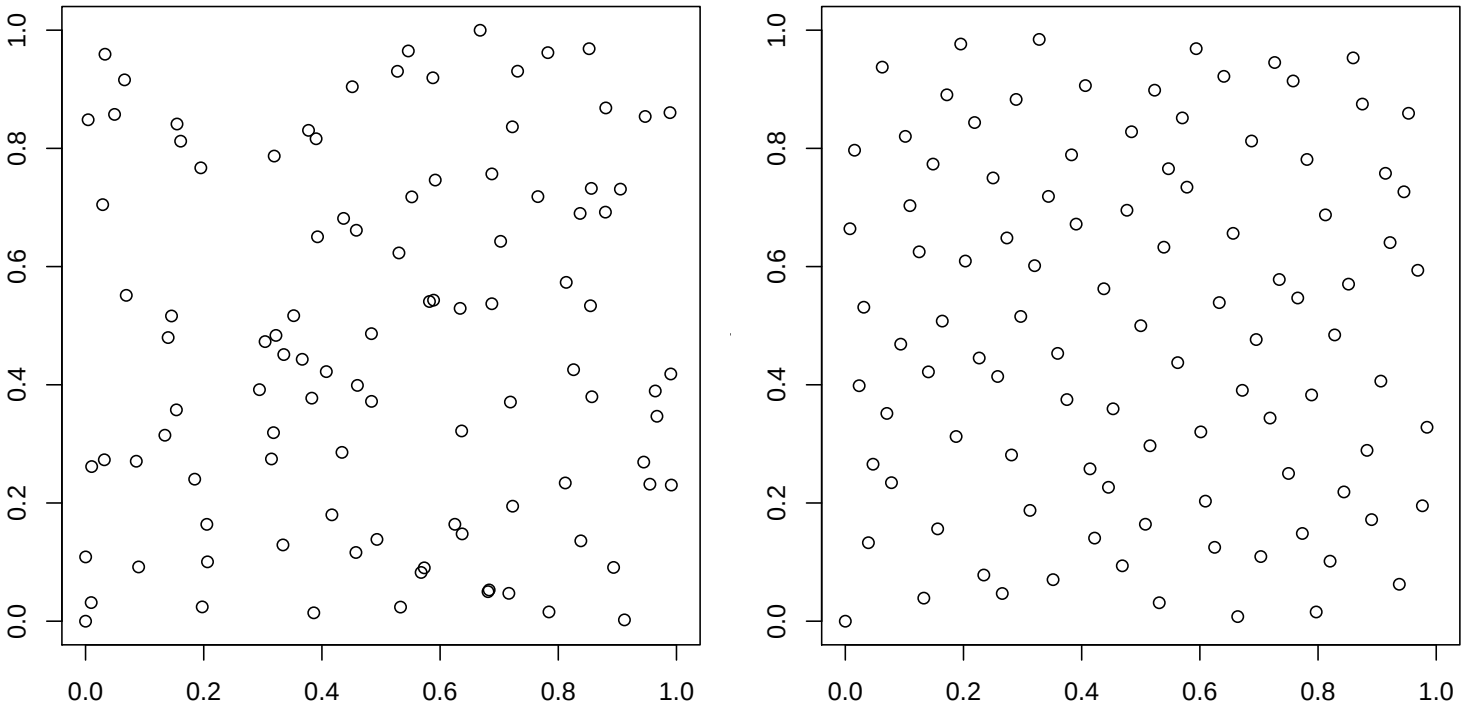


Generation of Initial Solutions

Point generators:

Left: Uniform random distribution (pseudo random number generator)

Right: Quasi-Monte Carlo method: low discrepancy sequence generator [Bratley, Fox, and Niederreiter, 1994].



(for other methods see *spatial point process from spatial statistics*)

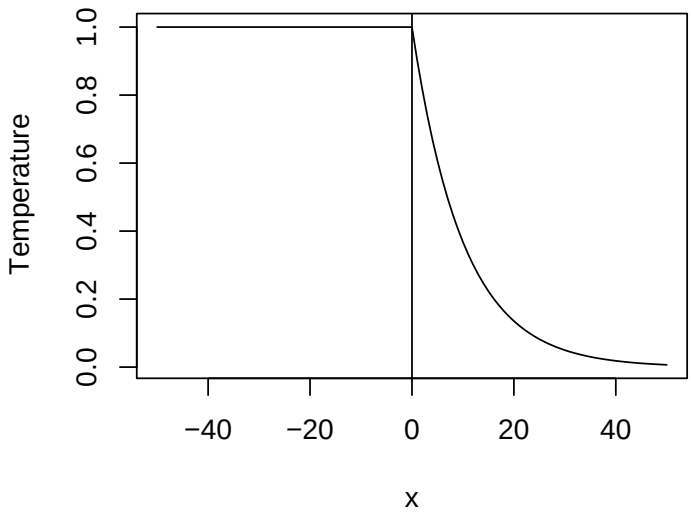
Simulated Annealing

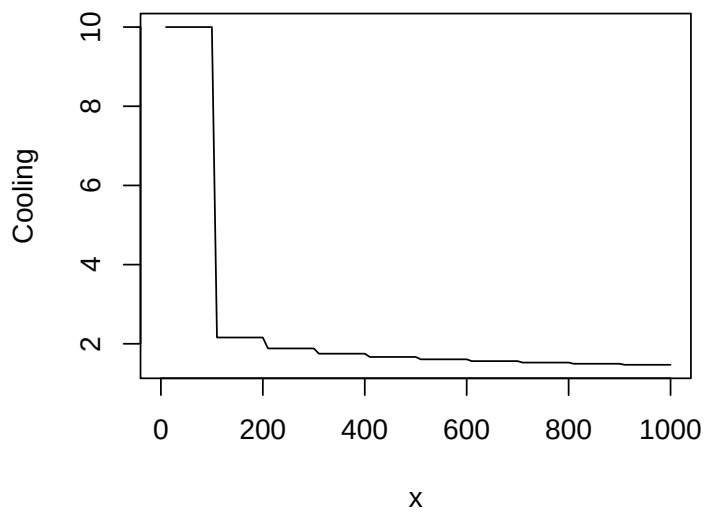
Simulated Annealing (SA):
determine initial candidate solution s
set initial temperature $T = T_0$
while termination condition is not satisfied **do**
  while keep T constant, that is, T_{max} iterations not elapsed **do**
    probabilistically choose a neighbor s' of s
      using proposal mechanism
    accept s' as new search position with probability:

$$p(T, s, s') := \begin{cases} 1 & \text{if } f(s') \leq f(s) \\ \exp \frac{f(s) - f(s')}{T} & \text{otherwise} \end{cases}$$

    update T according to annealing schedule

Simulated Annealing





Proposal mechanism

The next candidate point is generated from a Gaussian Markov kernel with scale proportional to the actual temperature.

Annealing schedule

logarithmic cooling schedule $T = \frac{T_0}{\ln(\lfloor \frac{i-1}{I_{max}} \rfloor I_{max} + e)}$ [Belisle (1992, p. 890)]

Differential Evolution

Differential Evolution (DE)

determine initial population P

while termination criterion is not satisfied **do**

  for each solution x of P **do**

    generate solution u from three solutions of P by mutation

    generate solution v from u by recombination with solution x

    select between x and v solutions

- Solution representation: $x = (x_1, x_2, \ldots, x_p)$
- Mutation:

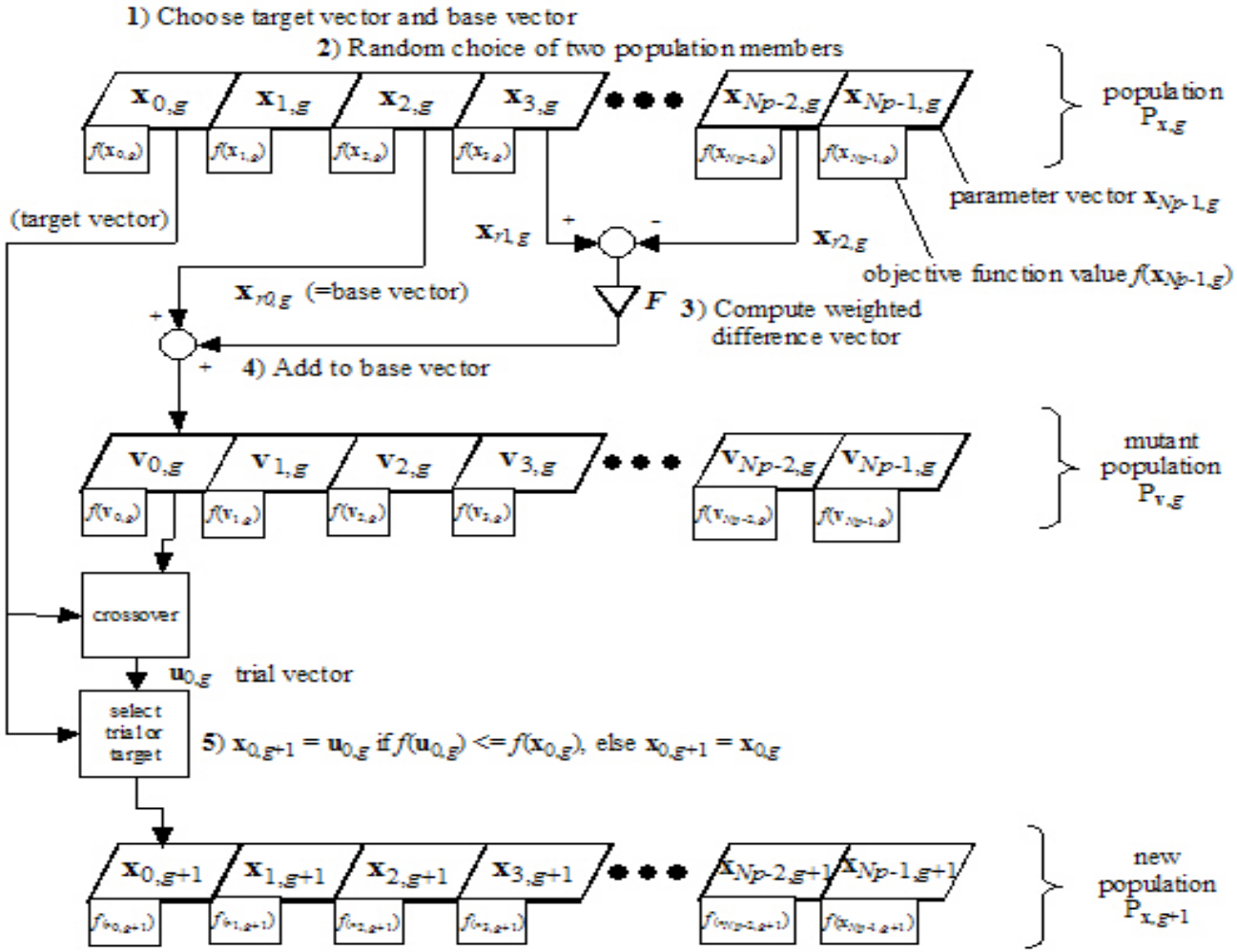
$$u = r_1 + F \cdot (r_2 - r_3) \qquad F \in [0, 2] \text{ and } (r_1, r_2, r_3) \in P$$

- Recombination:

$$v_j = \begin{cases} u_j & \text{if } p < CR \text{ or } j = r \\ x_j & \text{otherwise} \end{cases} \qquad j = 1, 2, \ldots, p$$

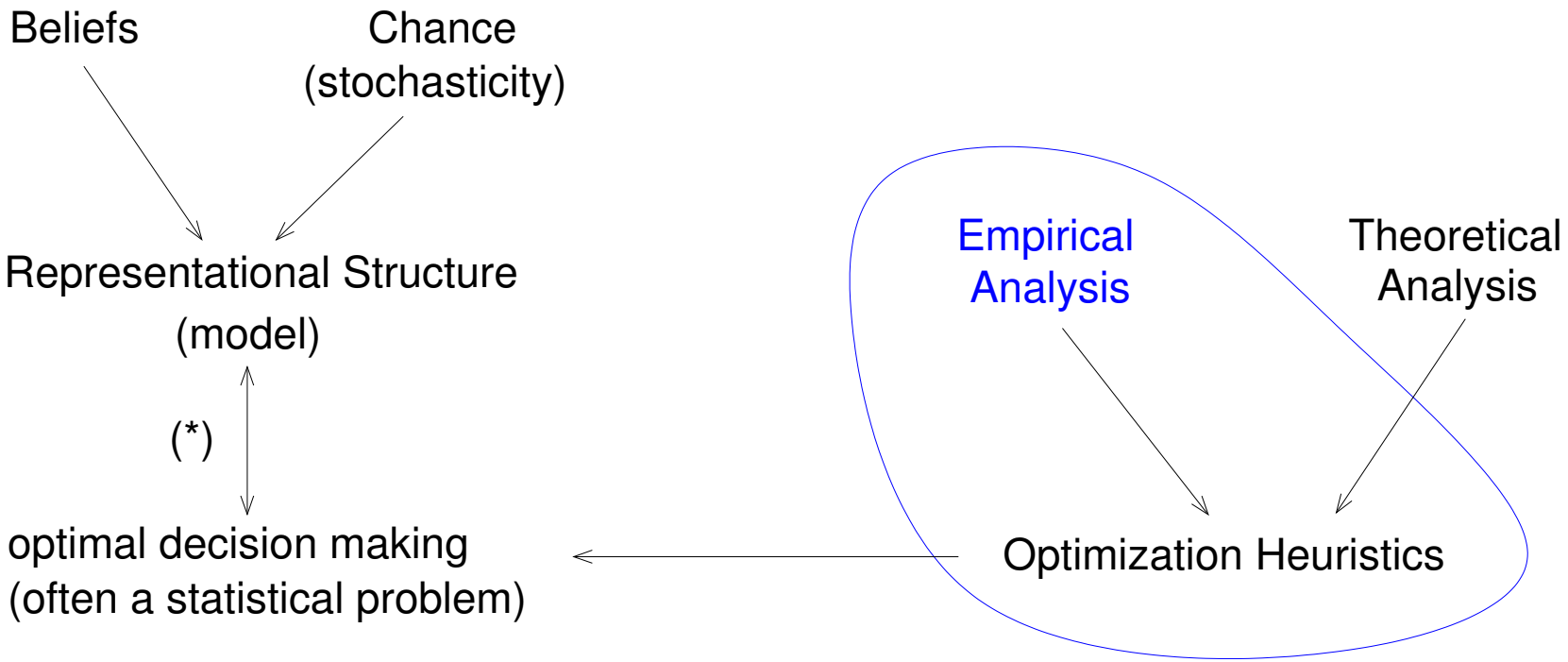
- Selection: replace x with v if $f(v)$ is better

Differential Evolution



[http://www.icsi.berkeley.edu/~storn/code.html
K. Price and R. Storn, 1995]

Dealing with Uncertainty



(*)

- Dodge reality to models that are amenable to mathematical solutions
- Model reality at best without constraints imposed by mathematical complexity

In the CAPM Case Study

Two research questions:

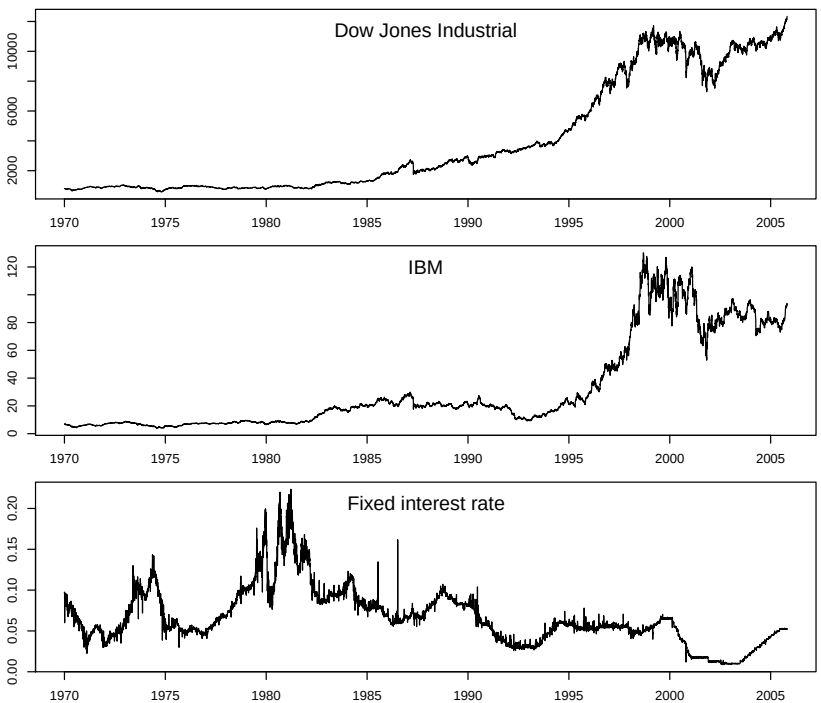
1. Optimization problem
2. Prediction problem (model assessment)

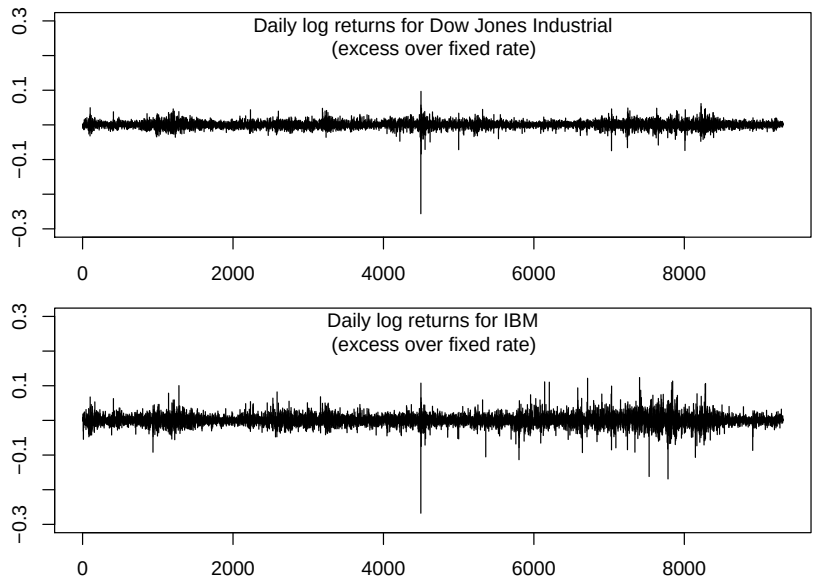
They require different ways to evaluate.

1. Given the model, find algorithm that yields best solutions.
   NM *vs* SA *vs* DE
2. Given that we can solve/tune the model effectively, find the model that yields best predictions
   Least squares method *vs* Least median of squares method
   CAPM *vs* others

Test Data

- Data from the Dow Jones Industrial Average, period 1970-2006.
- Focus on one publicly traded stock
- Use windows of 200 days: $\lfloor 9313/200 \rfloor = 46$
- Each window is an instance from which we determine α and β

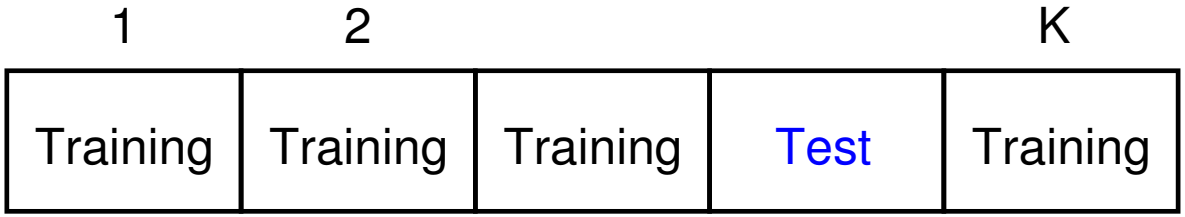




K-Fold Cross Validation

[Stone, 1974]

If goal is estimating prediction error:



1. select kth part for testing
2. train on the other $K-1$ parts for
3. calculate prediction error of the fitted model on the kth part
4. Repeat for $k = 1, \ldots, K$ times and combine the K estimates of prediction error

Outline

Outline

Mathematical analysis

- Through Markov chains modelling some versions of SA, evolutionary algorithms, ant colony optimization can be made to converge with probability 1 to the best possible solutions in the limit [Michiels et al., 2007].

Convergency theory is often derived by sufficient decrease.
x_c current solution x': trial solution

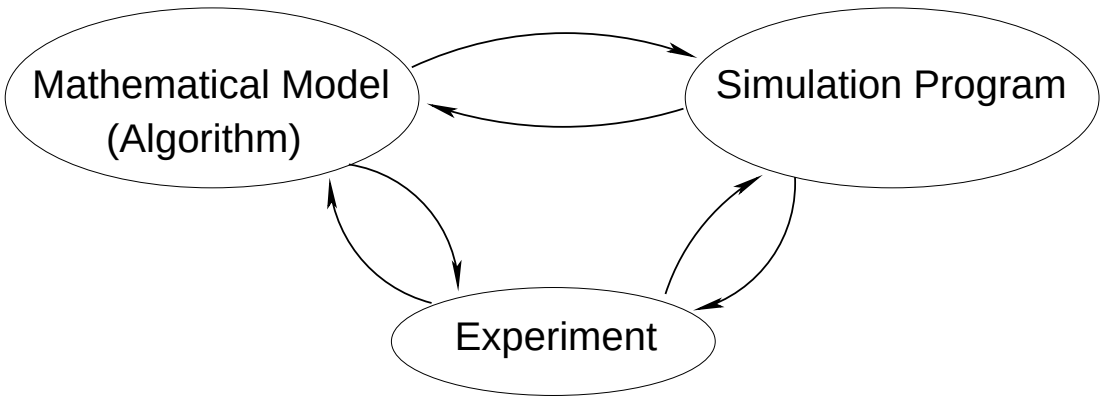
simple decrease	$x = x'$	if $f(x') < f(x_c)$
sufficient decrease	$x = x_c$	if $f(x_c) - f(x') < \epsilon$

Mathematical analysis

- Convergence rates on mathematically tractable functions or with local approximations [Beyer, 2001; Bäck and Hoffmeister, 2004].
- Identification of heuristic component such that they are, for example, "functionally equivalent" to linear transformation of the data of the instance [Birattari et al., 2007]
- Analysis of run time until reaching optimal solution with high probability on pseudo-boolean functions ((1+1)EA,ACO) [Gutjahr, 2008][Dorste et al. 2002, Neumann and Witt, 2006].
- No Free Lunch Theorem: For all possible performance measures, no algorithm is better than another when its performance is averaged over all possible discrete functions [Wolpert and Macready, 1997].

Outline

Experimental Algorithmics



In empirical studies we consider simulation programs which are the implementation of a mathematical model (the algorithm)

[McGeoch (1996), *Toward an Experimental Method for Algorithm Simulation*]

Algorithmic models of programs can vary according to their level of instantiation:

- minimally instantiated (algorithmic framework), e.g., simulated annealing
- mildly instantiated: includes implementation strategies (data structures)
- highly instantiated: includes details specific to a particular programming language or computer architecture

Experimental Algorithmics

[*Theoretician's Guide to the Experimental Analysis of Algorithms*
D.S. Johnson, 2002]

Do publishable work:

- Tie your paper to the literature
  (if your work is new, create benchmarks).
- Use instance testbeds that support general conclusions.
- Ensure comparability.

Efficient:

- Use efficient and effective experimental designs.
- Use reasonably efficient implementations.

Convincing:

- Statistics and data analysis techniques
- Ensure reproducibility
- Report the full story.
- Draw well-justified conclusions and look for explanations.
- Present your data in informative ways.

Goals of Computational Experiments

[*Theoretician's Guide to the Experimental Analysis of Algorithms*
D.S. Johnson, 2002]

As authors, readers or referees, recognize the goal of the experiments and check that the methods match the goals

- To use the code in a particular application. (Application paper) [Interest in output for feasibility check rather than efficiency.]
- To provide evidence of the superiority of your algorithm ideas. (Horse race paper) [Use of benchmarks.]
- To better understand the strengths, weaknesses, and operations of interesting algorithmic ideas in practice. (Experimental analysis paper)
- To generate conjectures about average-case behavior where direct probabilistic analysis is too hard. (Experimental average-case paper)

Outline

Definitions

For each general problem Π (e.g., TSP, CAPM) we denote by C_Π a set (or class) of instances and by $\pi \in C_\Pi$ a single instance.

The object of analysis are randomized search heuristics (with no guarantee of optimality).

- single-pass heuristics: have an embedded termination, for example, upon reaching a certain state

  Eg, Construction heuristics, iterative improvement (eg, Nelder-Mead)

- asymptotic heuristics: do not have an embedded termination and they might improve their solution asymptotically

  Eg., metaheuristics

Scenarios

- Univariate: Y
  Asymptotic heuristics in which:

  Y=X and time limit is an external parameter decided *a priori*

  Y=T and solution quality is an external parameter decided *a priori* (Value To be Reached, approximation error)

- Bivariate: $Y = (X, T)$
  - Single-pass heuristics
  - Asymptotic heuristics with idle iterations as termination condition
- Multivariate: $Y = X(t)$
  - Development over time of cost for asymptotic heuristics

Generalization of Results

On a specific instance, the random variable Y that defines the performance measure of an algorithm is described by its probability distribution/density function

$$Pr(\, Y = y \,|\, \pi)$$

It is often more interesting to generalize the performance on a class of instances C_Π, that is,

$$Pr(\, Y = y, C_\Pi) = \sum_{\pi \in \Pi} Pr(\, Y = y \,|\, \pi) Pr(\pi)$$

Sampling

In experiments,

1. we sample the population of instances and
2. we sample the performance of the algorithm on each sampled instance

If on an instance π we run the algorithm r times then we have r replicates of the performance measure Y, denoted $Y_1, \ldots, Y_r$, which are independent and identically distributed (i.i.d.), i.e.

$$Pr(y_1, \ldots, y_r|\pi) = \prod_{j=1}^{r} Pr(y_j \mid \pi)$$

$$Pr(y_1, \ldots, y_r) = \sum_{\pi \in C_\Pi} Pr(y_1, \ldots, y_r \mid \pi) Pr(\pi).$$

Measures and Transformations

On a class of instances

Computational effort indicators

- process time (user + system time, no wall time).
  it is reliable if process takes > 1.0 seconds
- number of elementary operations/algorithmic iterations (e.g., search steps, cost function evaluations, number of visited nodes in the search tree, etc.)

- no transformation if the interest is in studying scaling
- no transformation if instances from an homogeneously class
- standardization if a fixed time limit is used
- geometric mean (used for a set of numbers whose values are meant to be multiplied together or are exponential in nature)

Measures and Transformations

On a class of instances

Solution quality indicators

Different instances ➡ different scales ➡ need for invariant measures

- Distance or error from a reference value (assume minimization):

$$e_1(x,\pi) = \frac{x(\pi) - \bar{x}(\pi)}{\widehat{\sigma}(\pi)} \quad \text{standard score}$$

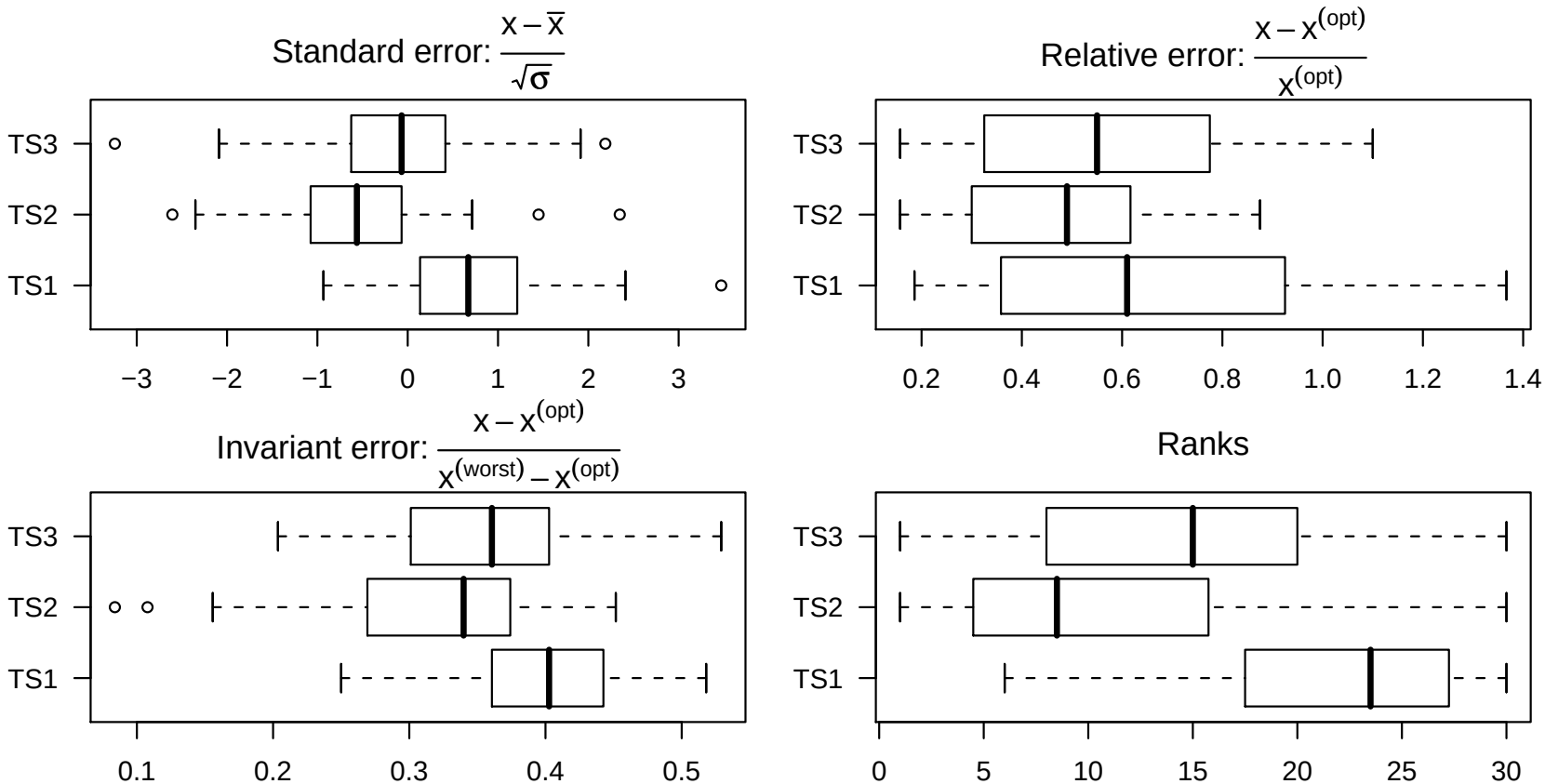
$$e_2(x,\pi) = \frac{x(\pi) - x^{opt}(\pi)}{x^{opt}(\pi)} \quad \text{relative error}$$

$$e_3(x,\pi) = \frac{x(\pi) - x^{opt}(\pi)}{x^{worst}(\pi) - x^{opt}(\pi)} \quad \text{invariant error [Zemel, 1981]}$$

  - optimal value computed exactly or known by instance construction
  - surrogate value such bounds or best known values
- Rank (no need for standardization but loss of information)

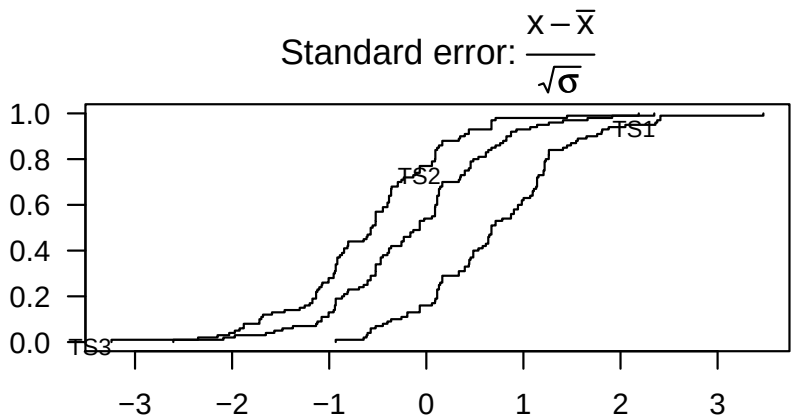
Graphical Representation

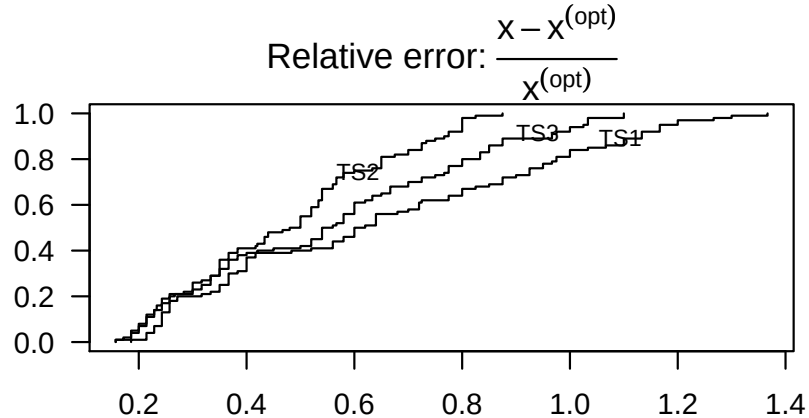
On a class of instances

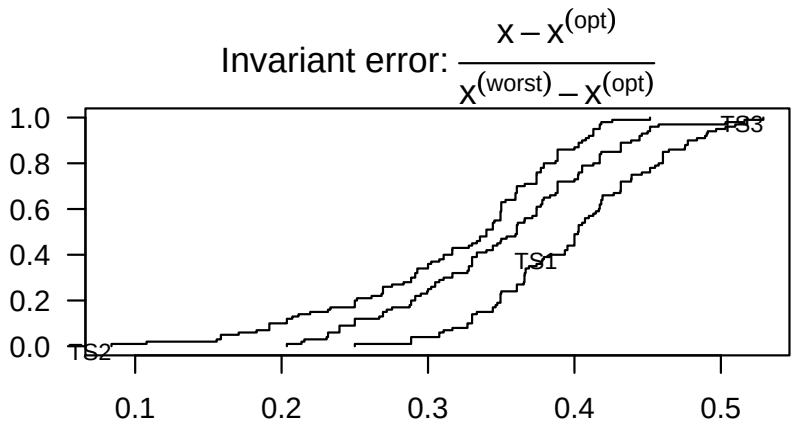


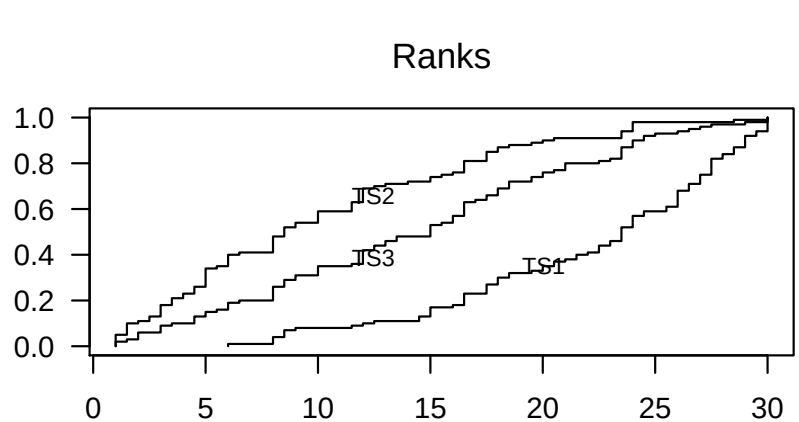
Graphical Representation

On a class of instances



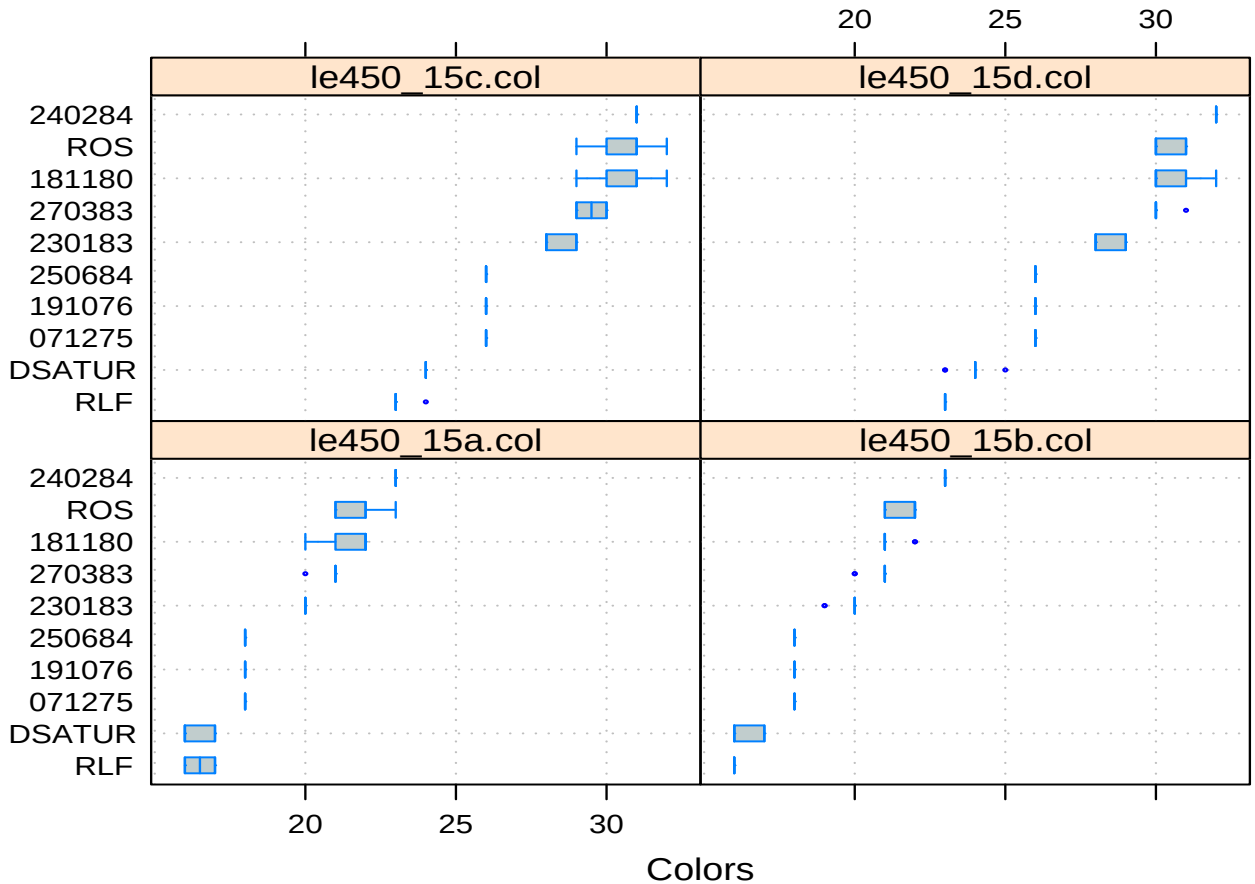






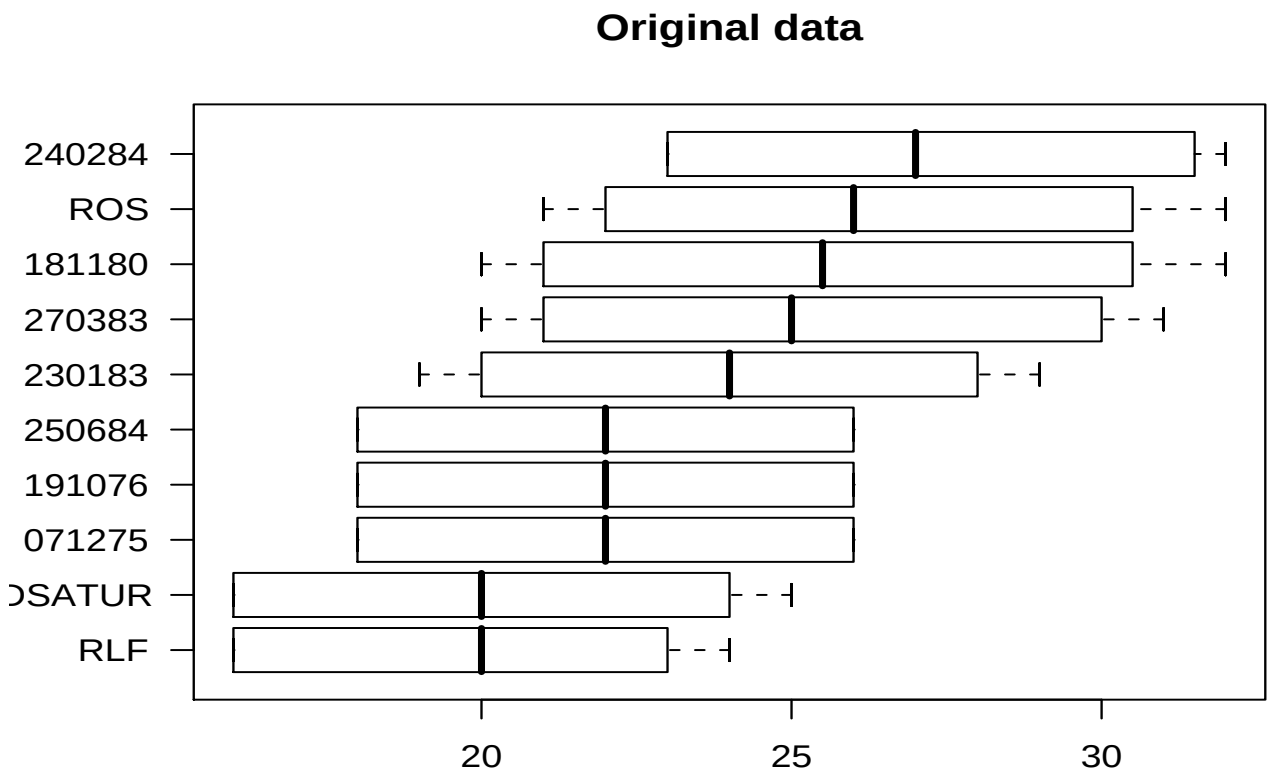
Examples

View of raw data within each instance



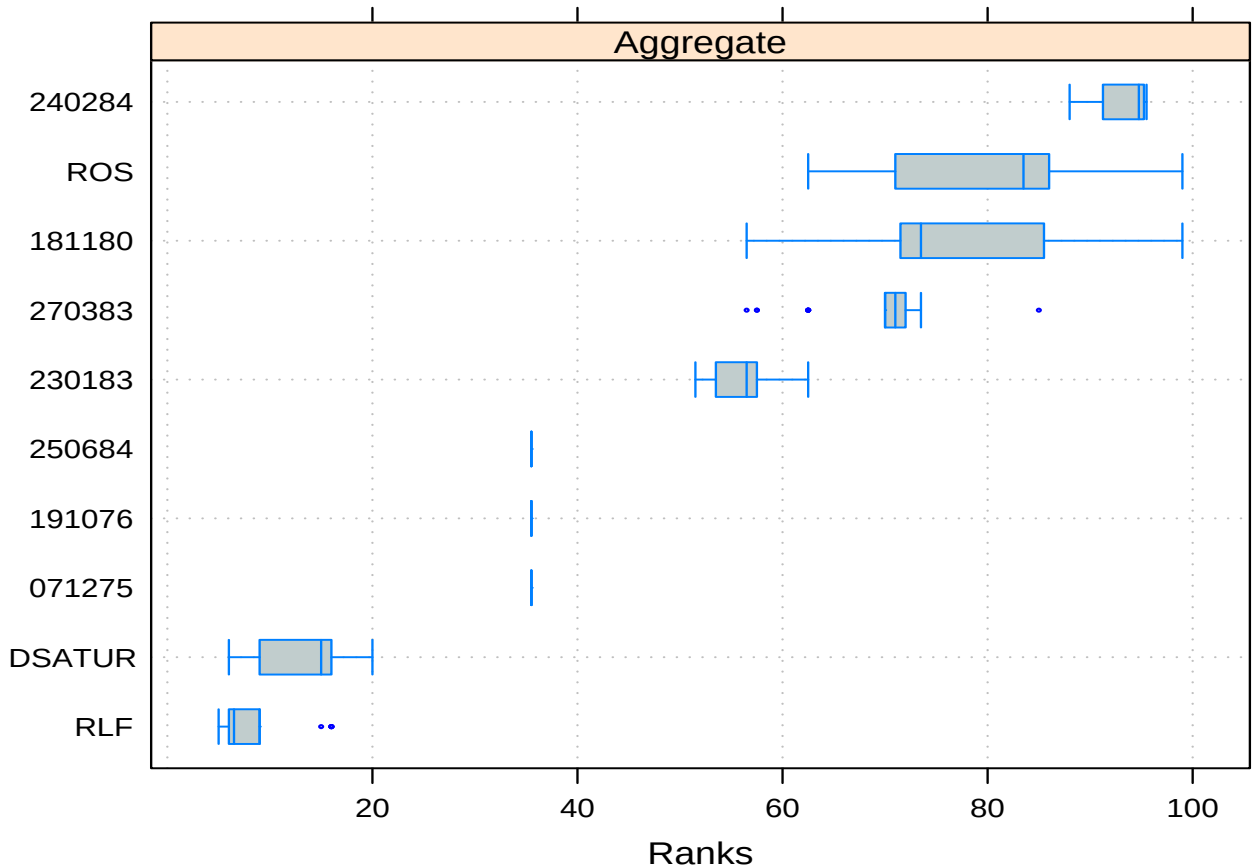
Examples

View of raw data aggregated for the 4 instances



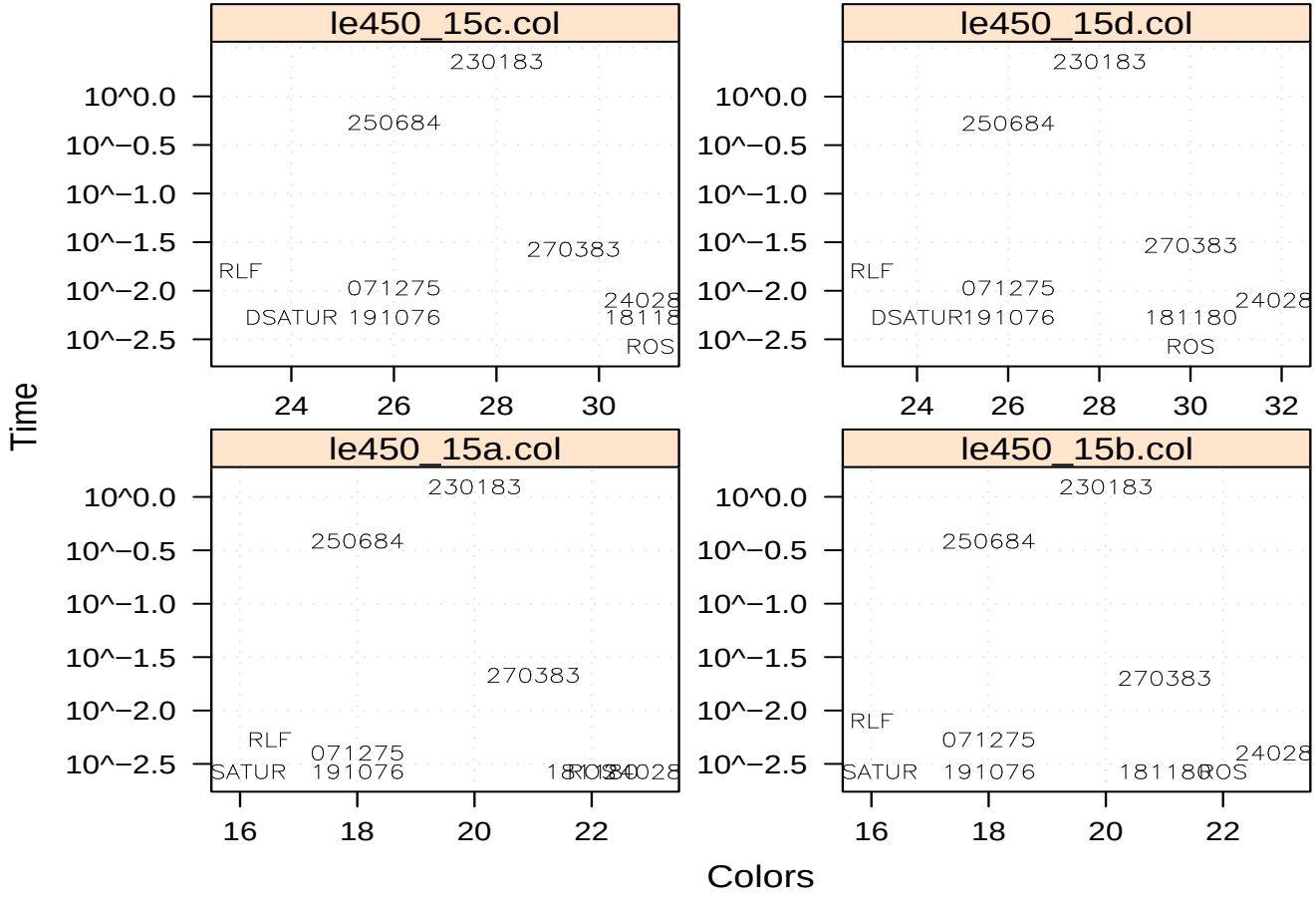
Examples

View of raw data ranked within instances
and aggregated for the 4 instances



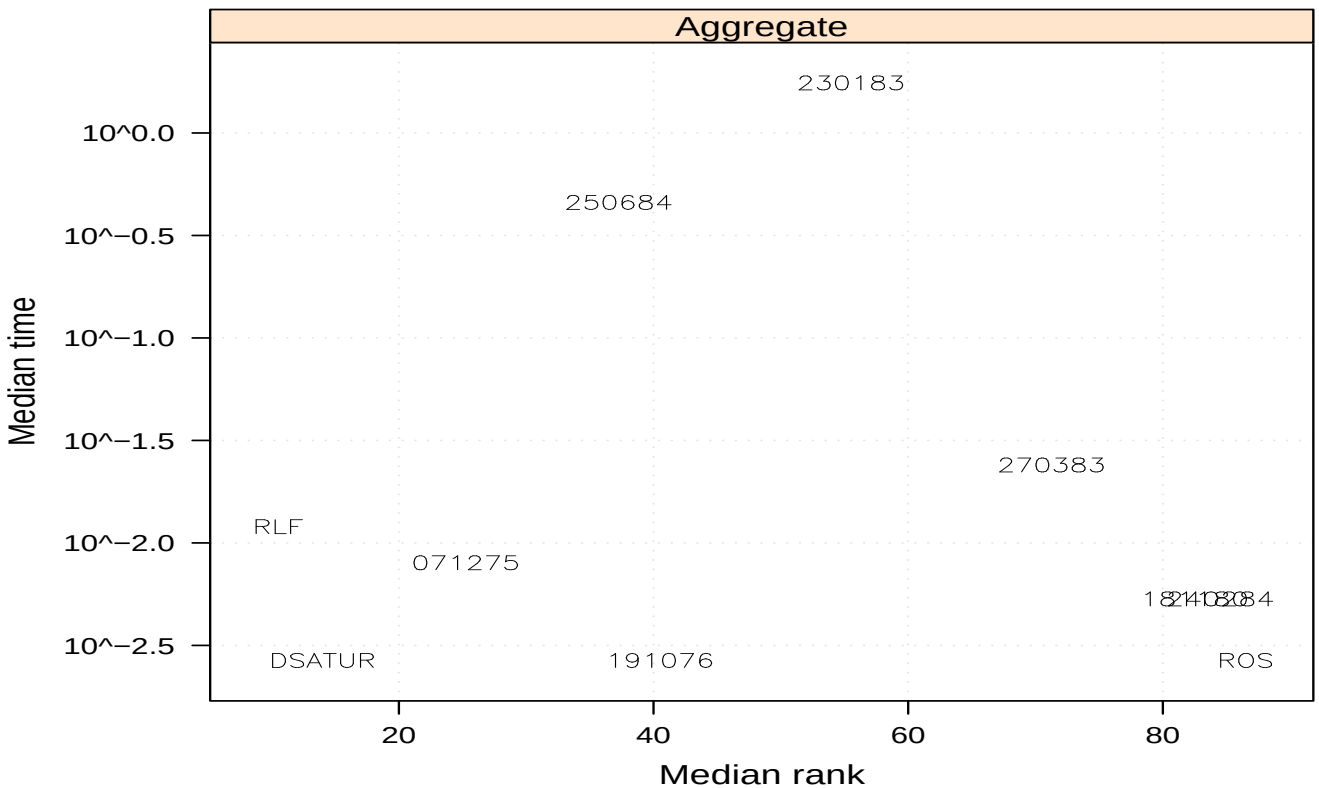
Examples

The trade off computation time *vs* sol quality. Raw data.



Examples

The trade off computation time *vs* sol quality.
Solution quality ranked within the instances and computation time in raw terms



Variance Reduction Techniques

[McGeoch 1992]

- Same instances
- Same pseudo random seed
- Common quantity for every random quantity that is positively correlated with the algorithms

Variance of the original performance will not vary but the variance of the difference will decrease because covariance =0

Subtract out a source of random noise if its expectation is known and it is positively correlated with outcome (eg, initial solution, cost of simple algorithm)

$$X' = X + (R - E[R])$$

Outline

Algorithm Configuration

- Which algorithm solves best our problem? (RRNM, SA, DE) (categorical)
- Which values should be assigned to the parameters of the algorithms? Eg, how many restarts of NM? Which temperature in SA? (numerical)
- How many times should we have random restart before chances to find better solutions become irrelevant? (numerical, integer)
- Which is the best way to generate initial solutions? (categorical) Theoretical motivated question: Which is the tradeoff point, where quasi random is not anymore helpful?
- Do instances that come from different applications of Least Median of Squares need different algorithm? (Instance families separation)
- ...

Organization of the Experiments

Questions:

- What (input, program) parameters to control?
- Which levels for each parameter?
- What kind of experimental design?
- How many sample points?
- How many trials per sample point?
- What to report?
- Sequential or one-shot trials?

Develop an experimental environment, run pilot tests

Work Done

- ANOVA
- Regression trees [Bartz-Beielstein and Markon, 2004]
- Racing algorithms [Birattari et al., 2002]
- Search approaches
  [Minton 1993, 1996, Cavazos & O'Boyle 2005],
  [Adenso-Diaz & Laguna 2006, Audet & Orban 2006][Hutter et al., 2007]
- Response surface models, DACE
  [Bartz-Beielstein, 2006; Ridge and Kudenko, 2007a,b]

Outline

Sources of Variance

- Treatment factors:
  - $A_1, A_2, \ldots, A_k$ algorithm factors: initial solution, temperature, ...
  - $B_1, B_2, \ldots, B_m$ instance factors: structural differences, application, size, hardness, ...
- Controllable nuisance factors:
  - $I_1, I_2, \ldots, I_n$ single instances
  - algorithm replication

$$\left\langle \begin{array}{c}\text{algorithm}\\ \text{factors}\end{array} \middle/ \begin{array}{c}\text{instance}\\ \text{factors}\end{array} \middle/ \begin{array}{c}\text{number of}\\ \text{instances}\end{array} \middle/ \begin{array}{c}\text{number of}\\ \text{runs}\end{array} \right\rangle$$

$\langle$-/-/1/$r\rangle$ $\langle$k/-/1/$r\rangle$ $\langle$-/m/n/1$\rangle$ $\langle$k/m/n/1$\rangle$
$\langle$-/-/n/1$\rangle$ $\langle$k/-/n/1$\rangle$ $\langle$k/-/n/$r\rangle$

The Random Effect Design

- **Factors**: $\langle$-/-/n/$r\rangle$
  - Instance: 10 instances randomly sampled from a class
  - Replicates five runs of RRNM on the 10 instances from the class
- **Response**:
  - Quality: solution cost or transformations thereof

$$Y_{il} = \mu + I_i + \varepsilon_{il},$$

where

- μ an overall mean,
- I_i a random effect of instance i, [i.i.d. $N(0, \sigma_\tau^2)$]
- ε_{il} a random error for replication l [i.i.d. $N(0, \sigma^2)$]

Random vs Blocking Factors

$$Y_{il} = \mu + I_i + \varepsilon_{il}$$

Random

I_i a random effect of instance i

$$\begin{array}{rcl} Y_{il}|I_i & \sim & N(\mu + I_i, \sigma^2 \quad\quad) \\ Y_{il} & \sim & N(\mu \quad\quad, \sigma^2 + \sigma_I^2) \end{array}$$

We draw conclusions on the entire population of levels

$\Downarrow$

corresponds to looking at $Pr(y)$

Blocking

τ_i the fixed effect of instance i

$$\begin{array}{rcl} Y_{il}|I_i & \sim & N(\mu + I_i, \sigma^2) \\ Y_{il} & \sim & N(\mu + I_i, \sigma^2) \end{array}$$

The results hold only for those levels tested

$\Downarrow$

corresponds to looking at $Pr(y|\pi)$

The Mixed Effects Design

- **Factors**: $\langle$k/-/n/$r\rangle$

  Algorithm: {RRNM,SA,DE}
  Instance: 10 instances randomly sampled from a class
  Replicates five runs per algorithm on the 10 instances from the class

- **Response**:

  Quality: solution cost or transformations thereof

$$Y_{ijl} = \mu + A_j + I_i + \gamma_{ij} + \varepsilon_{ijl}$$

- μ an overall mean,
- A_j a fixed effect of the algorithm j,
- I_i a random effect of instance i, [i.i.d. $N(0, \sigma^2_\tau)$]
- γ_{ij} a random interaction instance–algorithm, [i.i.d. $N(0, \sigma^2_\gamma)$]
- ε_{ijl} a random error for replication l of alg. j on inst. i [i.i.d. $N(0, \sigma^2)$]

Replicated or Unreplicated?

Which is the best design?

$$3 \text{ runs} \times 10 \text{ instances} = 30 \text{ experiments}$$
(replicated design) ⟨k/-/n/r⟩

OR

$$1 \text{ runs} \times 30 \text{ instances} = 30 \text{ experiments}$$
(unreplicated design) ⟨k/-/n/1⟩

If possible, ⟨k/-/n/1⟩ is better:

- it minimizes the variance of the estimates [Birattari, 2004]
- blocking and random design correspond mathematically

The Factorial Nested Design

- **Factors**: $\langle$-/m/n/$r\rangle$

  Instance Factors: Application = {Random, Dow Jones}
  Instance: four instances randomly sampled from a class
  Replicates 3 runs per algorithm on the 4 instances from the class

- **Response**:

  Quality: solution cost or transformations thereof

	Class 1 (Random)				**Class 2** (Dow Jones)			
Instances	1	2	3	4	5	6	7	8
Observations	Y_{111}	Y_{121}	Y_{131}	Y_{141}	Y_{251}	Y_{261}	Y_{271}	Y_{281}
	Y_{112}	Y_{122}	Y_{132}	Y_{142}	Y_{252}	Y_{262}	Y_{272}	Y_{282}
	Y_{113}	Y_{123}	Y_{133}	Y_{143}	Y_{253}	Y_{263}	Y_{273}	Y_{283}

The Factorial Nested Design

- **Factors**: $\langle$-/m/n/$r\rangle$

  Instance Factors: Application = {Random, Dow Jones}
  Instance: four instances randomly sampled from a class
  Replicates 3 runs per algorithm on the 4 instances from the class

- **Response**:

  Quality: solution cost or transformations thereof

$$Y_{ijl} = \mu + B_j + I_{i(j)} + \epsilon_{ijl}$$

- μ an overall mean,
- B_j a fixed effect of the feature j,
- $I_{i(j)}$ a random effect of the instance i nested in j
- ε_{ijl} a random error for replication l on inst. i nested in j

An Example for CAPM

Study on Random Restart Nelder-Mead for CAPM

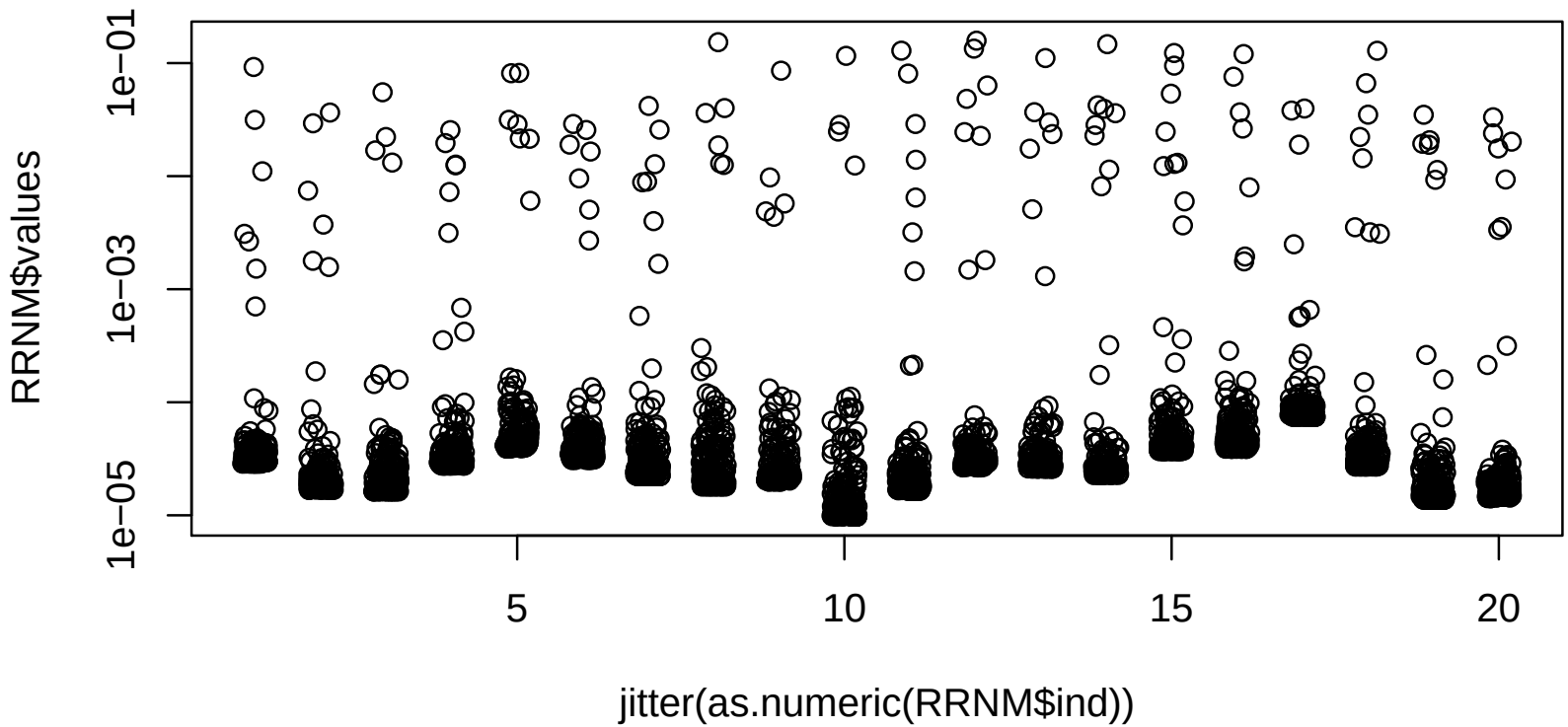
Factors:

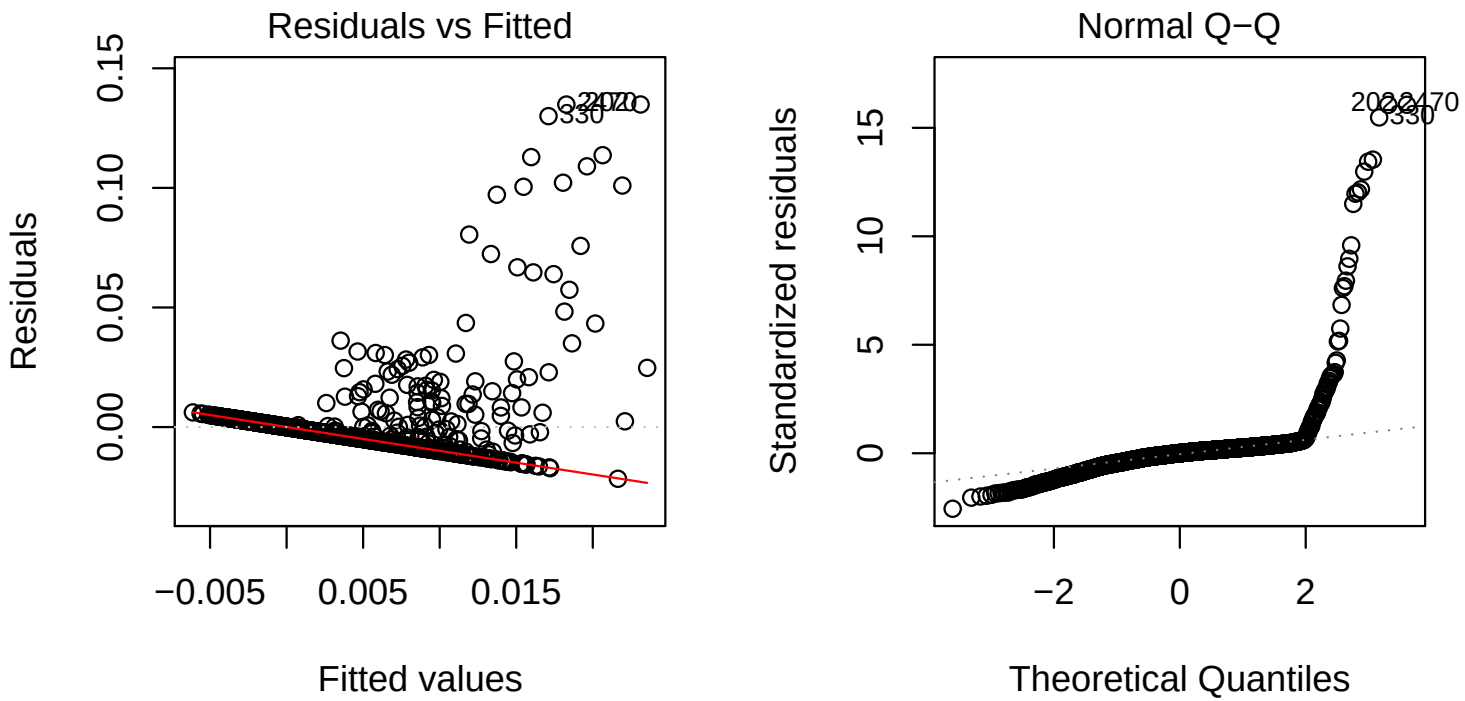
Factor	Type	Levels
initial.method	Categorical	{random, quasi-random}
max.reinforce	Integer	{1;3;5}
alpha	Real	{0.5;1;1.5}
beta	Real	{0;0.5;1}
gamma	Real	{1.5;2;2.5}

Instances: 20 randomly sampled from the Dow Jones application
Replicates: only one per instance

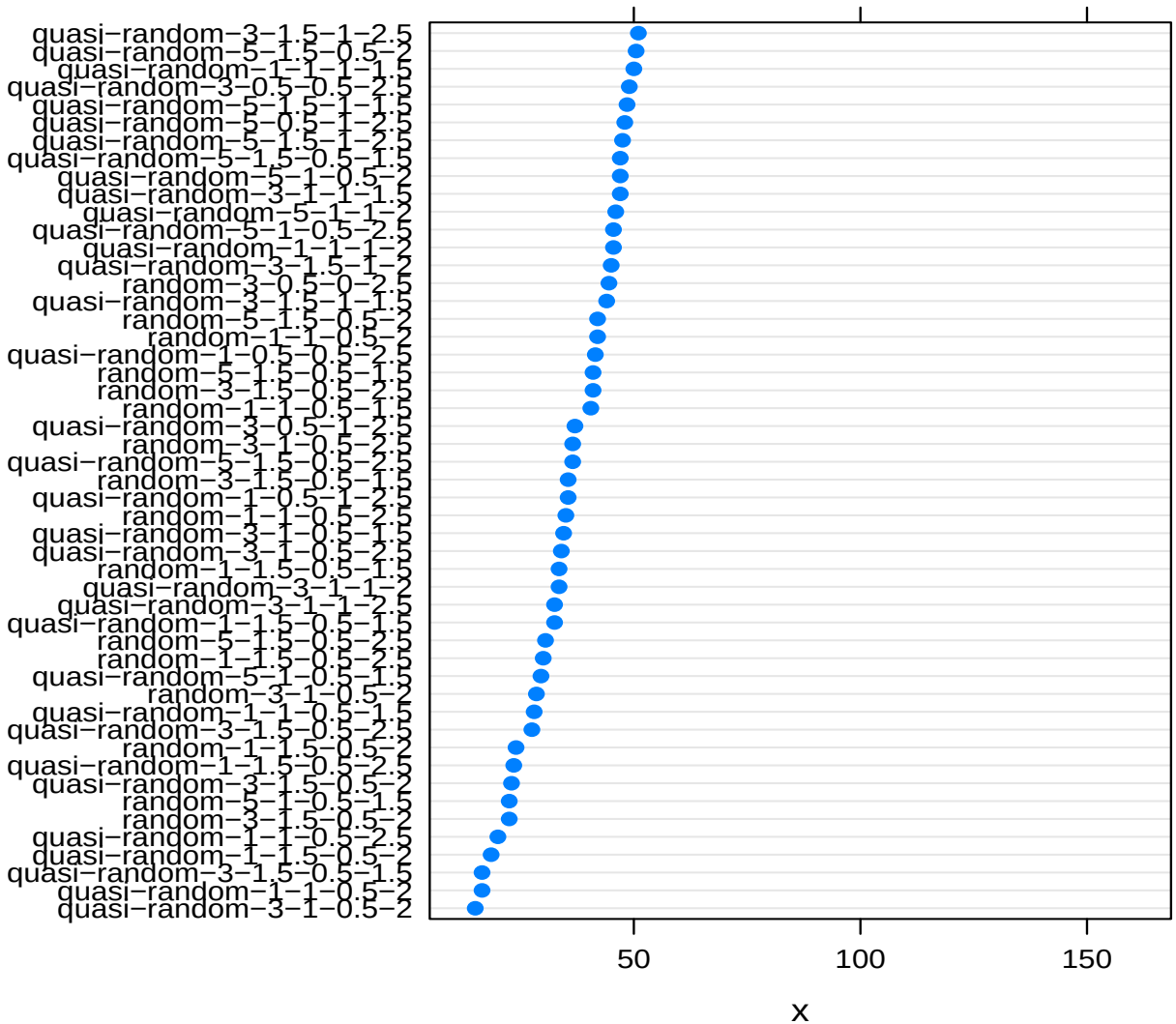
Response measures

- time is similar for all configurations because we stop after 500 random restart
- measure solution cost


RRNM$values
1e−01
1e−03
1e−05
5
10
15
20
jitter(as.numeric(RRNM$ind))



- Main problem is heteroschdasticity
- Possible transformations: ranks + likelihood based Box-Cox
- Only max.reinforce is not significant, all the rest is


quasi−random−3−1.5−1−2.5
quasi−random−5−1.5−0.5−2
quasi−random−1−1−1−1.5
quasi−random−3−0.5−0.5−2.5
quasi−random−5−1.5−1−1.5
quasi−random−5−0.5−1−2.5
quasi−random−5−1.5−1−2.5
quasi−random−5−1.5−0.5−1.5
quasi−random−5−1−0.5−2
quasi−random−3−1−1−1.5
quasi−random−5−1−1−2
quasi−random−5−1−0.5−2.5
quasi−random−1−1−1−2
quasi−random−3−1.5−1−2
random−3−0.5−0−2.5
quasi−random−3−1.5−1−1.5
random−5−1.5−0.5−2
random−1−1−0.5−2
quasi−random−1−0.5−0.5−2.5
random−5−1.5−0.5−1.5
random−3−1.5−0.5−2.5
random−1−1−0.5−1.5
quasi−random−3−0.5−1−2.5
random−3−1−0.5−2.5
quasi−random−5−1.5−0.5−2.5
random−3−1.5−0.5−1.5
quasi−random−1−0.5−1−2.5
random−1−1−0.5−2.5
quasi−random−3−1−0.5−1.5
quasi−random−3−1−0.5−2.5
random−1−1.5−0.5−1.5
quasi−random−3−1−1−2
quasi−random−3−1−1−2.5
quasi−random−1−1.5−0.5−1.5
random−5−1.5−0.5−2.5
random−1−1.5−0.5−2.5
quasi−random−5−1−0.5−1.5
random−3−1−0.5−2
quasi−random−1−1−0.5−1.5
quasi−random−3−1.5−0.5−2.5
random−1−1.5−0.5−2
quasi−random−1−1.5−0.5−2.5
quasi−random−3−1.5−0.5−2
random−5−1−0.5−1.5
random−3−1.5−0.5−2
quasi−random−1−1−0.5−2.5
quasi−random−1−1.5−0.5−2
quasi−random−3−1.5−0.5−1.5
quasi−random−1−1−0.5−2
quasi−random−3−1−0.5−2
50
100
150
x

Outline

Regression Trees

Recursive partitioning: Some history: AID, [Morgan and Sonquist, 1963], CHAID [Kass 1980], CART [Breiman, Friedman, Olshen, and Stone 1984] C4.5 [Quinlan 1993].

Conditional inference trees estimate a regression relationship by binary recursive partitioning in a conditional inference framework.

[Hothorn, Hornik, and Zeileis, 2006]

Step 1: Test the global null hypothesis of independence between any of the input variables and the response.
Stop if this hypothesis cannot be rejected.

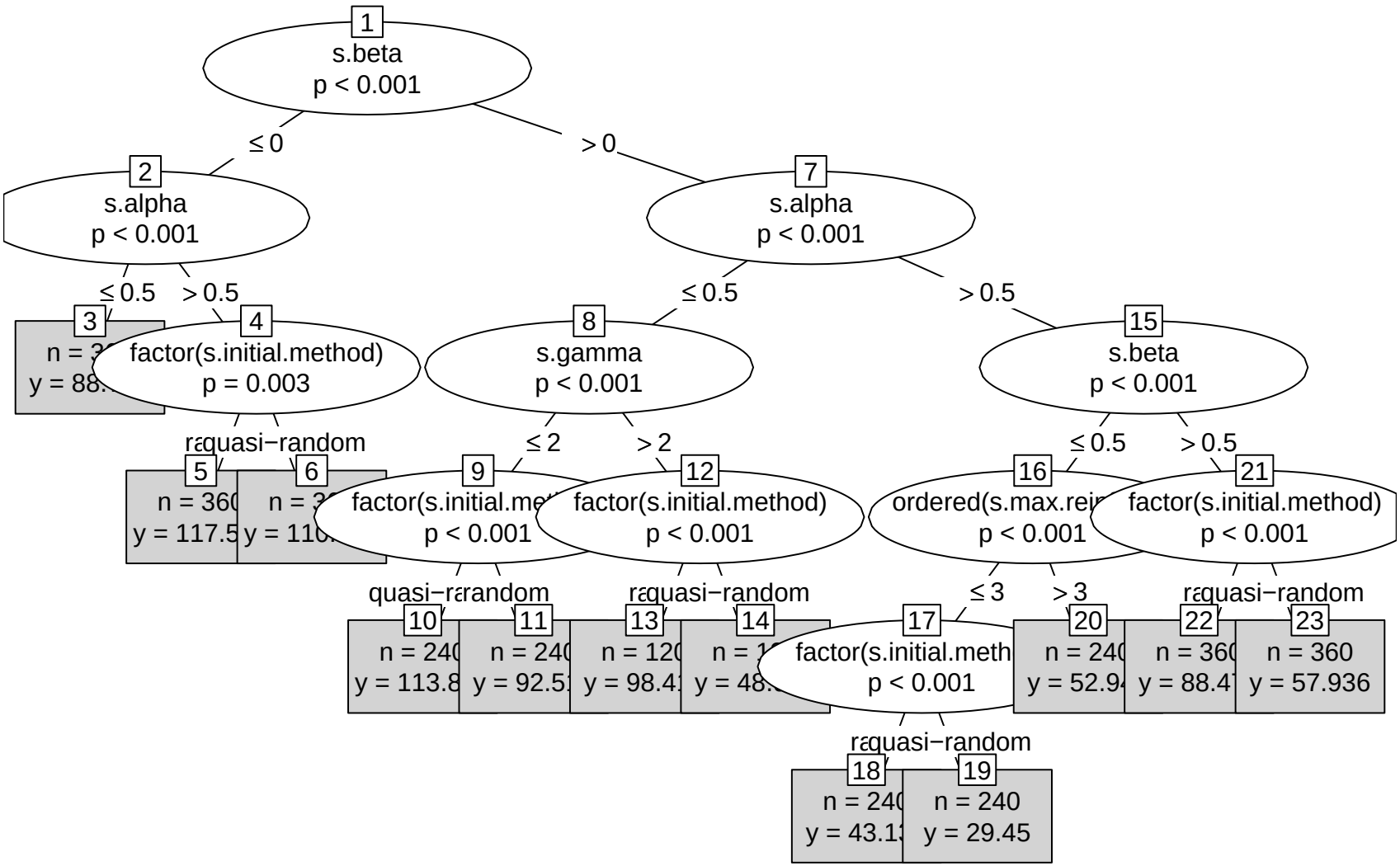
Otherwise test for the partial null hypothesis of a single input variable and the response.

Select the input variable with most important p-value

Step 2: Implement a binary split in the selected input variable.

Step 3: Recursively repeat steps 1) and 2).

Example: RRNM for CAPM



Outline

Racing Methods

- Idea from model selection problem in machine learning
- Sequential testing:
  configurations are discarded as soon as statistical evidence arises
- Based on full factorial design

Procedure `Race` [Birattari, 2005]: ⟨k/-/n/1⟩

repeat
  Randomly select an unseen instance
  Execute all candidates on the chosen instance
  Compute *all-pairwise comparison* statistical tests
  Drop all candidates that are significantly inferior to the best algorithm
until only one candidate left or no more unseen instances ;

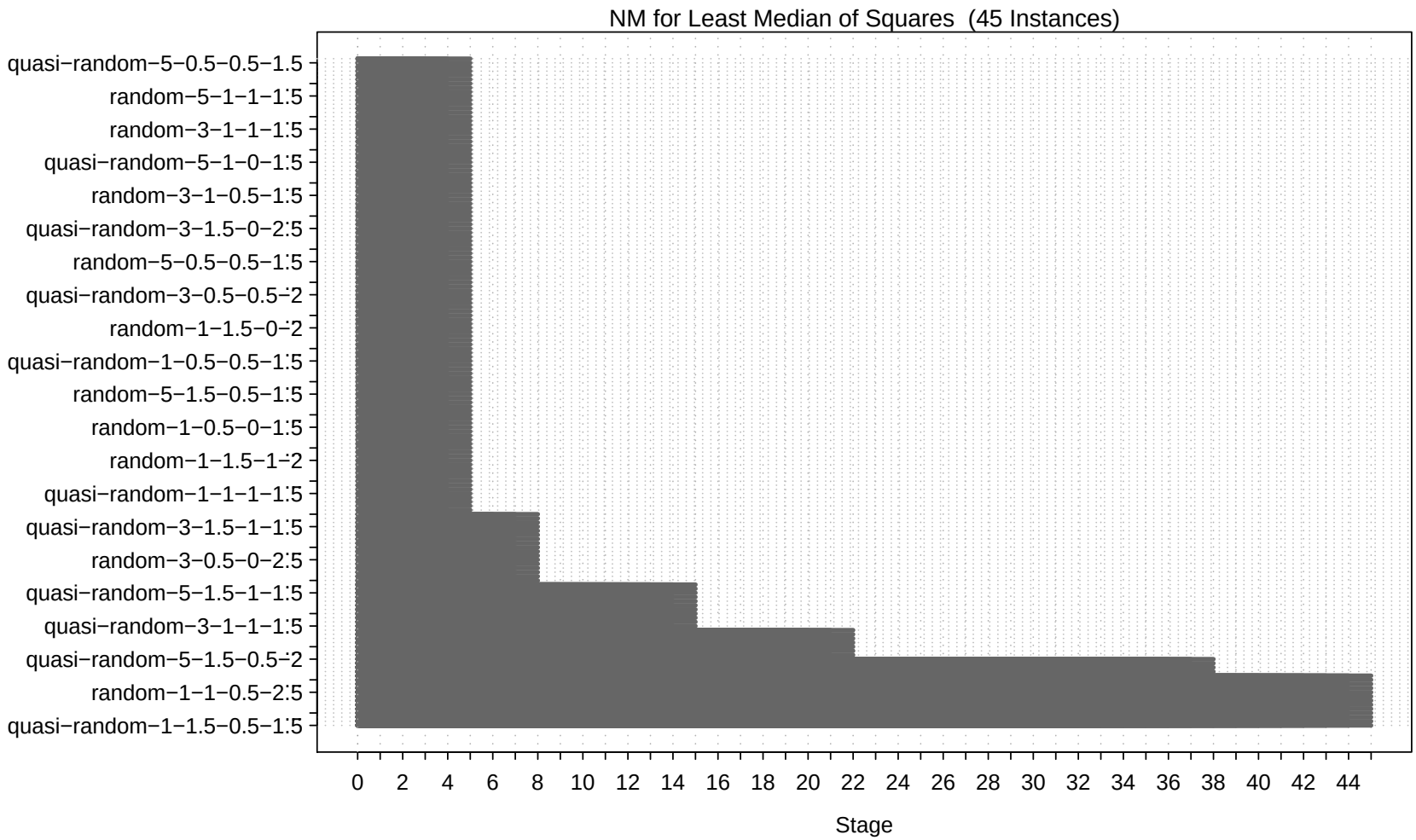
Statistical tests:

- t test, Friedman 2-ways analysis of variance (`F-Race`)
- all-pairwise comparisons ➨ p-value adjustment (Holm, Bonferroni)

Example: RRNM for CAPM

```
Race name.....................NM for Least Median of Squares
Number of candidates.....................................162
Number of available tasks.................................45
Max number of experiments................................3240
Statistical test.....................................Friedman test
Tasks seen before discarding..............................5
Initialization function...................................ok
Parallel Virtual Machine..................................no

x No test is performed.
- The test is performed and some candidates are discarded.
= The test is performed but no candidate is discarded.

+-+----------+----------+----------+----------+----------+
| |      Task|     Alive|      Best| Mean best|Exp so far|
+-+----------+----------+----------+----------+----------+
|x|         1|       162|        81| 2.869e-05|       162|
...
|x|         4|       162|       140| 2.887e-05|       648|
|-|         5|        52|       140| 3.109e-05|       810|
|=|         6|        52|        34| 3.892e-05|       862|
...
|=|        45|        13|        32|  4.55e-05|      1742|
+-+----------+----------+----------+----------+----------+

Selected candidate:         32 mean value:   4.55e-05
```

Application Example



Race Extension

Full factorial design is still costly ➡ Simple idea: random sampling design

Step 1: Sample N_{max} points in the parameter space according to a prior probability P_X (d-variate uniform distribution).

Step 2: Execute the race

Step 3: P_X becomes a sum of normal distributions centered around each N survivors with parameters: $\mu_s = (\mu_{s_1}, \ldots, \mu_{s_d})$ and $\sigma_s = (\sigma_{s_1}, \ldots, \sigma_{s_d})$
At each iteration t reduce the variance: $\sigma_{sk}^t = \sigma_{sk}^{t-1} (\frac{1}{N})^{\frac{1}{d}}$
Sample each of $N_{max} - N^s$ points from the parameter space:
a) select a d-variate normal distribution $N(\mu_s, \sigma_s)$ with probability

$$P_z = \frac{N^s - z + 1}{N^s(N^s + 1)/2}, \qquad z\text{is rank of } s$$

b) sample the point from this distribution

Initial conditions linked to parameters

$$\sigma^2_{sk} = \frac{max_k - min_k}{2}$$

Stopping conditions for intermediate races:

- when N_{min} $(= d)$ configurations remain
- when computational budget B is finished $(B = \frac{B_{tot}}{5})$
- I_{max} instances seen

Outline

The ParamILS Heuristic

The Tuning problem is a Mixed variables stochastic optimization problem

[Hutter, Hoos, and Stützle, 2007]

The space of parameters Θ is discretized and a combinatorial optimization problem solved by means of iterated local search

Procedure ParamILS
Choose initial parameter configuration $\theta \in \Theta$
Perform subsidiary local search from θ
while time left **do**
  $\theta' := \theta$ perform perturbation on θ
  perform subsidiary local search from θ

  based on acceptance criterion,
    keep θ or revert $\theta := \theta'$

  with probability P_R restart from a new θ from Θ

ParamILS

ParamILS components:

- Initialization: Pick a configuration $(\theta_1, \ldots, \theta_p) \in \Theta$ according to d-variate uniform distribution.
- Subsidiary local search: iterative first improvement, change one parameter in each step
- Perturbation: change s randomly chosen parameters
- Acceptance criterion: always select better local optimum

ParamILS

Evaluation of a parameter configuration θ:

- Sample N instances from given set (with repetitions)
- For each of the N instances:
  - Execute algorithm with configuration θ
  - Record scalar cost of the run (user-defined: e.g. run-time, solution quality, ...)
- Compute scalar statistic $c_N(\theta)$ of the N costs (user-defined: e.g. empirical mean, median, ...)

Note: N is a crucial parameter. In an enhanced version, $N(\theta)$ is increased for good configurations and decreased for bad ones at run-time.

Observation

- All algorithms solving these problems have parameters in their own and tuning them is paradoxical
- It is crucial finding methods that minimize the number of evaluations

Outline

Response Surface Method

[Kutner et al., 2005; Montgomery, 2005; Ridge and Kudenko, 2007b,a]

In optimizing a stochastic function
direct search methods, such as NM, SA, DE and ParamILS,

- are derivative free
- do not attempt to model

Response Surface Method (RSM) tries to build a model of the surface from the sampled data.

Procedure:

- Model the relation between most important algorithm parameters, instance characteristics and responses.
- Optimize the responses based on this relation

Two steps:

- screening
- response surface modelling

Step 1: Screening

Used to identify the parameters that are not relevant to include in the RSM

- Fractional factorial design
- Collect data
- Fit model: first only main effects, then add interactions, then quadratic terms, continue until resolution allows, compare terms with t-test.
- Diagnostic + transformations
  Method by [Box and Cox, 1964] to decide the best transformation on the basis of likelihood function
- Rank factor effect coefficients and assess significance

Fractional Factorial Designs

ANOVA model for three factors:

$$Y_i = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \beta_3 X_{i3} + \beta_{12} X_{i12} + \beta_{13} X_{i13} + \beta_{23} X_{i23} + \beta_{123} X_{i123} + \epsilon_i$$

- Study factors at only two levels ➡ 2^k designs

numerical real, numerical integer, categorical } encoded as $-1, 1$

Treat.	X1	X2	X3
1	−1	−1	−1
2	1	−1	−1
3	−1	1	−1
4	1	1	−1
5	−1	−1	1
6	1	−1	1
7	−1	1	1
8	1	1	1

- Single replication per design point
- High order interactions are likely to be of little consequence ➡ confound with each other

Fractional Factorial Designs

$$Y_i = \beta_0 X_{i0} + \beta_1 X_{i1} + \beta_2 X_{i2} + \beta_3 X_{i3} + \beta_{12} X_{i12} + \beta_{13} X_{i13} + \beta_{23} X_{i23} + \beta_{123} X_{i123} + \epsilon_i$$

Treat.	X0	X1	X2	X3	X12	X13	X23	X123
1	1	−1	−1	−1	1	1	1	−1
2	1	1	−1	−1	−1	−1	1	1
3	1	−1	1	−1	−1	1	−1	1
4	1	1	1	−1	1	−1	−1	−1
5	1	−1	−1	1	1	−1	−1	1
6	1	1	−1	1	−1	1	−1	−1
7	1	−1	1	1	−1	−1	1	−1
8	1	1	1	1	1	1	1	1

- 2^{k-f}, k factors, f fraction
- 2^{3-1} if X_0 confounded with X_{123} (half-fraction design) but also $X_1 = X_{23}$, $X_2 = X_{13}$, $X_3 = X_{12}$
- 2^{3-2} if X_0 confounded with X_{23}

Fractional Factorial Designs

Resolution is the number of factors involved in the lowest-order effect in the defining relation

Example:

$R = \mathrm{V} \Rightarrow 2_V^{5-1} \Rightarrow X_0 = X_{12345}$

$R = \mathrm{III} \Rightarrow 2_{III}^{6-2} \Rightarrow X_0 = X_{1235} = X_{123} = X_{456}$

$R \geq \mathrm{III}$ in order to avoid confounding of main effects

It is not so simple to identify defining relation with the maximum resolution, hence they are catalogued

A design can be augmented by folding over, that is, reversing all signs

Example: DE for CAPM

Differential Evolution for CAPM

- Termination condition: Number of idle iterations
- Factors:

	Factor	Type	Low (−)	High (−)
NP	Number of population members	Int	20	50
F	weighting factor	Real	0	2
CR	Crossover probability from interval	Real	0	1
initial	An initial population	Cat.	Uniform	Quasi MC
strategy	Defines the DE variant used in mutation	Cat.	rand	best
idle iter	Number of idle iteration before terminating	Int.	10	30

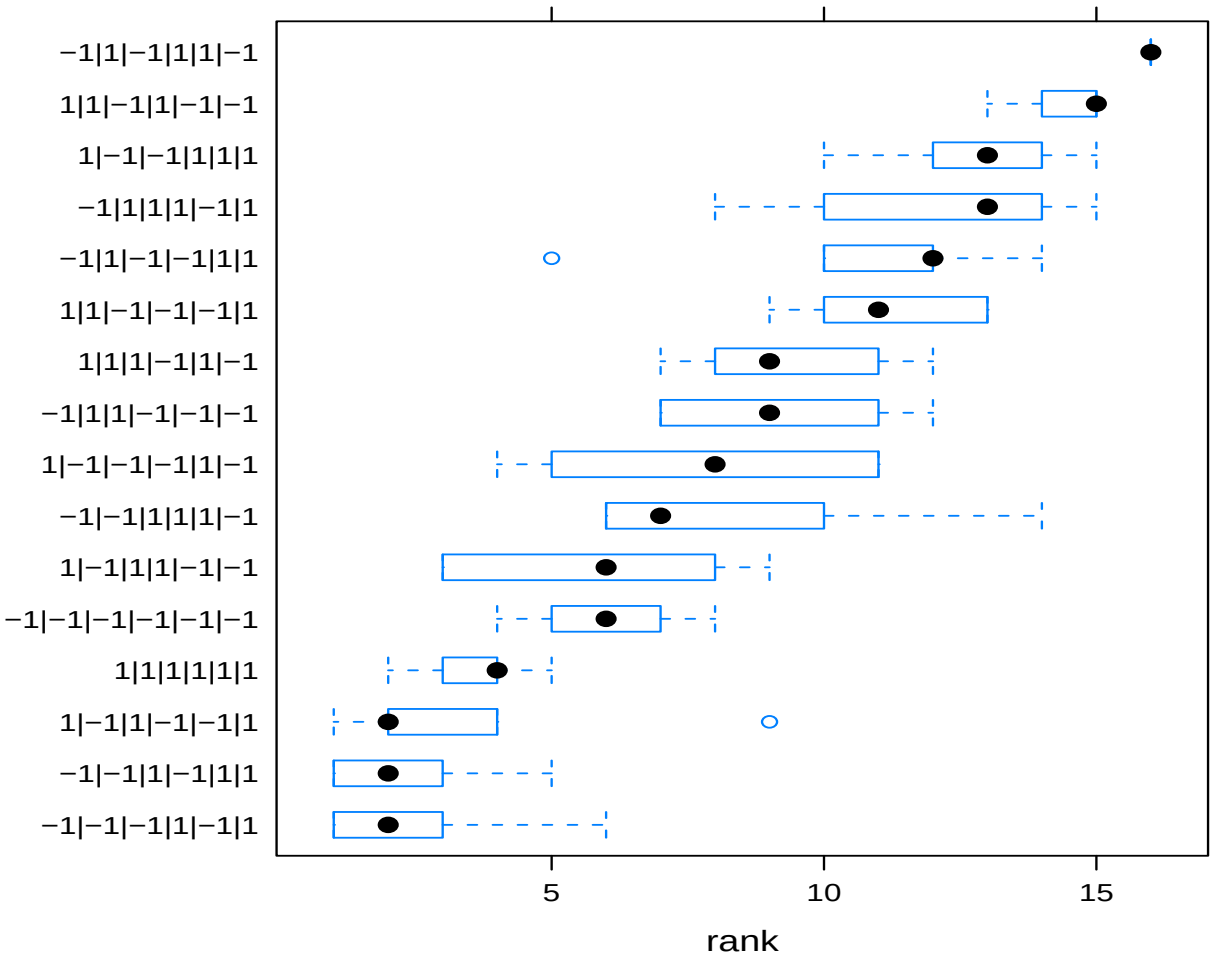
- Performance measures:
  - computational cost: number of function evaluations
  - quality: solution cost
- Blocking on 5 instances ➡ design replicates ➡ $2^6 \cdot 5 = 320$

Fractional Design: $2^{6-2}_{IV} \cdot 5 = 80$
main effects and second order interactions not confounded.

Example: DE for CAPM

	instance	NP	F	CR	initial	strategy	idleiter	value	time	nfeval
1	1	-1	-1	-1	-1	-1	-1	5.358566e-05	0.216	440
2	1	1	-1	-1	-1	1	-1	5.564804e-05	0.448	880
3	1	-1	1	-1	-1	1	1	6.803661e-05	0.660	1240
4	1	1	1	-1	-1	-1	1	6.227293e-05	1.308	2480
5	1	-1	-1	1	-1	1	1	4.993460e-05	0.652	1240
6	1	1	-1	1	-1	-1	1	4.993460e-05	1.305	2480
7	1	-1	1	1	-1	-1	-1	5.869048e-05	0.228	440
8	1	1	1	1	-1	1	-1	6.694168e-05	0.448	880
9	1	-1	-1	-1	1	-1	1	5.697797e-05	0.676	1240
10	1	1	-1	-1	1	1	1	7.267454e-05	1.308	2480
11	1	-1	1	-1	1	1	-1	2.325979e-04	0.220	440
12	1	1	1	-1	1	-1	-1	9.098808e-05	0.452	880
13	1	-1	-1	1	1	1	-1	8.323734e-05	0.228	440
14	1	1	-1	1	1	-1	-1	6.015744e-05	0.460	880
15	1	-1	1	1	1	-1	1	6.244267e-05	0.668	1240
16	1	1	1	1	1	1	1	5.348372e-05	1.352	2480

Example: DE for CAPM



Example: DE for CAPM

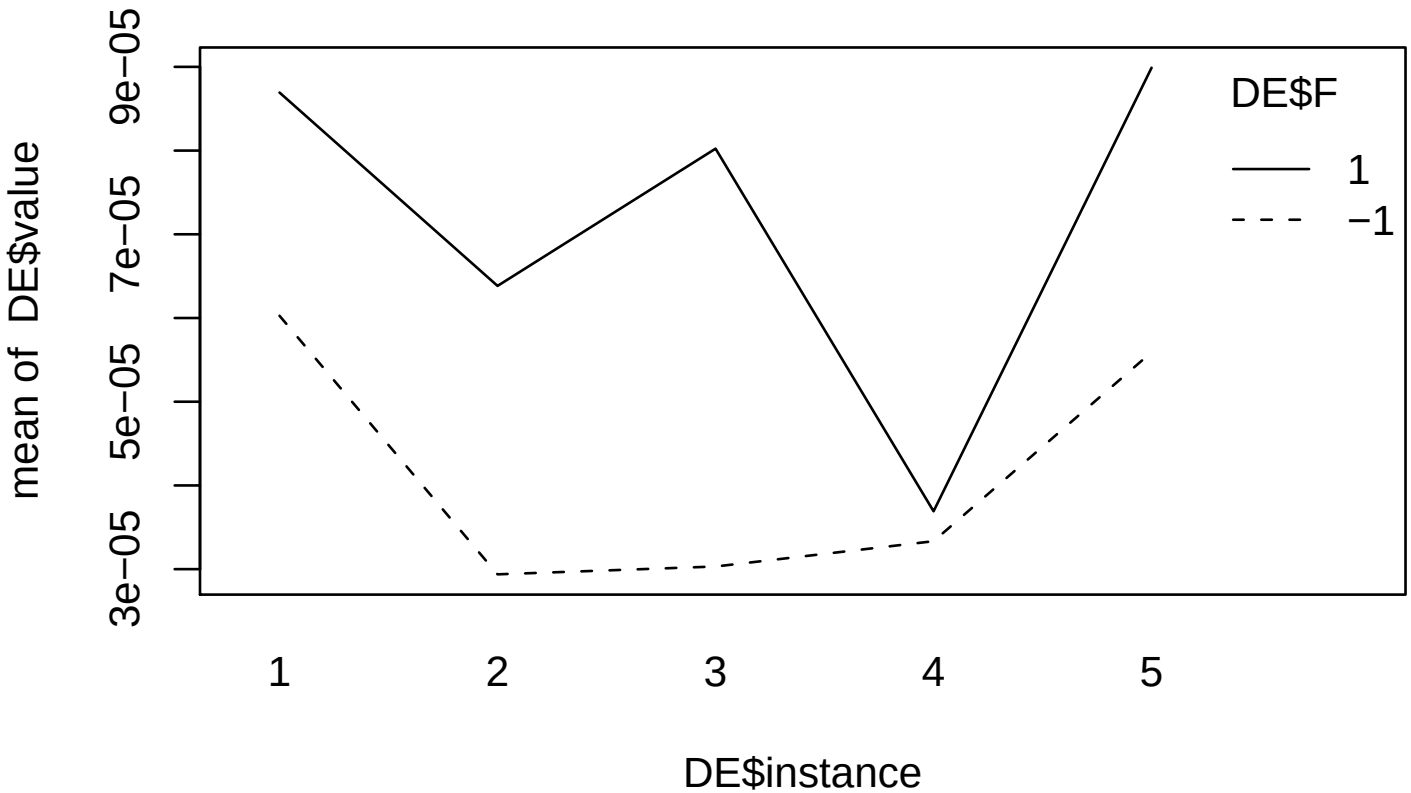
```
Call:
lm(formula = (rank^(1.2) - 1)/1.2 ~ (NP + F + CR + ini-
tial +
    strategy + idleiter + instance)^2 - 1, data = DE)

Residuals:
    Min      1Q  Median      3Q     Max
-10.277  -1.959   1.056   6.423  13.979

Coefficients: (8 not defined because of singularities)
                  Estimate Std. Error t value Pr(>|t|)
NP                -1.32447    1.76772  -0.749   0.4566
F                  3.40635    1.76772   1.927   0.0587 .
CR                -2.21180    1.76772  -1.251   0.2157
initial            2.47629    1.76772   1.401   0.1664
strategy           1.47545    1.76772   0.835   0.4072
idleiter          -1.81289    1.76772  -1.026   0.3092
instance           2.85013    0.22727  12.541   <2e-
16 ***
NP:F              -1.84492    0.75376  -2.448   0.0173 *
NP:CR             -1.92013    0.75376  -2.547   0.0134 *
NP:initial        -0.62881    0.75376  -0.834   0.4075
NP:strategy       -0.96685    0.75376  -1.283   0.2045
NP:idleiter        0.54652    0.75376   0.725   0.4712
NP:instance        0.46387    0.53299   0.870   0.3876
F:initial         -0.29205    0.75376  -0.387   0.6998
F:idleiter        -0.61857    0.75376  -0.821   0.4151
F:instance         0.01824    0.53299   0.034   0.9728
CR:instance       -0.12302    0.53299  -0.231   0.8182
initial:instance  -0.29898    0.53299  -0.561   0.5769
strategy:instance -0.28582    0.53299  -0.536   0.5938
idleiter:instance  0.05713    0.53299   0.107   0.9150
--
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1 ' ' 1
```

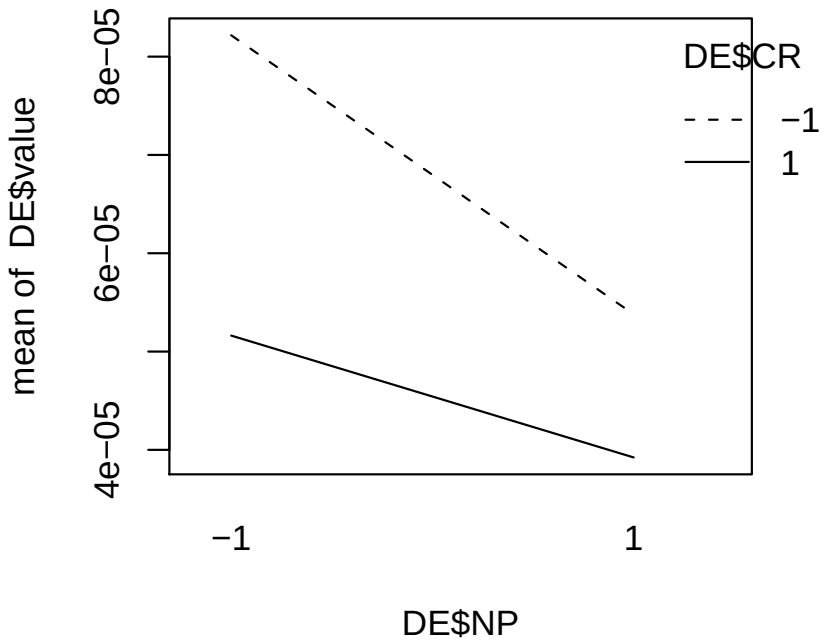
```
Call:
lm(formula = (nfeval^2 - 1)/2 ~ (NP + F + CR + initial + strategy
    idleiter + instance)^2 - 1, data = DE)

Residuals:
    Min      1Q  Median      3Q     Max
-393454  -98364  196727  491818  786909

Coefficients: (8 not defined because of singularities)
                   Estimate Std. Error   t value Pr(>|t|)
NP                6.492e+05  1.397e+05     4.648 1.89e-05 ***
F                 1.661e-12  1.397e+05  1.19e-17        1
CR               -1.624e-10  1.397e+05 -1.16e-15        1
initial           2.584e-11  1.397e+05  1.85e-16        1
strategy         -9.993e-11  1.397e+05 -7.15e-16        1
idleiter          8.400e+05  1.397e+05     6.014 1.17e-07 ***
instance          2.951e+05  1.796e+04    16.432  < 2e-16 ***
NP:F             -8.736e-12  5.956e+04 -1.47e-16        1
NP:CR             2.430e-11  5.956e+04  4.08e-16        1
NP:initial        1.737e-11  5.956e+04  2.92e-16        1
NP:strategy       1.603e-11  5.956e+04  2.69e-16        1
NP:idleiter       5.040e+05  5.956e+04     8.462 8.02e-12 ***
NP:instance       8.712e-11  4.212e+04  2.07e-15        1
F:initial        -1.663e-11  5.956e+04 -2.79e-16        1
F:idleiter        3.122e-11  5.956e+04  5.24e-16        1
F:instance       -5.101e-12  4.212e+04 -1.21e-16        1
CR:instance       5.035e-11  4.212e+04  1.20e-15        1
initial:instance -2.903e-12  4.212e+04 -6.89e-17        1
strategy:instance 3.272e-11  4.212e+04  7.77e-16        1
idleiter:instance 7.097e-11  4.212e+04  1.69e-15        1
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1 ' ' 1
```

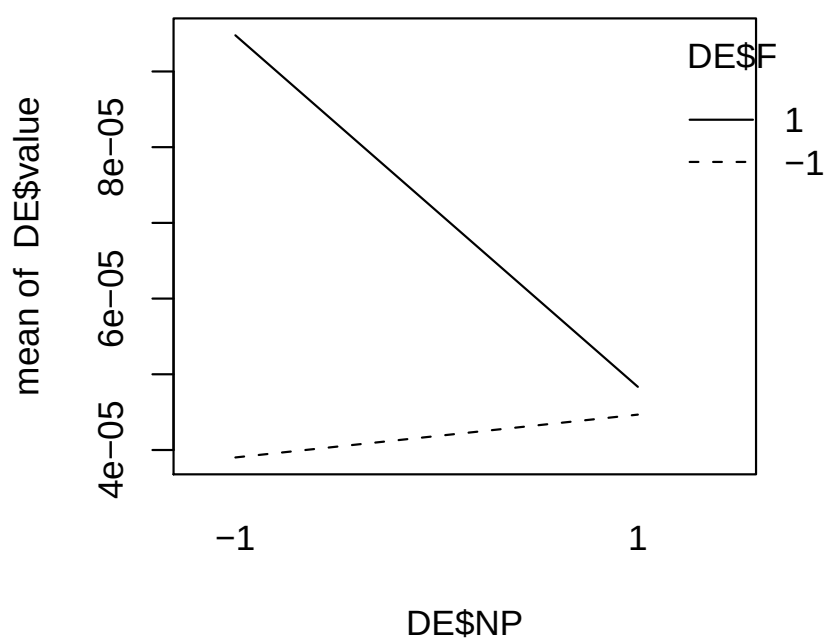
Example: DE for CAPM

Factor	Estimate effect cost effect	F-test	Estimate time effect	F-test
F	3.40635	.	1.661e-12	
CR	-2.21180		-1.624e-10	
initial	2.47629		2.584e-11	
idleiter	-1.81289		8.400e+05	***
strategy	1.47545		-9.993e-11	
NP	-1.32447		6.492e+05	***

However, screening ignore possible curvatures ➨ augment design by replications at the center points
If lack of fit then there is curvature in one or more factors ➨ more experimentations is needed


mean of DE$value
3e−05
5e−05
7e−05
9e−05
1
2
3
4
5
DE$instance
DE$F
1
−1


mean of DE$value
4e−05
6e−05
8e−05
−1
1
DE$NP
DE$CR
−1
1


mean of DE$value
4e−05
6e−05
8e−05
−1
1
DE$NP
DE$F
1
−1

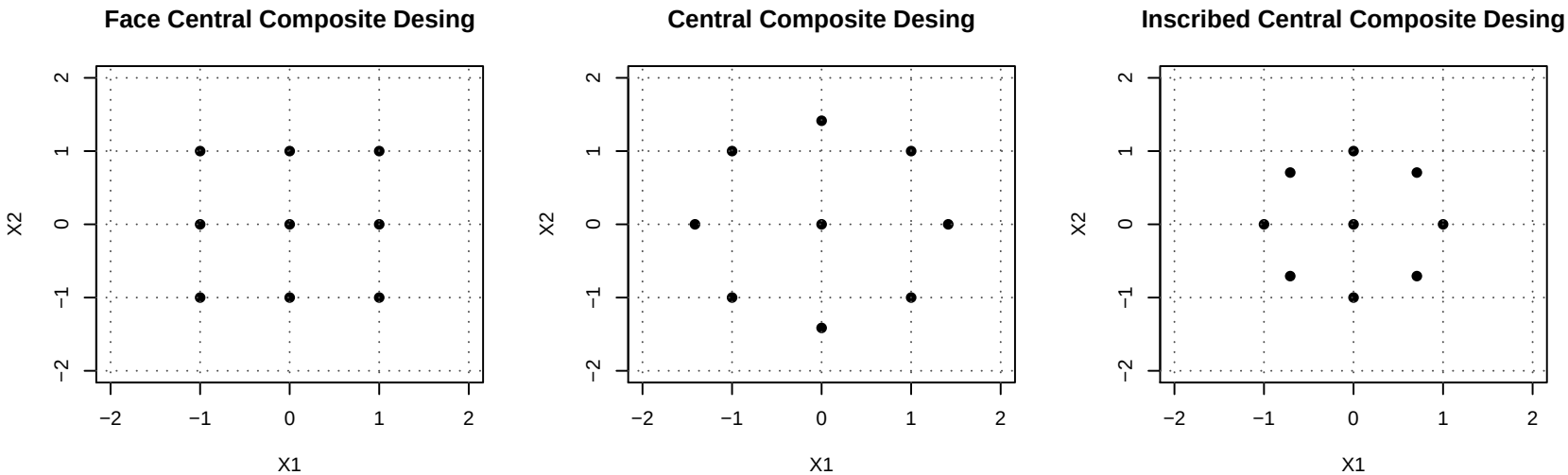
Step 2: Response surface modelling

- considers only quantitative factors ➡ repeat analysis for all categorical factors
- levels X_j of the jth factor are coded as:

$$X_j = \frac{\text{actual level} - \frac{\text{high level}+\text{low level}}{2}}{\frac{\text{high level}-\text{low level}}{2}}$$

Response Surface Designs

Designs for estimating second-order term response surface models:
Rotatability: equal precision at any distance from the center point.
($\sigma^2\{Y_h\}$ is the same at any X_h)



number of experiments $= 2^{k-f}\, n_c$ corner points $+ kn_s$ star points $+ n_0$

Decide n_c, n_s, n_o considering power and computational cost

[Lenth, R. V. (2006). Java Applets for Power and Sample Size (Computer software). `http://www.stat.uiowa.edu/~rlenth/Power`.]

Analysis

Analysis of response surface experiments

- estimate response function by general linear regression for each response variable. Hierarchical approach, backward elimination.
- interpret the model by visualization
  3D surface, contour plots, conditional effects plots, overlay contour plots
- identification of optimum operating conditions (or sequential search for optimum conditions)
  - desirability function $d_i(Y_i) : \mathbf{R} \mapsto [0, 1]$:

$$d_i(Y_i) = \begin{cases} 1 & \text{if } \widehat{Y}_i(x) < T \text{ (target value)} \\ \frac{\widehat{Y}_i(x) - U_i}{T_i - U_i} & \text{if } T_i \leq \widehat{Y}_i(x) \leq U_i \\ 0 & \text{if } \widehat{Y}_i(x) > U_i \end{cases}$$

  - minimize $\left(\prod_{i=1}^{k} d_i\right)^{1/k}$

Example: SA for CAPM

SA for CAPM

	Factor	Low (−)	High (−)
Eval	Max number of evaluations	10000	30000
Temp	Starting temperature for the cooling schedule	5	15
Tmax	Function evaluations at each temperature	50	150

- We use an inscribed central composite design with 4 replicates at the center ➡ 18 points
- 10 replicates for each of the 18 points blocking on 10 different instances.

Example: SA for CAPM

The Design in Encoded Variables (internal central composite design)

	X1	X2	X3
1	-0.7071068	-0.7071068	-0.7071068
2	0.7071068	-0.7071068	-0.7071068
3	-0.7071068	0.7071068	-0.7071068
4	0.7071068	0.7071068	-0.7071068
5	-0.7071068	-0.7071068	0.7071068
6	0.7071068	-0.7071068	0.7071068
7	-0.7071068	0.7071068	0.7071068
8	0.7071068	0.7071068	0.7071068
9	-1.0000000	0.0000000	0.0000000
10	1.0000000	0.0000000	0.0000000
11	0.0000000	-1.0000000	0.0000000
12	0.0000000	1.0000000	0.0000000
13	0.0000000	0.0000000	-1.0000000
14	0.0000000	0.0000000	1.0000000
15	0.0000000	0.0000000	0.0000000
16	0.0000000	0.0000000	0.0000000
17	0.0000000	0.0000000	0.0000000
18	0.0000000	0.0000000	0.0000000

Example: SA for CAPM

```
> sa.q <- stepAIC(lm(scale ~ ((Eval * Temp * Tmax) + I(Eval^2) +
+     I(Eval^3) + I(Temp^2) + I(Temp^3) + I(Tmax^2) + I(Tmax^3)),
+     data = SA), trace = FALSE)
> sa.q$anova

Stepwise Model Path
Analysis of Deviance Table

Initial Model:
scale ~ ((Eval * Temp * Tmax) + I(Eval^2) + I(Eval^3) + I(Temp^2) +
    I(Temp^3) + I(Tmax^2) + I(Tmax^3))

Final Model:
scale ~ Temp + I(Eval^2) + I(Temp^3) + I(Tmax^2)


               Step Df   Deviance Resid. Df Resid. Dev        AIC
1                                       166   149.6135  -5.282312
2       - I(Temp^2)  1 0.01157123       167   149.6250  -7.268391
3       - I(Tmax^3)  1 0.49203977       168   150.1171  -8.677435
4       - I(Eval^3)  1 0.97771081       169   151.0948  -9.508898
5  - Eval:Temp:Tmax  1 1.36868574       170   152.4635  -9.885717
6       - Temp:Tmax  1 0.21569471       171   152.6792 -11.631245
7       - Eval:Tmax  1 0.34530754       172   153.0245 -13.224607
8           - Tmax  1 1.09116851       173   154.1157 -13.945639
9       - Eval:Temp  1 1.17697426       174   155.2926 -14.576210
10          - Eval  1 0.53324991       175   155.8259 -15.959178
```

Example: SA for CAPM

```
> sa.t <- stepAIC(lm(time ~ ((Eval * Temp * Tmax) + I(Eval^2) +
+     I(Eval^3) + I(Temp^2) + I(Temp^3) + I(Tmax^2) + I(Tmax^3)),
+     data = SA), trace = FALSE)
> sa.t$anova

Stepwise Model Path
Analysis of Deviance Table

Initial Model:
time ~ ((Eval * Temp * Tmax) + I(Eval^2) + I(Eval^3) + I(Temp^2) +
    I(Temp^3) + I(Tmax^2) + I(Tmax^3))

Final Model:
time ~ Eval + I(Eval^2) + I(Tmax^2)


               Step Df     Deviance Resid. Df Resid. Dev       AIC
1                                         166   5.033365 -615.8363
2  - Eval:Temp:Tmax  1 0.0007938000       167   5.034159 -617.8079
3       - I(Temp^3)  1 0.0012386700       168   5.035397 -619.7636
4       - I(Tmax^3)  1 0.0020043172       169   5.037402 -621.6920
5       - Eval:Tmax  1 0.0020402000       170   5.039442 -623.6191
6       - I(Eval^3)  1 0.0062009141       171   5.045643 -625.3977
7       - Temp:Tmax  1 0.0062658000       172   5.051909 -627.1743
8           - Tmax   1 0.0005494828       173   5.052458 -629.1548
9       - Eval:Temp  1 0.0071442000       174   5.059602 -630.9004
10          - Temp   1 0.0001133300       175   5.059716 -632.8964
11      - I(Temp^2)  1 0.0137637556       176   5.073479 -634.4074
```

Example: SA for CAPM

Quality:

```
(Intercept)        Temp   I(Eval^2)   I(Temp^3)   I(Tmax^2)
 -0.3318884  -0.7960063   0.4793772   1.0889321   0.5162880
```

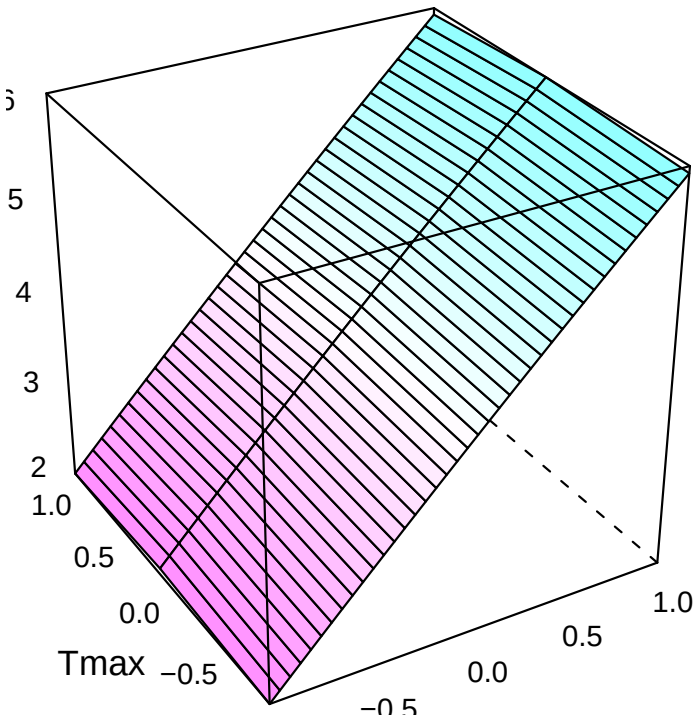
Computation Time:

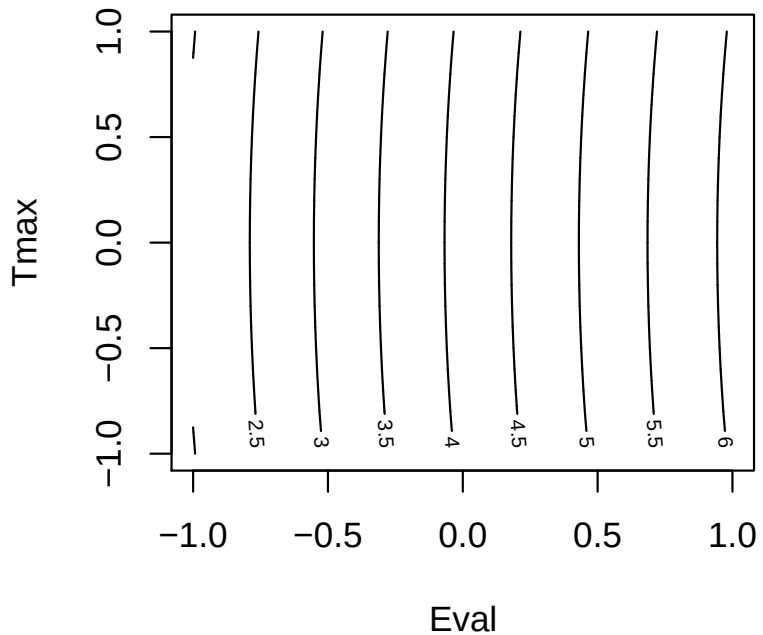
```
(Intercept)        Eval   I(Eval^2)   I(Tmax^2)
 4.13770000  2.02807697 -0.05713333 -0.06833333
```

Desirability function approach:

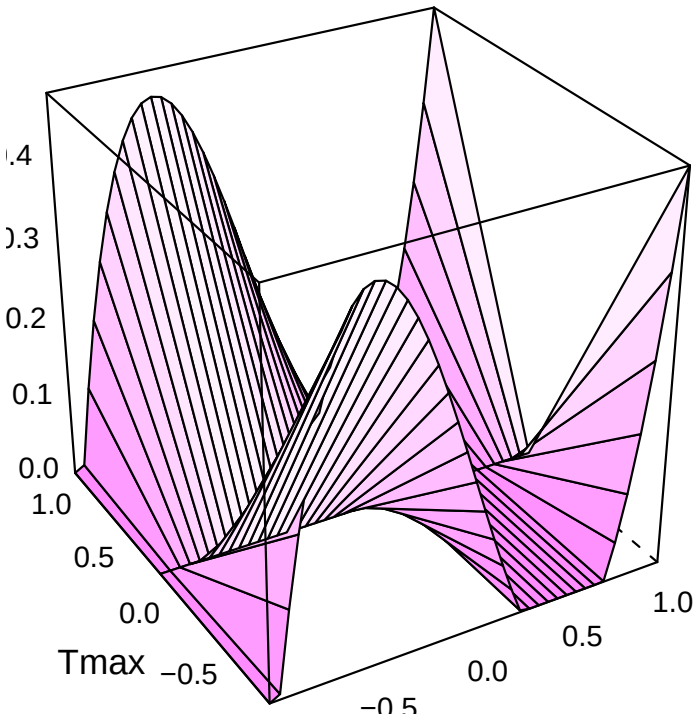
$$\min \left(\prod_{i=1}^{k} d_i \right)^{1/k} \approx \left(\widehat{\text{quality}} \cdot \widehat{\text{time}} \right)^{1/2}$$

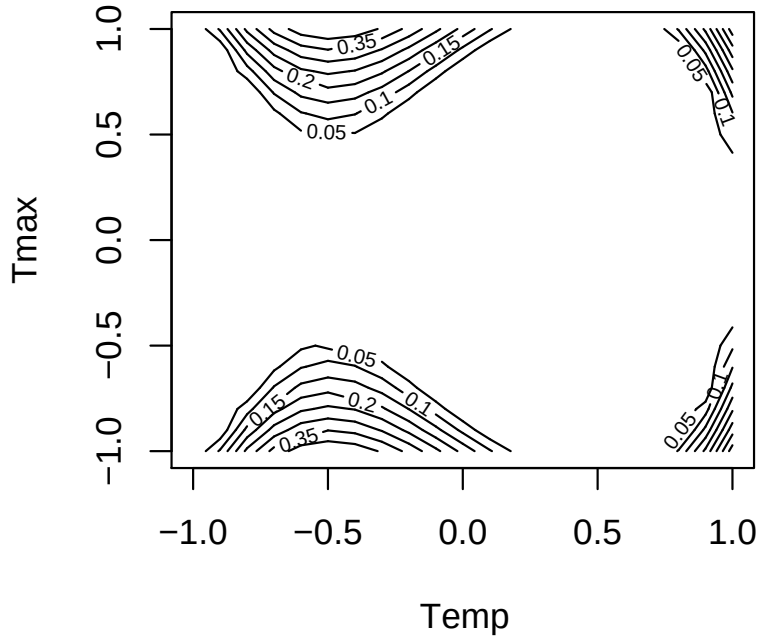
Example: SA for CAPM





Example: SA for CAPM





Conclusions:

- Eval=0, Temp=0.5, Tmax=0 (encoded variables)
- Eval=20000, Temp=13, Tmax=100
- But this is just only a local optimum!

Summary

ANOVA

- works well only if few factors
- analysis can be rather complicated

Regression Trees

+ very intuitive visualization of results
- require full factorial and no nesting
- problems with blocking
- black box and not used so far

Response Surface Methods:

- only for numerical parameters
- not automatic but interactive and time consuming
- restricted to analysis with crossing factors

Summary

Search methods

+ fully automatic (black box)
+ allow a very large search space
+ can handle nesting on algorithm factors
– not statistically sound
– Too many free parameters (paradoxical)

Race

+ fully automatic
+ statistically sound
+ can handle very well nesting on the algorithm factors
– indentifies the best but does not provide factorial analysis
– might still be lengthy but faster variants exists
– handles only univariate case, but bivariate examples exists [den Besten, 2004]

Outline

Analysis Scenarios

If the analysis scenario allows we can gain more precise insights by distribution modelling:

	Minimum Known	Minimum Unknown
Run Time	(VTR or gap) Restart Strategies	Time or idle iterations as parameters (see previous part)
Solution Quality	––	Estimation of Optima

It is good to keep always in mind what case one is considering

- $\langle$-/-/1/$r\rangle$
- $\langle$-/-/n/1$\rangle$

Outline

Characterization of Run-time

Parametric models used in the analysis of run-times to:

- provide more informative experimental results
- make more statistically rigorous comparisons of algorithms
- exploit the properties of the model
  (eg, the character of long tails and completion rate)
- predict missing data in case of censored distributions
- better allocation of resources
  - Restart strategies [Gagliolo and Schmidhuber, 2006]
  - Algorithm portfolios (multiple copies of the same algorithm in parallel) [Gomes and Selman 2001, Gagliolo and Schmidhuber, 2008]
  - Anytime algorithms (estimate the quality given the input and the amount of time that they will be executed) [Boddy and Dean, 1989]

Restart strategy: a sequence of cutoff times $T(k)$ for each restart k

Restart Strategies

$\langle$-/-/1/$r\rangle$

Theorem: [Luby, Sinclair, and Zuckerman, 1993]

If the RTD of an instance is known

$\Downarrow$

optimal restart strategy is uniform, that is,
it is based on constant cutoff, $T(r) = T^*$
To find T^*:
minimize expected value of total run time t_T which is given by:

$$E(t_T) = \frac{T - \int_0^T F(\tau)\, d\tau}{F(T)}$$

Two issues:

1. the theorem is valid only for one instance
2. $F(t)$ is the cdf of the run-time t of an unbounded run of the algorithm which is not know and its estimation might be costly

Distribution Modelling

We accept two approximations:

1. we generalize to a class of instances accepting that instances might be similar
2. we use $\widehat{F}(t)$ estimated from data, if necessary, censored.

Three offline methods:

A: modelling the full distribution

B: modelling the distribution with censored data

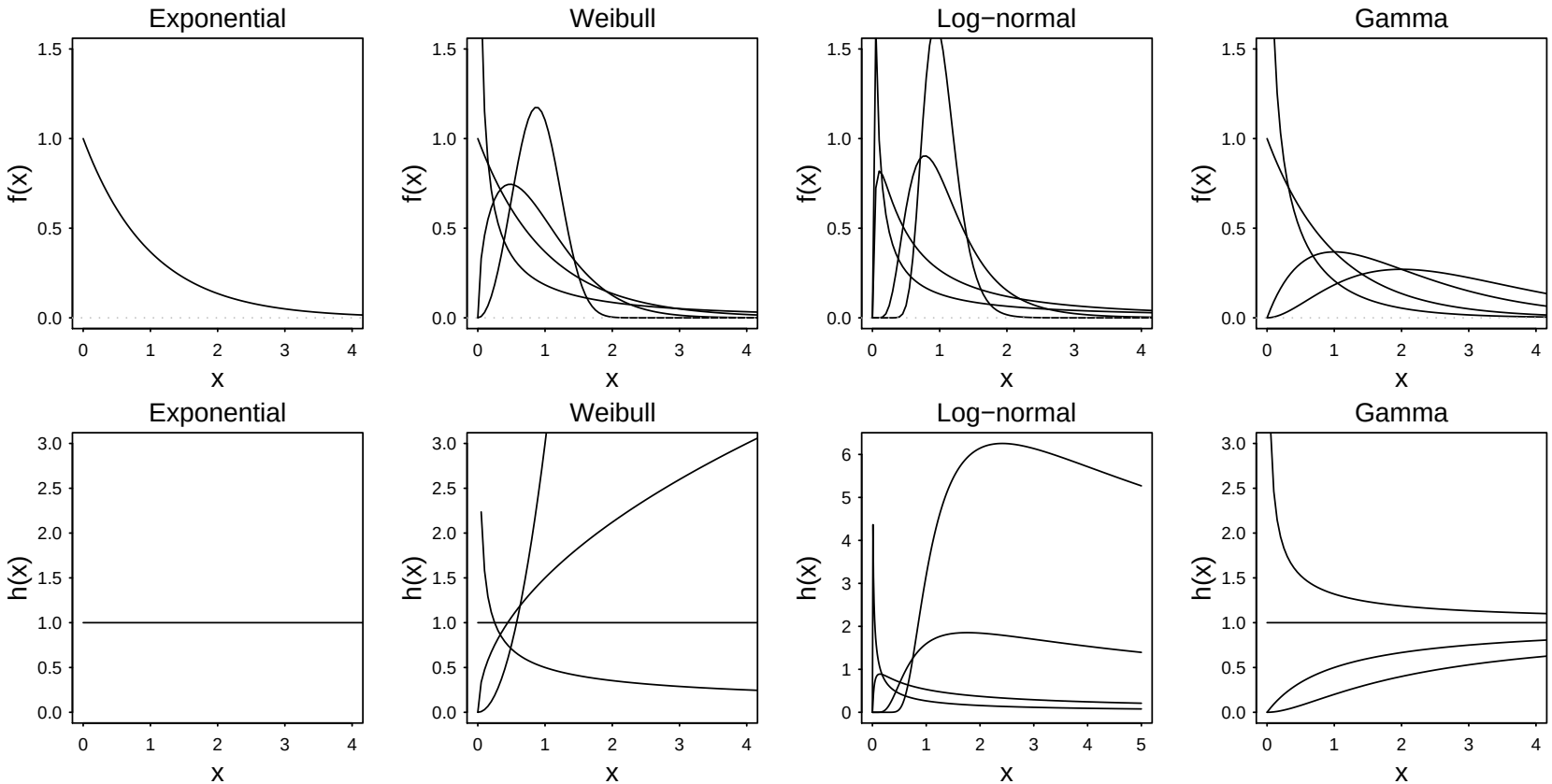
C: modelling the tails (extreme value statistics)

Procedure:

- choose a model, *i.e.*, probability function $f(x, \theta)$
- apply fitting method to determine the parameters
  Eg, maximum likelihood estimation method
- test the model (Kolmogorov-Smirnov goodness of fit tests)

Some Parametric Distributions

The distributions used are [Frost et al., 1997; Gomes et al., 2000]:



Run Time Distributions

Motivations for these distributions:

- qualitative information on the completion rate (= hazard function)
- empirical good fitting

Most of the work on RTDs is on SAT or CSP instances and $\langle$-/-/1/$r\rangle$

For complete backtracking algorithms:

- shown to be Weibull or lognormal distributed on CSP [Frost et al., 1997]
- shown to have heavy tails on CSP and SAT [Gomes et al., 1997]

For stochastic local search algorithms:

- shown to have mixture of exponential distributions [Hoos, 2002]

Model Fitting in Practice

Which parametric family of models is best for our data?

- underlying knowledge
- try to make plots that should be linear. Departures from linearity of the data can be easily appreciated by eye.

Example: for an exponential distribution, it is:

$$\log S(t) = -\lambda t, \qquad \text{where } S(t) = 1 - F(t) \text{ survivor function}$$

hence the plot of $\log S(t)$ against t should be linear.

Similarly, for the Weibull
the cumulative hazard function is linear on a log-log plot

Application Example

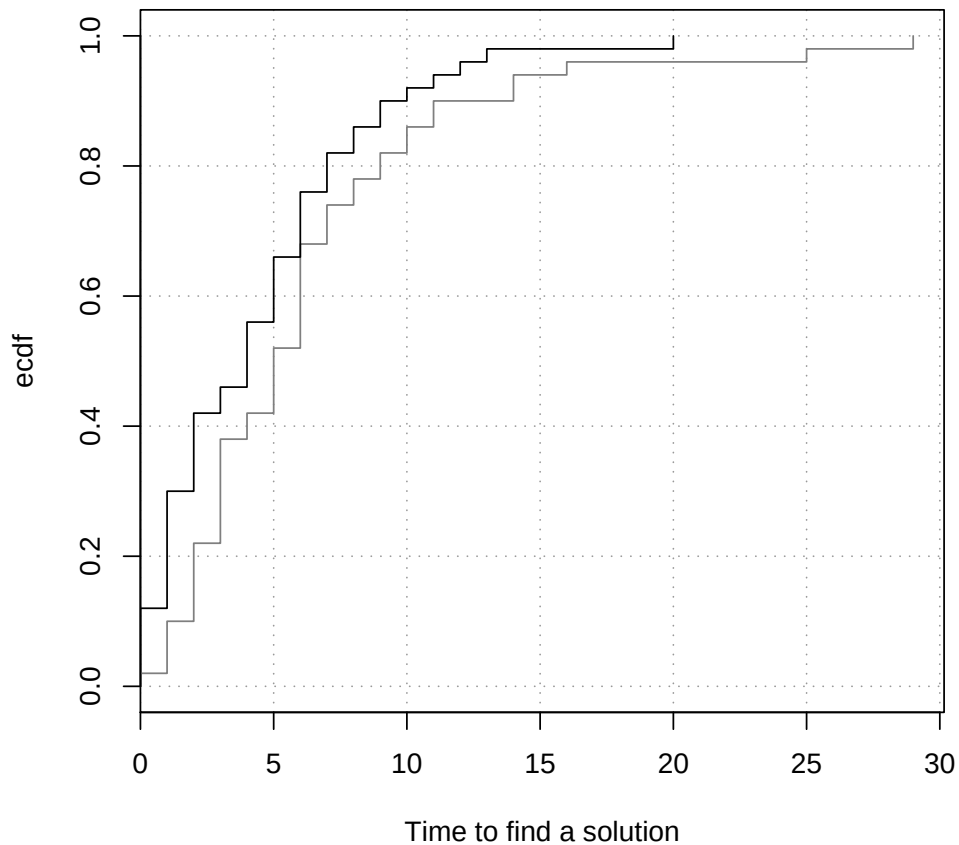
Difficulties in the application to the CAPM case:

- The best algorithm is random restart Nelder-Mead (which already uses restart!)
- SA and DE never reach the solutions returned by RRNM hence all runs would be censored!
- Optimum unknown. Deciding a VTR or a gap: Which one? Why?
- In these cases the analysis provided before is enough to tell us when to restart.

Example on CSP

Characterization of Run-time

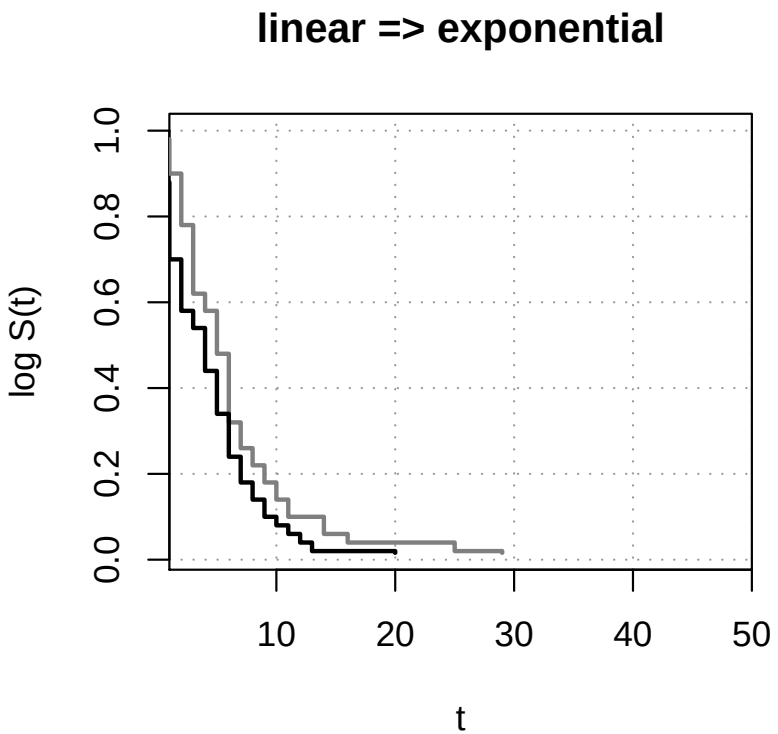
Two algorithms for a CSP problem. 50 runs on a single instance with time limit 100 seconds.

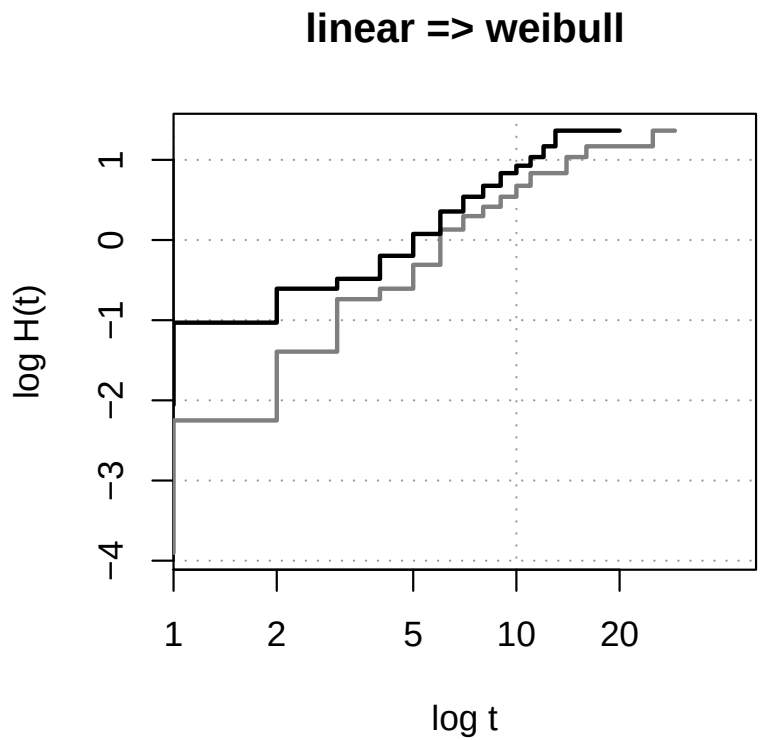


Example on CSP

Characterization of Run-time

Two algorithms for a CSP problem. 50 runs on a single instance with time limit 100 seconds.





Characterization of Run-time

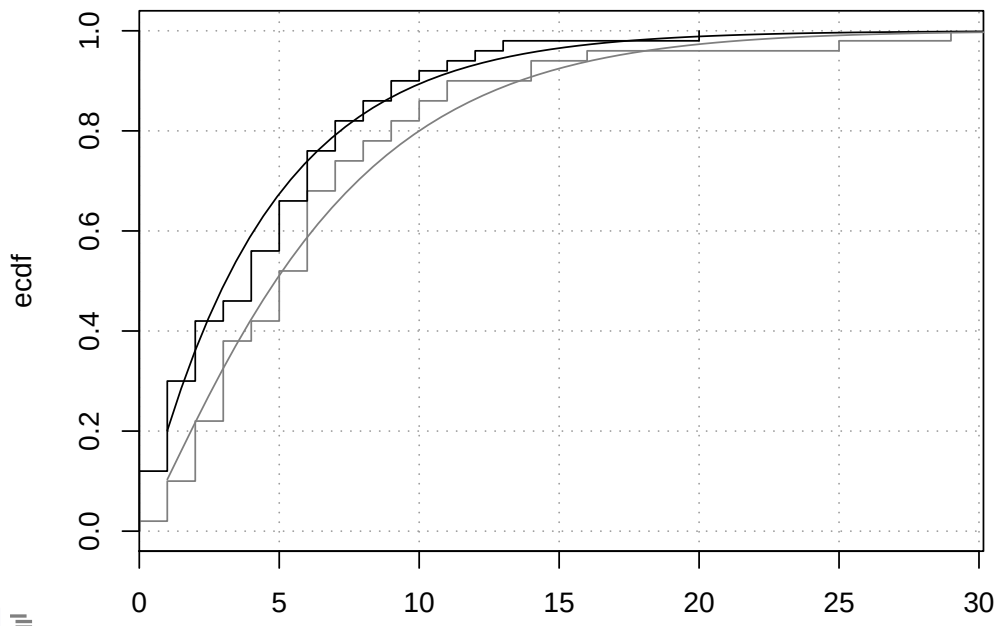
Example

Distribution fitting

$f(t, \theta)$ probability density function of solution time t with parameter θ.
Maximum likelihood method:

$$\max_{\theta} L(T_1, T_2, \ldots, T_k \mid \theta) = \prod_{i}^{k} Pr(T_i \mid \theta) = \prod_{i=1}^{k} f(T_i \mid \theta)$$

Example: $f()$ exponential or Weibull



- grey curve: Weibull distributed with KS test p-value 0.4955
- black curve: exponential distributed with KS test p-value 0.3470

B: Fitting Censored Distributions

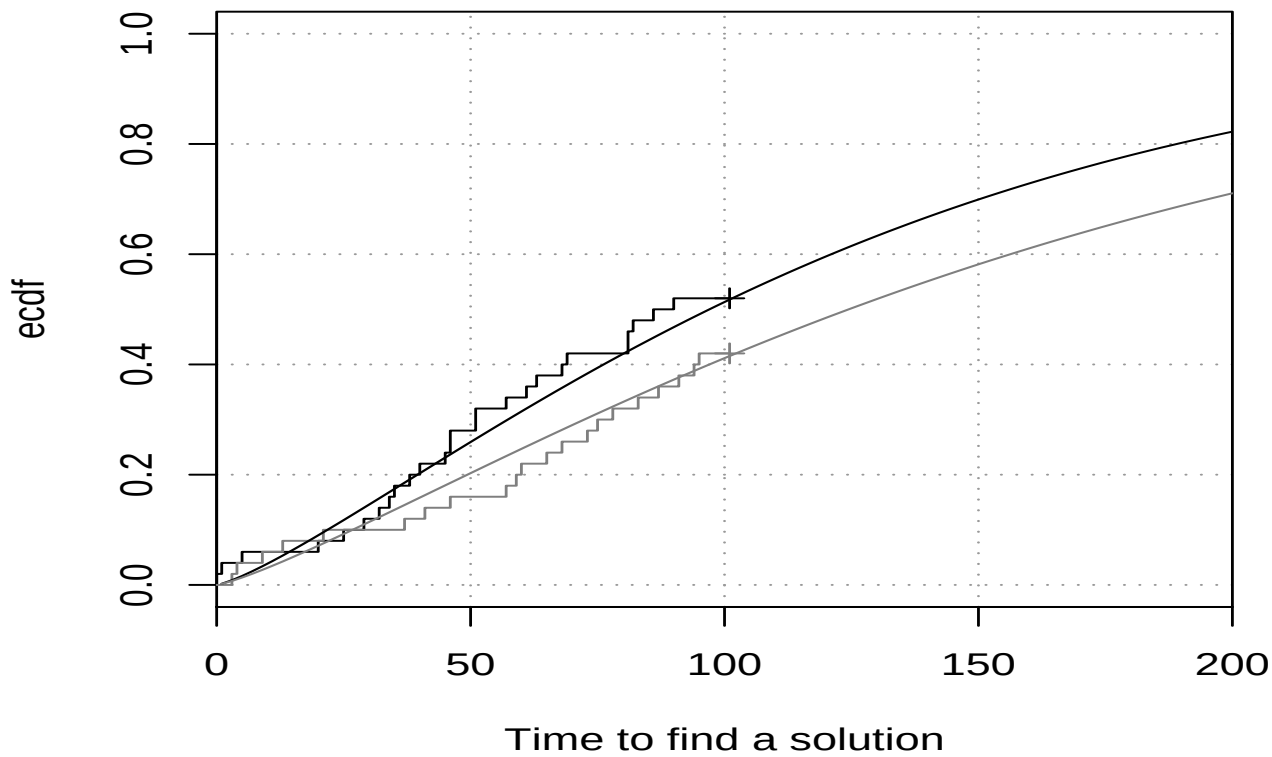
Type I censor sampling:
decide a cutoff time t_c and stop experiments that exceed that cutoff
Using indicator function δ_i:

$$L(T|\theta) = \prod_{i=1}^{k} f(T_i|\theta)^{\delta_i} \left(\int_{t_c}^{\infty} f(\tau|\theta)\, d\tau \right)^{1-\delta_i}$$

Type II censor sampling:
r experiments are run in parallel and stopped whenever u uncensored samples are obtained.
Thus, $c = (r-u)/r$ are set in advance, and t_c is equal to time of uth fastest.

Example on CSP

$\langle -/-/1/r \rangle$



Application Example
Learning Restart Strategy

[*Impact of Censored Sampling on the Performance of Restart Strategies*
Gagliolo & Schmidhuber, CP 2006]
⟨-/-/n/r⟩

Use the following learning scheme based on Type II censoring to estimate $\widehat{F}$:

- pick $n = 50$ instances at random and start $r = 20$ runs with different seed on each instance ➡ $k = nr$ experiments
- fix a censoring threshold $c \in [0, 1]$.
  As the first $\lfloor (1-c)k \rfloor$ runs terminate, stop also the remaining $\lceil ck \rceil$.
- data are used to train a model $\widehat{F}$ of RTD by solving max likelihood
- from $\widehat{F}$ a uniform strategy is derived by solving:

$$\min_T \frac{T - \int_0^T F(\tau)\, d\tau}{F(T)}$$

- test performance on the remaining instances of the class

Note: tradeoff training time *vs* censor threshold u

C: Haevy Tails

Extreme Value Statistics

- Extreme value statistics focuses on characteristics related to the tails of a distribution function.
  1. indices describing tail decay
  2. extreme quantiles (*e.g.*, minima)

- 'Classical' statistical theory: analysis of means.
  Central limit theorem: $X_1, \ldots, X_n$ i.i.d. with F_X

$$\sqrt{n}\frac{\bar{X} - \mu}{\sqrt{Var(X)}} \xrightarrow{D} N(0,1), \qquad \text{as } n \to \infty$$

  Heavy tailed distributions: mean and/or variance may not be finite!

Heavy Tails

[Gomes, Selman, Crato, and Kautz, 2000] analyze the mean computational cost of backtracking algorithms to find a solution on a single instance of CSP $\langle \text{-}/\text{-}/1/r \rangle$

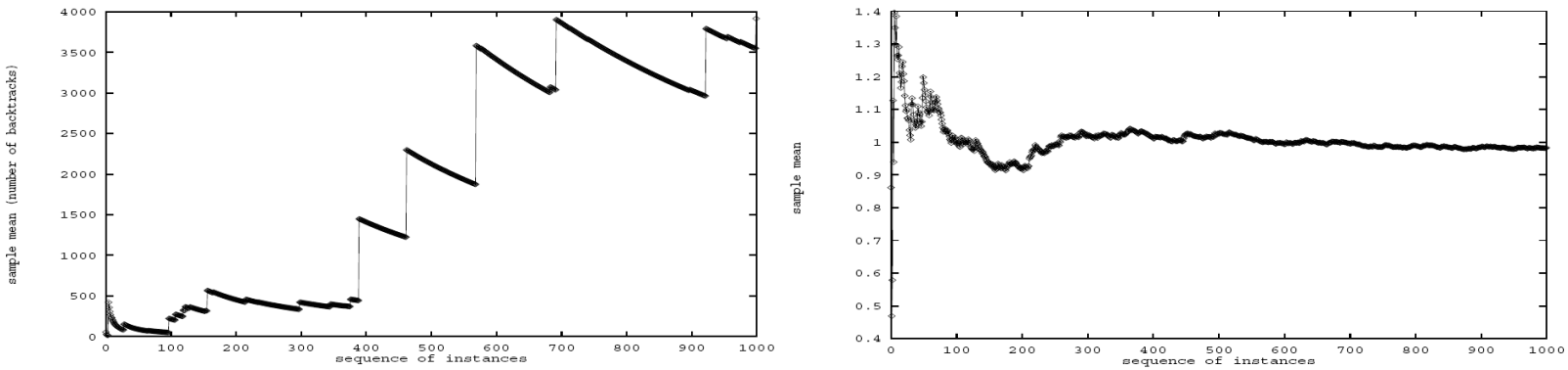


Figure: Mean calculated over an increasing number of runs. Left, erratic behavior, long tail. Right, the case of data drawn from normal or gamma distributions.

- The existence of the moments (*e.g.*, mean, variance) is determined by the tails behavior: long tails imply non existence
- This suggests the use of the median rather than the mean for reporting

Extreme Value Statistics

Tail theory

- Work with data exceeding a high threshold.
- Conditional distribution of exceedances over threshold τ

$$1 - F_\tau(y) = P(X - \tau > y \mid X > \tau) = \frac{P(X > \tau + y)}{P(X > \tau)}$$

- Theorem of [Fisher and Tippett, 1928]:
  the distribution of extremes tends in distribution to a generalized extreme value distribution (GEV) $\Leftrightarrow$ exceedances tend to a generalized Pareto distribution

  Pareto-type distribution function

$$1 - F_X(x) = x^{-\frac{1}{\gamma}} \ell_F(x), \qquad x > 0,$$

  where $\ell_F(x)$ is a slowly varying function at infinity.

In practice, fit a function $Cx^{-\frac{1}{\gamma}}$ to the exceedances:
$Y_j = X_i - \tau$, provided $X_i > \tau$, $j = 1, \ldots, N_\tau$.
γ determines the nature of the tail

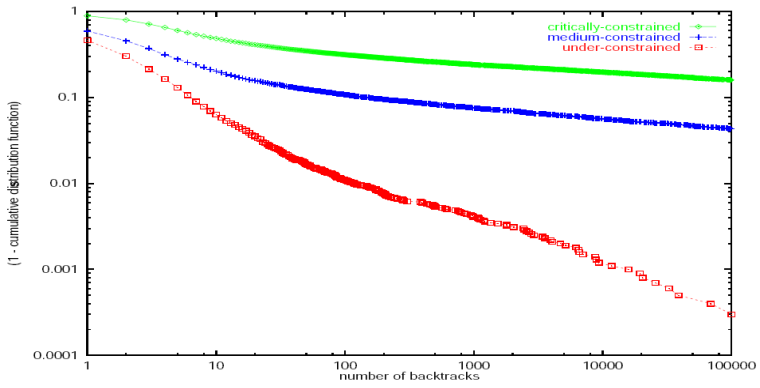
Heavy Tails

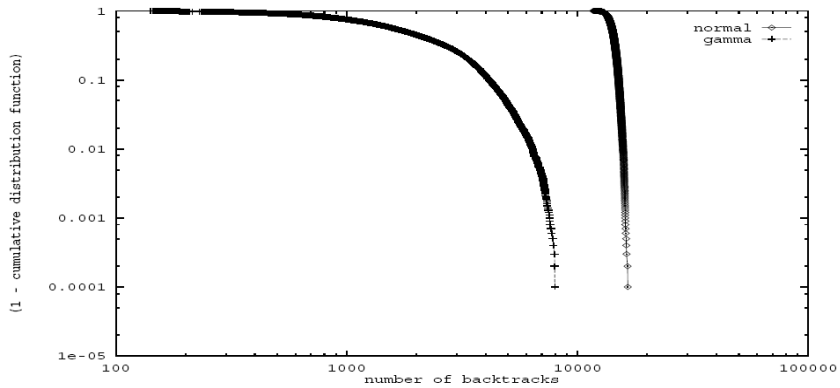
The estimated values of γ give indications on the tails:

- $\gamma > 1$: long tails, hyperbolic decay and mean not finite (the completion rate decreases with t)
- $\gamma < 1$: tails exhibit exponential decay

Graphical check using a log-log plot (or a Pareto qqplot)

- heavy tail distributions approximate linear decay,
- exponentially decreasing tail has faster-than linear decay



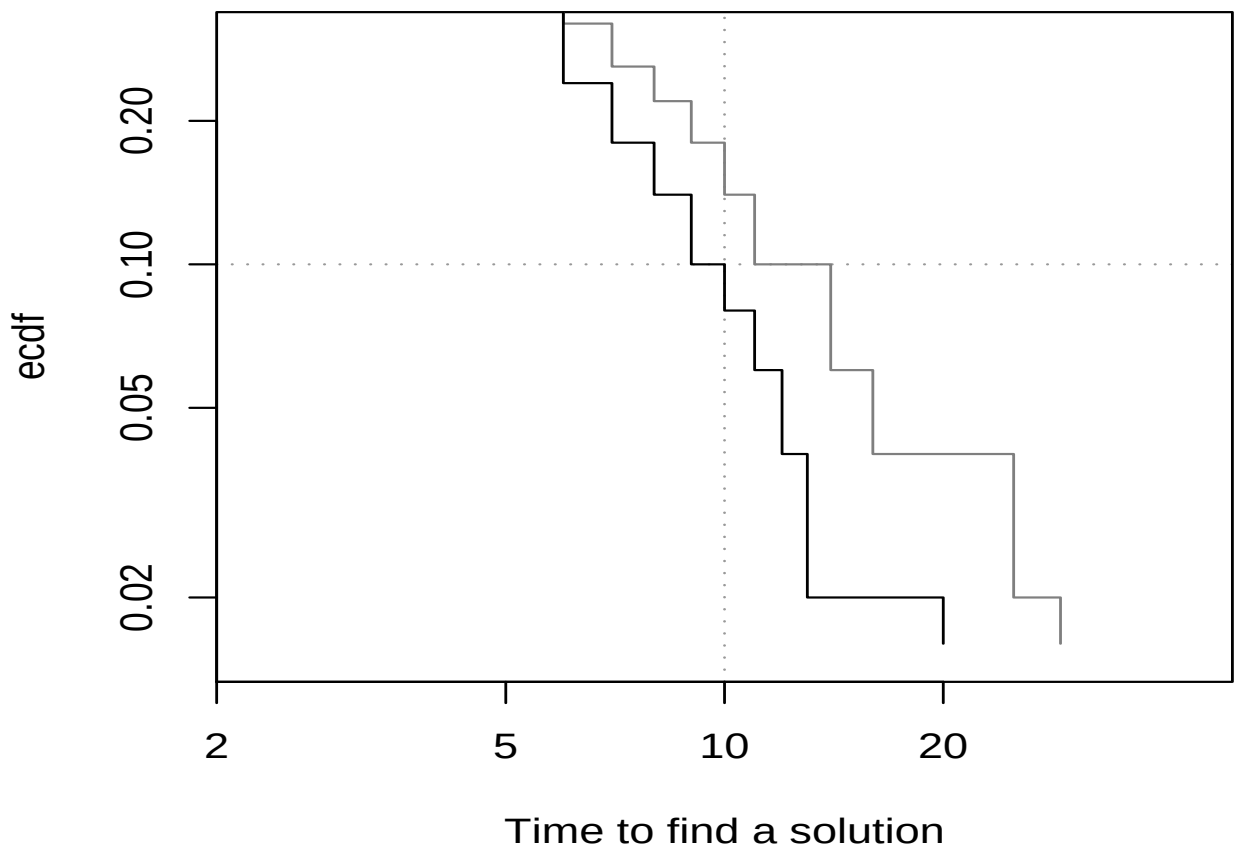


Long tails explain the goodness of random restart. Determining the cutoff time is however not trivial.

Example on CSP

Heavy Tails

$\langle -/-/1/r \rangle$



Outline

Analysis Scenarios

If the analysis scenario allows we can gain more precise insights by distribution modelling:

	Minimum Known	Minimum Unknown
Run Time	(VTR or gap) Restart Strategies	Time or idle iterations as parameters (see previous part)
Solution Quality	––	Estimation of Optima

It is good to keep always in mind what case one is considering

- $\langle$-/-/1/$r\rangle$
- $\langle$-/-/n/1$\rangle$

Extreme Values Statistics

Extreme values theory

- $X_1, X_2, \ldots, X_n$ i.i.d. F_X
  Ascending order statistics $X_n^{(1)} \leq \ldots \leq X_n^{(n)}$
- For the minimum $X_n^{(1)}$ it is $F_{X_n^{(1)}} = 1 - [1 - F_X^{(1)}]^n$ but not very useful in practice as F_X unknown
- Theorem of [Fisher and Tippett, 1928]:
  "almost always" the normalized extreme tends in distribution to a generalized extreme value distribution (GEV) as $n \to \infty$.

In practice, the distribution of extremes is approximated by a GEV:

$$F_{X_n^{(1)}}(x) \sim \begin{cases} \exp(-1(1-\gamma\frac{x-\mu}{\sigma})^{-1/\gamma}, & 1-\gamma\frac{x-\mu}{\sigma} > 0, \gamma \neq 0 \\ \exp(-\exp(\frac{x-\mu}{\sigma})), & x \in \mathbf{R}, \gamma = 0 \end{cases}$$

Parameters estimated by simulation by repeatedly sampling k values $X_{1n}, \ldots, X_{kn}$, taking the extremes $X_{kn}^{(1)}$, and fitting the distribution.
γ determines the type of distribution: Weibull, Fréchet, Gumbel, ...

Characterization of Quality

On a single instance

Application of distribution modelling and extreme values theory for the characterization of solution quality.

- In random picking, final quality is the minimum cost of k i.i.d. solutions generated, that is, $Y_k^{(1)}$.
  Hence, possible to simulate the distribution of minima by repeating n times.
- In other stochastic optimizers, steps are dependent, but possible to simulate independence by taking the minimum over $l < k$ and over k and repeating for n times
- Studies conducted by [Ovacik et al., 2000; Hüsler et al., 2003].
  Possible to estimate the distance from the optimum: If the fitting indicates the Weibull (finite left tail) as the best then solutions near to the optimum

Note: extreme value theory applies only to asymptotically continuous functions!

Outline

Summary

- Common practice in CS and OR to report results on benchmark instances in numerical tables.
- Graphics are complementary to tables and are often better suitable for summarizing data.
- Not a single standard tool for analysis but several tools and several aspects to look at. Look at every case as a different one.
- For configuration and tuning: racing methodologies make things easy.
  Alternatively: Regression trees, search methods, response surface, ANOVA
- Modelling can be insightful but limited to problems that can be solved.
  Restart, comparisons, prediction.

References

Bäck T. and Hoffmeister F. (2004). **Basic aspects of evolution strategies**. *Statistics and Computing*, 4(2), pp. 51–63.

Bartz-Beielstein T. (2006). **Experimental Research in Evolutionary Computation – The New Experimentalism**. Natural Computing Series. Springer, Berlin.

Bartz-Beielstein T. and Markon S. (2004). **Tuning search algorithms for real-world applications: A regression tree based approach**. In *Congress on Evolutionary Computation (CEC'04)*, pp. 1111–1118. IEEE Press, Piscataway NJ.

Beyer H.G. (2001). **On the performance of the (1,λ)-evolution strategies for the ridge function class**. *IEEE Transactions on Evolutionary Computation*, 5(3), pp. 218–235.

Birattari M. (2004). **On the estimation of the expected performance of a metaheuristic on a class of instances. how many instances, how many runs?** Tech. Rep. TR/IRIDIA/2004-01, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium.

Birattari M. (2005). **The Problem of Tuning Metaheuristics as Seen from a Machine Learning Perspective**. DISKI 292. Infix/Aka, Berlin, Germany.

Birattari M., Pellegrini P., and Dorigo M. (2007). **On the invariance of ant colony optimization**. *IEEE Transactions on Evolutionary Computation*, 11(6), pp. 732–742.

Birattari M., Stützle T., Paquete L., and Varrentrapp K. (2002). **A racing algorithm for configuring metaheuristics**. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002)*, edited by W.B. Langdon, E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. Potter, A. Schultz, J. Miller, E. Burke, and N. Jonoska, pp. 11–18. Morgan Kaufmann Publishers, New York.

Bratley P., Fox B.L., and Niederreiter H. (1994). **Algorithm-738 - programs to generate niederreiters low-discrepancy sequences**. *ACM Transactions On Mathematical Software*, 20(4), pp. 494–495.

Coffin M. and Saltzman M.J. (2000). **Statistical analysis of computational tests of algorithms and heuristics**. *INFORMS Journal on Computing*, 12(1), pp. 24–44.

References (2)

Conover W. (1999). **Practical Nonparametric Statistics**. John Wiley & Sons, New York, NY, USA, third ed.

den Besten M.L. (2004). **Simple Metaheuristics for Scheduling: An empirical investigation into the application of iterated local search to deterministic scheduling problems with tardiness penalties**. Ph.D. thesis, Darmstadt University of Technology, Darmstadt, Germany.

Frost D., Rish I., and Vila L. (1997). **Summarizing CSP hardness with continuous probability distributions.** In *Proceedings of AAAI/IAAI*, pp. 327–333.

Gomes C., Selman B., and Crato N. (1997). **Heavy-tailed distributions in combinatorial search**. In *Principles and Practices of Constraint Programming, CP-97*, vol. 1330 of **lncs**, pp. 121–135. springer-lncs, Linz, Austria.

Gomes C., Selman B., Crato N., and Kautz H. (2000). **Heavy-tailed phenomena in satisfiability and constraint satisfaction problems**. *Journal of Automated Reasoning*, 24(1-2), pp. 67–100.

Gutjahr W.J. (2008). **First steps to the runtime complexity analysis of ant colony optimization**. *Computers & OR*, 35(9), pp. 2711–2727.

Hoos H.H. (2002). **A mixture-model for the behaviour of sls algorithms for sat.** In *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI-02)*, pp. 661–667. AAAI Press / The MIT Press.

Hothorn T., Hornik K., and Zeileis A. (2006). **Unbiased recursive partitioning: A conditional inference framework**. *Journal of Computational and Graphical Statistics*, 15(3), pp. 651–674.

Hüsler J., Cruz P., Hall A., and Fonseca C.M. (2003). **On optimization and extreme value theory**. *Methodology and Computing in Applied Probability*, 5, pp. 183–195.

Hutter F., Hoos H.H., and Stützle T. (2007). **Automatic algorithm configuration based on local search**. In *Proc. of the Twenty-Second Conference on Artifical Intelligence (AAAI '07)*, pp. 1152–1157.

Kutner M.H., Nachtsheim C.J., Neter J., and Li W. (2005). **Applied Linear Statistical Models**. McGraw Hill, fifth ed.

Lawless J.F. (1982). **Statistical Models and Methods for Lifetime Data**. Wiley Series in Probability and Mathematical Statistics. jws.

Luby M., Sinclair A., and Zuckerman D. (1993). **Optimal speedup of las vegas algorithms**. *Information Processing Letters*, 47(4), pp. 173–180.

References (3)

McGeoch C.C. (1992). **Analyzing algorithms by simulation: Variance reduction techniques and simulation speedups**. *ACM Computing Surveys*, 24(2), pp. 195–212.

McGeoch C.C. (1996). **Toward an experimental method for algorithm simulation**. *INFORMS Journal on Computing*, 8(1), pp. 1–15.

Michiels W., Aarts E., and Korst J. (2007). **Theoretical Aspects of Local Search**. Monographs in Theoretical Computer Science, An EATCS Series. Springer Berlin Heidelberg.

Montgomery D.C. (2005). **Design and Analysis of Experiments**. John Wiley & Sons, sixth ed.

Montgomery D.C. and Runger G.C. (2007). **Applied Statistics and Probability for Engineers**. John Wiley & Sons, fourth ed.

Nelder J.A. and Mead R. (1965). **A simplex method for function minimization**. *The Computer Journal*, 7(4), pp. 308–313. An Errata has been published in The Computer Journal 1965 8(1):27.

Ovacik I.M., Rajagopalan S., and Uzsoy R. (2000). **Integrating interval estimates of global optima and local search methods for combinatorial optimization problems**. *Journal of Heuristics*, 6(4), pp. 481–500.

Petruccelli J.D., Nandram B., and Chen M. (1999). **Applied Statistics for Engineers and Scientists**. Prentice Hall, Englewood Cliffs, NJ, USA.

Ridge E. and Kudenko D. (2007a). **Analyzing heuristic performance with response surface models: prediction, optimization and robustness**. In *Proceedings of GECCO*, edited by H. Lipson, pp. 150–157. ACM.

Ridge E. and Kudenko D. (2007b). **Screening the parameters affecting heuristic performance**. In *Proceedings of GECCO*, edited by H. Lipson, p. 180. ACM.

Seber G. (2004). **Multivariate observations**. Wiley series in probability and statistics. John Wiley.

Wolpert D.H. and Macready W.G. (1997). **No free lunch theorems for optimization**. *IEEE Transactions on Evolutionary Computation*, 1(1), pp. 67–82.

Empirical Methods for the Analysis of Optimization Heuristics

Marco Chiarandini

Department of Mathematics and Computer Science
University of Southern Denmark, Odense, Denmark
www.imada.sdu.dk/~marco
www.imada.sdu.dk/~marco/COMISEF08

October 16, 2008
COMISEF Workshop