DM811
Heuristics for Combinatorial Optimization

Lecture 13
Experimental Analysis

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Outline

Outline

1. Experimental Analysis
    Definitions
    Performance Measures
    Sample Statistics
    Scenarios of Analysis
    Guidelines for Presenting Data

2. Examples
    Results Task 1
    Results Task 2

Contents and Goals

Provide a view of issues in Experimental Algorithmics

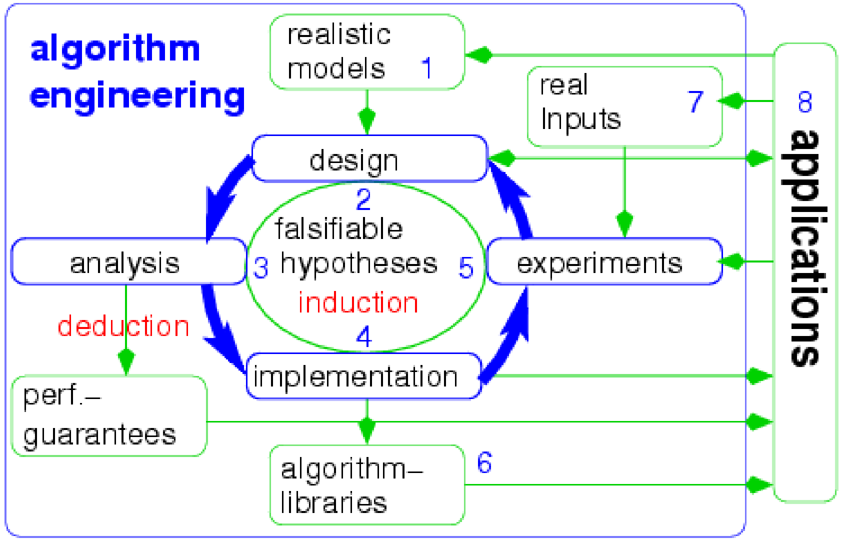
- Exploratory data analysis
- Presenting results in a concise way with graphs and tables
- Organizational issues and Experimental Design

- Basics of inferential statistics
- Sequential statistical testing: race, a methodology for tuning

The goal of Experimental Algorithmics is not only producing a sound analysis but also adding an important tool to the development of a good solver for a given problem.

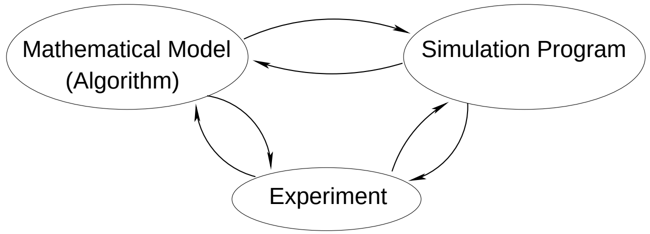
Experimental Algorithmics is an important part in the algorithm production cycle, which is referred to as Algorithm Engineering

The Engineering Cycle



from http://www.algorithm-engineering.de/

Experimental Algorithmics



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

[McGeoch, 1996]

Experimental Algorithmics

Goals

- Defining standard methodologies
- Comparing relative performance of algorithms so as to identify the best ones for a given application
- Characterizing the behavior of algorithms
- Identifying algorithm separators, *i.e.*, families of problem instances for which the performance differ
- Providing new insights in algorithm design

Fairness Principle

Fairness principle: being completely fair is perhaps impossible but try to remove any possible bias

- possibly all algorithms must be implemented with the same style, with the same language and sharing common subprocedures and data structures
- the code must be optimized, e.g., using the best possible data structures
- running times must be comparable, e.g., by running experiments on the same computational environment (or redistributing them randomly)

Definitions

The most typical scenario considered in analysis of search heuristics

Asymptotic heuristics with time (or iteration) limit decided *a priori*

The algorithm $\mathcal{A}^\infty$ is halted when time expires.

Deterministic case: $\mathcal{A}^\infty$ on π returns a solution of cost x.

The performance of $\mathcal{A}^\infty$ on π is a scalar $y = x$.

Randomized case: $\mathcal{A}^\infty$ on π returns a solution of cost X, where X is a random variable.

The performance of $\mathcal{A}^\infty$ on π is the univariate $Y = X$.

[This is not the only relevant scenario: to be refined later]

Random Variables and Probability

Statistics deals with random (or stochastic) variables.

A variable is called random if, prior to observation, its outcome cannot be predicted with certainty.

The uncertainty is described by a probability distribution.

Discrete variables

Probability distribution:

$$p_i = P[x = v_i]$$

Cumulative Distribution Function (CDF)

$$F(v) = P[x \leq v] = \sum_i p_i$$

Mean

$$\mu = E[X] = \sum x_i p_i$$

Variance

$$\sigma^2 = E[(X - \mu)^2] = \sum (x_i - \mu)^2 p_i$$

Continuous variables

Probability density function (pdf):

$$f(v) = \frac{dF(v)}{dv}$$

Cumulative Distribution Function (CDF):

$$F(v) = \int_{-\infty}^{v} f(v) dv$$

Mean

$$\mu = E[X] = \int x f(x) dx$$

Variance

$$\sigma^2 = E[(X - \mu)^2] = \int (x - \mu)^2 f(x)\, dx$$

Generalization

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

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

$$Pr(Y = y \mid \pi)$$

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

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

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.

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

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

Instance Selection

In real-life applications a simulation of $p(\pi)$ can be obtained by historical data.

In simulation studies instances may be:

- real world instances
- random variants of real world-instances
- online libraries
- randomly generated instances

They may be grouped in classes according to some features whose impact may be worth studying:

- type (for features that might impact performance)
- size (for scaling studies)
- hardness (focus on hard instances)
- application (e.g., CSP encodings of scheduling problems), ...

Within the class, instances are drawn with uniform probability $p(\pi) = c$

Statistical Methods

The analysis of performance is based on finite-sized sampled data.
Statistics provides the methods and the mathematical basis to

- describe, summarizing, the data (descriptive statistics)
- make inference on those data (inferential statistics)

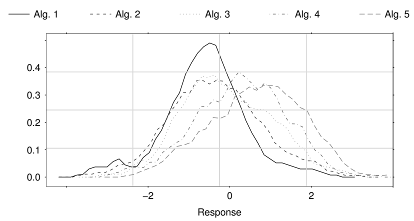
Statistics helps to

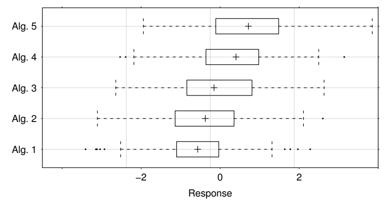
- guarantee reproducibility
- make results reliable
  (are the observed results enough to justify the claims?)
- extract relevant results from large amount of data

In the practical context of heuristic design and implementation (i.e., engineering), statistics helps to take correct design decisions with the least amount of experimentation

Objectives of the Experiments

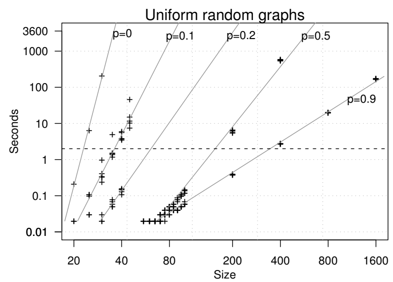
- **Comparison**:
  bigger/smaller, same/different,
  Algorithm Configuration,
  Component-Based Analysis
  - Standard statistical methods: *experimental designs, test hypothesis and estimation*





Objectives of the Experiments

- **Comparison**:
  bigger/smaller, same/different,
  Algorithm Configuration,
  Component-Based Analysis
  - Standard statistical methods: *experimental designs, test hypothesis and estimation*
- **Characterization**:
  Interpolation: fitting models to data
  Extrapolation: building models of data, explaining phenomena
  - Standard statistical methods: *linear and non linear regression* model fitting



Measures and Transformations

On a single instance

Computational effort indicators

- number of elementary operations/algorithmic iterations (e.g., search steps, objective function evaluations, number of visited nodes in the search tree, consistency checks, etc.)
- total CPU time consumed by the process (sum of *user* and *system* times returned by `getrusage`)

Solution quality indicators

- value returned by the cost function
- error from optimum/reference value
- (optimality) gap $\frac{|UB-LB|}{UB}$
- ranks

Measures and Transformations

On a class of instances

Computational effort indicators

- no transformation if the interest is in studying scaling
- 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),
- otherwise, better to group homogeneously the instances

Solution quality indicators

Different instances imply different scales $\Rightarrow$ need for an invariant measure

(However, many other measures can be taken both on the algorithms and on the instances [McGeoch, 1996])

Measures and Transformations

On a class of instances (cont.)

Solution quality indicators

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

$$e_1(x,\pi) = \frac{x(\pi) - \bar{x}(\pi)}{\sqrt{\hat{\sigma(\pi)}}} \qquad \text{standard score}$$

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

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

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

Summary Measures

Measures to describe or characterize a population

- Measure of central tendency, location
- Measure of dispersion

One such a quantity is

- a **parameter** if it refers to the population (Greek letters)
- a **statistics** if it is an *estimation* of a population parameter from the sample (Latin letters)

Measures of central tendency

- Arithmetic Average (Sample mean)

$$\bar{X} = \frac{\sum x_i}{n}$$

- *Quantile*: value above or below which lie a fractional part of the data (used in nonparametric statistics)
  - Median

  $$\mathcal{M} = x_{(n+1)/2}$$

  - Quartile

  $$Q_1 = x_{(n+1)/4} \qquad Q_3 = x_{3(n+1)/4}$$

  - q-quantile

    q of data lies below and $1-q$ lies above
- Mode

  value of relatively great concentration of data
  (*Unimodal* vs *Multimodal* distributions)

Measure of dispersion

- Sample range

$$R = x_{(n)} - x_{(1)}$$

- Sample variance

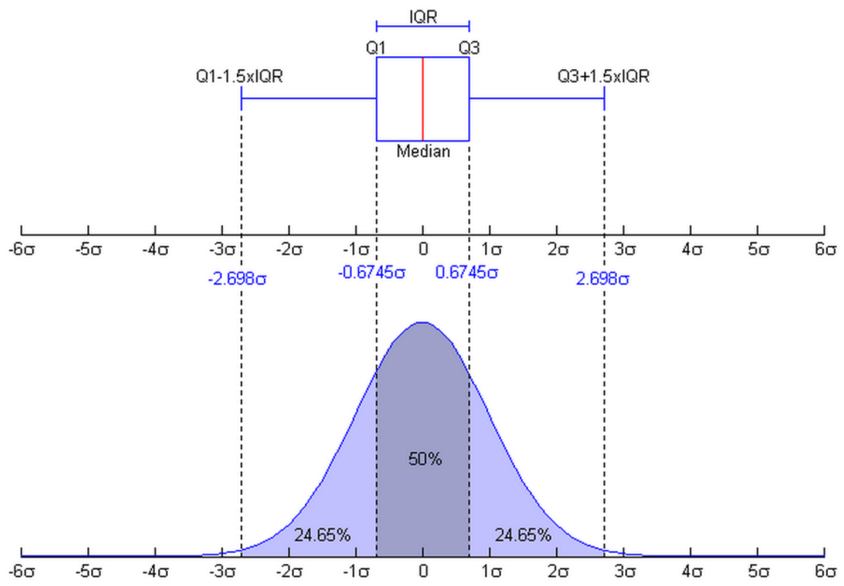
$$s^2 = \frac{1}{n-1}\sum(x_i - \bar{X})^2$$

- Standard deviation

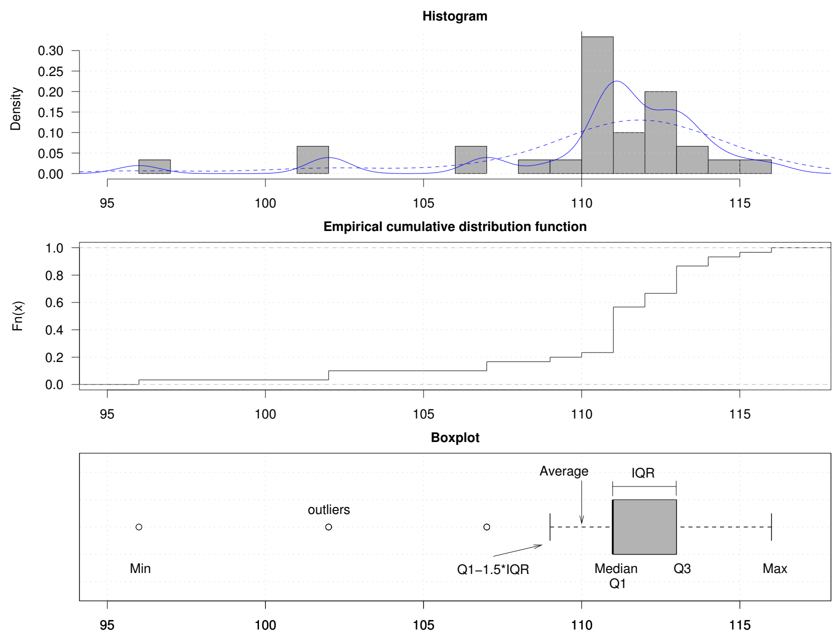
$$s = \sqrt{s^2}$$

- Inter-quartile range

$$IQR = Q_3 - Q_1$$



Boxplot and a probability density function (pdf) of a Normal N(0,1s2) Population. (source: Wikipedia)
[see also: `http://informationandvisualization.de/blog/box-plot`]



In R


```
> x<-runif(10,0,1)
 mean(x), median(x), quantile(x), quantile(x,0.25)
 range(x), var(x), sd(x), IQR(x)
> fivenum(x)
#(minimum, lower-hinge, median, upper-hinge, maximum)
[1] 0.18672 0.26682 0.28927 0.69359 0.92343
> summary(x)
> aggregate(x,list(factors),median)
> boxplot(x)
```


Scenarios

A. One-pass heuristics

B. Asymptotic heuristics:
   Two approaches:
   1. Univariate
      1.a Time as an external parameter decided *a priori*
      1.b Solution quality as an external parameter decided *a priori*
   2. Cost dependent on running time:

Scenario A

One-pass heuristics

Deterministic case: $\mathcal{A}^{\dashv}$ on class $\mathcal{C}_{\sqcap}$ returns a solution of cost x with computational effort t (*e.g.*, running time).

The performance of $\mathcal{A}^{\dashv}$ on class $\mathcal{C}_{\sqcap}$ is the vector $\vec{y} = (x, t)$.

Randomized case: $\mathcal{A}^{\dashv}$ on class $\mathcal{C}_{\sqcap}$ returns a solution of cost X with computational effort T, where X and T are random variables.

The performance of $\mathcal{A}^{\dashv}$ on class $\mathcal{C}_{\sqcap}$ is the bivariate $\vec{Y} = (X, T)$.

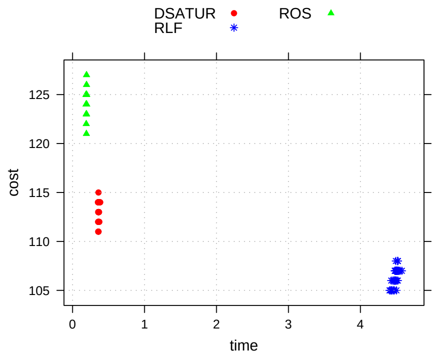
Example

Scenario:

- 3 heuristics $\mathcal{A}_1^{\dashv}$, $\mathcal{A}_2^{\dashv}$, $\mathcal{A}_3^{\dashv}$ on class $\mathcal{C}_{\sqcap}$.
- homogeneous instances or need for data transformation.
- 1 or r runs per instance
- Interest: inspecting solution cost and running time to observe and compare the level of approximation and the speed.

Tools:

- Scatter plots of solution-cost and run-time



Multi-Criteria Decision Making

Needed some definitions on dominance relations

In Pareto sense, for points in $\mathbf{R}^2$

$\vec{x}^1 \preceq \vec{x}^2$	weakly dominates	$x_i^1 \leq x_i^2$ for all $i = 1, \ldots, n$
$\vec{x}^1 \parallel \vec{x}^2$	incomparable	neither $\vec{x}^1 \preceq \vec{x}^2$ nor $\vec{x}^2 \preceq \vec{x}^1$

Scenario B

Asymptotic heuristics

There are two approaches:

1.a. Time as an external parameter decided *a priori*.
The algorithm is halted when time expires.

Deterministic case: $\mathcal{A}^\infty$ on class $\mathcal{C}_\Pi$ returns a solution of cost x.

The performance of $\mathcal{A}^\infty$ on class $\mathcal{C}_\Pi$ is the scalar $y = x$.

Randomized case: $\mathcal{A}^\infty$ on class $\mathcal{C}_\Pi$ returns a solution of cost X, where X is a random variable.

The performance of $\mathcal{A}^\infty$ on class $\mathcal{C}_\Pi$ is the univariate $Y = X$.

Example

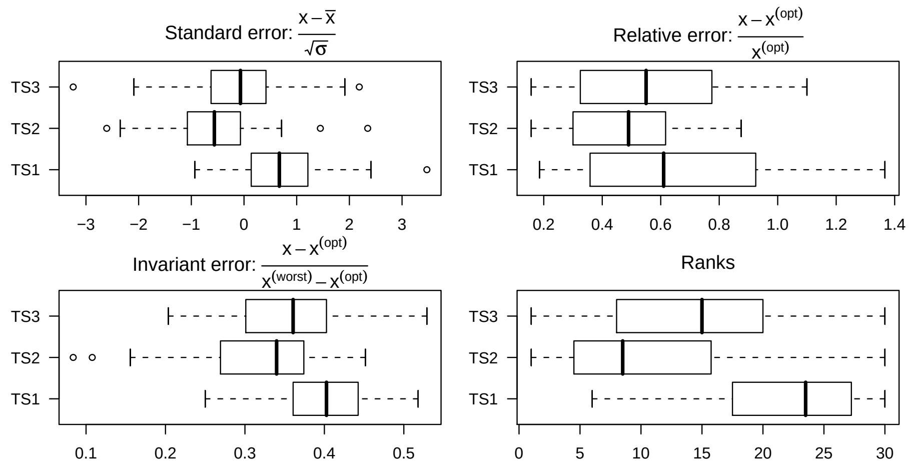
Scenario:

- 3 heuristics $\mathcal{A}_1^\infty$, $\mathcal{A}_2^\infty$, $\mathcal{A}_3^\infty$ on class $\mathcal{C}_\Pi$.
  (Or 3 heuristics $\mathcal{A}_1^\infty$, $\mathcal{A}_2^\infty$, $\mathcal{A}_3^\infty$ on class $\mathcal{C}_\Pi$ without interest in computation time because negligible or comparable)
- homogeneous instances (no data transformation) or heterogeneous (data transformation)
- 1 or r runs per instance
- a priori time limit imposed
- Interest: inspecting solution cost

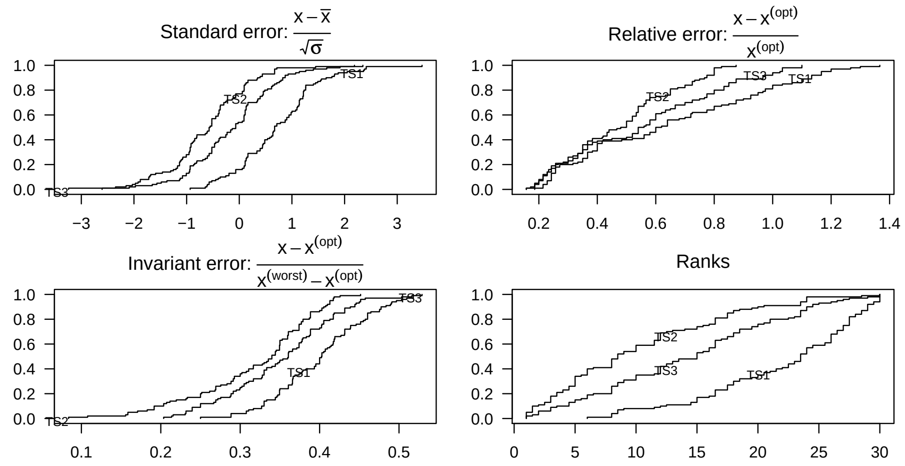
Tools:

- Histograms (summary measures: mean or median or mode?)
- Boxplots
- Empirical cumulative distribution functions (ECDFs)

On a class of instances



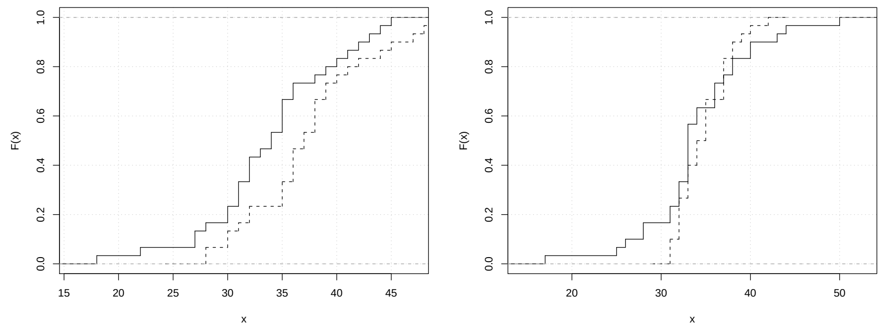
On a class of instances



Stochastic Dominance

Definition: Algorithm $\mathcal{A}_1$ probabilistically dominates algorithm $\mathcal{A}_2$ on a problem instance, iff its CDF is always "below" that of $\mathcal{A}_2$, *i.e.*:

$$F_1(x) \leq F_2(x), \qquad \forall x \in X$$



R code behind the previous plots

We load the data and plot the comparative boxplot for each instance.

```
> load("TS.class-G.dataR")
> G[1:5,]
  alg inst run sol time.last.imp tot.iter parz.iter exit.iter exit.time opt
1 TS1 G-1000-0.5-30-1.1.col 1 59 9.900619 5955 442 5955 10.02463 30
2 TS1 G-1000-0.5-30-1.1.col 2 64 9.736608 3880 130 3958 10.00062 30
3 TS1 G-1000-0.5-30-1.1.col 3 64 9.908618 4877 49 4877 10.03263 30
4 TS1 G-1000-0.5-30-1.1.col 4 68 9.948622 6996 409 6996 10.07663 30
5 TS1 G-1000-0.5-30-1.1.col 5 63 9.912620 3986 52 3986 10.04063 30
>
> library(lattice)
> bwplot(alg ~ sol | inst,data=G)
```

If we want to make an aggregate analysis we have the following choices:

- maintain the raw data,
- transform data in standard error,
- transform the data in relative error,
- transform the data in an invariant error,
- transform the data in ranks.

Maintain the raw data

```
> par(mfrow=c(3,2),las=1,font.main=1,mar=c(2,3,3,1))
> #original data
> boxplot(sol~alg,data=G,horizontal=TRUE,main="Original data")
```

Transform data in standard error

```
> #standard error
> T1 <- split(G$sol,list(G$inst))
> T2 <- lapply(T1,scale,center=TRUE,scale=TRUE)
> T3 <- unsplit(T2,list(G$inst))
> T4 <- split(T3,list(G$alg))
> T5 <- stack(T4)
> boxplot(values~ind,data=T5,horizontal=TRUE,main=expression(paste("
    Standard error: ",frac(x-bar(x),sqrt(sigma)))))
> library(latticeExtra)
> ecdfplot(~values,group=ind,data=T5,main=expression(paste("Standard error:
",frac(x-bar(x),sqrt(sigma)))))

> #standard error
> G$scale <- 0
> split(G$scale, G$inst) <- lapply(split(G$sol, G$inst), scale,center=TRUE,
    scale=TRUE)
```


Transform the data in relative error

```
> #relative error
> G$err2 <- (G$sol-G$opt)/G$opt
> boxplot(err2~alg,data=G,horizontal=TRUE,main=expression(paste("Relative
    error: ",frac(x-x^(opt),x^(opt)))))
> ecdfplot(G$err2,group=G$alg,main=expression(paste("Relative error: ",frac
    (x-x^(opt),x^(opt)))))
```


Transform the data in an invariant error
We use as surrogate of x^{worst} the median solution returned by the simplest algorithm for the graph coloring, that is, the ROS heuristic.

```
> #error 3
> load("ROS.class-G.dataR")
> F1 <- aggregate(F$sol,list(inst=F$inst),median)
> F2 <- split(F1$x,list(F1$inst))
> G$ref <- sapply(G$inst,function(x) F2[[x]])
> G$err3 <- (G$sol-G$opt)/(G$ref-G$opt)
> boxplot(err3~alg,data=G,horizontal=TRUE,main=expression(paste("Invariant
    error: ",frac(x-x^(opt),x^(worst)-x^(opt)))))
> ecdfplot(G$err3,group=G$alg,main=expression(paste("Invariant error: ",
    frac(x-x^(opt),x^(worst)-x^(opt)))))
```


Transform the data in ranks

```
> #rank
> G$rank <- G$sol
> split(G$rank, G$inst) <- lapply(split(G$sol, D$inst), rank)
> boxplot(rank~alg,data=G,horizontal=TRUE,main="Ranks")
> ecdfplot(rank,group=alg,data=G,main="Ranks")
```

Scenario B

Asymptotic heuristics

There are two approaches:

1.b. Solution quality as an external parameter decided *a priori*. The algorithm is halted when quality is reached.

Deterministic case: $\mathcal{A}^\infty$ on class C_Π finds a solution in running time t.

The performance of $\mathcal{A}^\infty$ on class C_Π is the scalar $y = t$.

Randomized case: $\mathcal{A}^\infty$ on class C_Π finds a solution in running time T, where T is a random variable.

The performance of $\mathcal{A}^\infty$ on class C_Π is the univariate $Y = T$.

Dealing with Censored Data

Asymptotic heuristics, Approach 1.b

- ▷ Heuristic $\mathcal{A}^\dashv$ stopped before completion or $\mathcal{A}^\infty$ truncated (always the case)
- ▶ Interest: determining whether a prefixed goal (optimal/feasible) has been reached

The computational effort to attain the goal can be specified by a cumulative distribution function $F(t) = P(T < t)$ with T in $[0, \infty)$.

If in a run i we stop the algorithm at time L_i then we have a Type I right censoring, that is, we know either

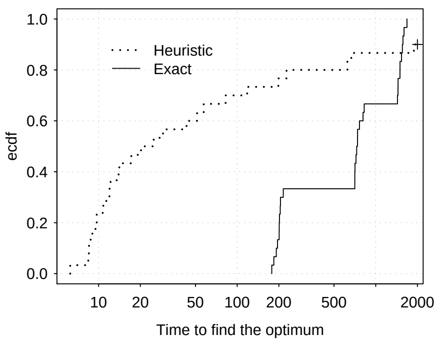
- T_i if $T_i \leq L_i$
- or $T_i \geq L_i$.

Hence, for each run i we need to record $\min(T_i, L_i)$ and the indicator variable for observed optimal/feasible solution attainment, $\delta_i = I(T_i \leq L_i)$.

Example

Asymptotic heuristics, Approach 1.b: Example

- ▷ An exact *vs* an heuristic algorithm for the *2-edge-connectivity augmentation problem*.
- ▶ Interest: time to find the optimum on different instances.



Uncensored:

$$F(t) = \frac{\#\text{ runs} < t}{n}$$

Censored:

$$F(t) = \frac{\#\text{ runs} < t}{n}$$

Scenario B

Asymptotic heuristics

There are two approaches:

2. Cost dependent on running time:

Deterministic case: $\mathcal{A}^\infty$ on π returns a current best solution x at each observation in $t_1, \ldots, t_k$.

The performance of $\mathcal{A}^\infty$ on π is the profile indicated by the vector $\vec{y} = \{x(t_1), \ldots, x(t_k)\}$.

Randomized case: $\mathcal{A}^\infty$ on π produces a monotone stochastic process in solution cost $X(\tau)$ with any element dependent on the predecessors.

The performance of $\mathcal{A}^\infty$ on π is the multivariate $\vec{Y} = (X(t_1), X(t_2), \ldots, X(t_k))$.

Example

Scenario:

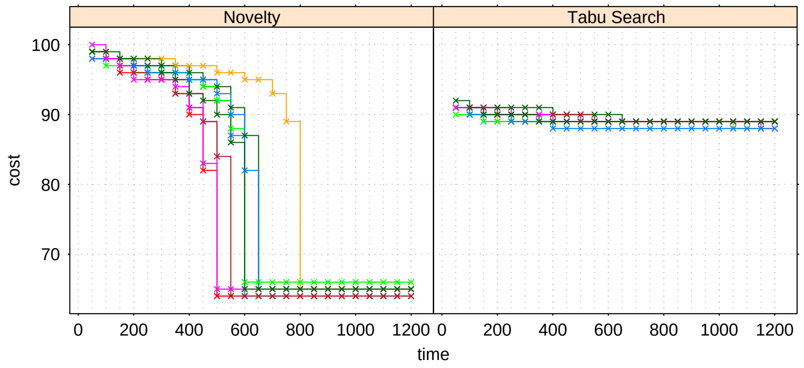
- ▷ 3 heuristics $\mathcal{A}_1^\infty$, $\mathcal{A}'^\infty_2$, $\mathcal{A}_3^\infty$ on instance π.
- ▷ single instance hence no data transformation.
- ▷ r runs
- ▶ Interest: inspecting solution cost over running time to determine whether the comparison varies over time intervals

Tools:

- Quality profiles

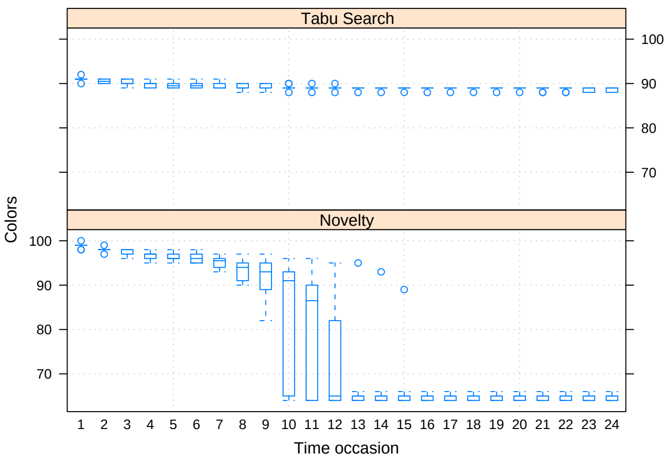
The performance is described by multivariate random variables of the kind $\vec{Y} = \{Y(t_1), Y(t_2), \dots, Y(l_k)\}$.

Sampled data are of the form $\vec{Y}_i = \{Y_i(t_1), Y_i(t_2), \dots, Y_i(t_k)\}$, $i = 1, \dots, 10$ (10 runs per algorithm on one instance)



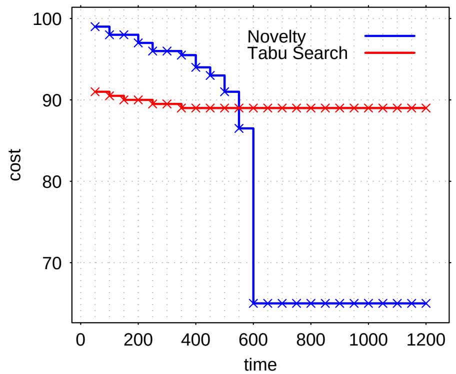
The performance is described by multivariate random variables of the kind $\vec{Y} = \{Y(t_1), Y(t_2), \dots, Y(l_k)\}$.

Sampled data are of the form $\vec{Y}_i = \{Y_i(t_1), Y_i(t_2), \dots, Y_i(t_k)\}$, $i = 1, \dots, 10$ (10 runs per algorithm on one instance)



The performance is described by multivariate random variables of the kind $\vec{Y} = \{Y(t_1), Y(t_2), \dots, Y(l_k)\}$.

Sampled data are of the form $\vec{Y}_i = \{Y_i(t_1), Y_i(t_2), \dots, Y_i(t_k)\}$, $i = 1, \dots, 10$ (10 runs per algorithm on one instance)



The median behavior of the two algorithms

Summary

Graph your data for your analysis and for communication to others

Explore your data:

- make plots: histograms, boxplots, empirical cumulative distribution functions, correlation/scatter plots
- look at the numerical data and interpret them in practical terms: computation times, distance from optimum
- look for patterns

All the above both at a single instance level and at an aggregate level.

Graphing Data

[W.S. Cleveland. The Elements of Graphing Data. Wadsworth Advanced Books and Software, 1985]

- Principles of graph construction
  terminology
- Graphical methods
- Graphical perception
  study of how people decode quantitative information from graphs

Making Plots

http://algo2.iti.uni-karlsruhe.de/sanders/courses/bergen/bergenPresenting.pdf

[?]

- Should the experimental setup from the exploratory phase be redesigned to increase conciseness or accuracy?
- What parameters should be varied? What variables should be measured?
- How are parameters chosen that cannot be varied?
- Can tables be converted into curves, bar charts, scatter plots or any other useful graphics?
- Should tables be added in an appendix?
- Should a 3D-plot be replaced by collections of 2D-curves?
- Can we reduce the number of curves to be displayed?
- How many figures are needed?
- Should the x-axis be transformed to magnify interesting subranges?

- Should the x-axis have a logarithmic scale? If so, do the x-values used for measuring have the same basis as the tick marks?
- Is the range of x-values adequate?
- Do we have measurements for the right x-values, i.e., nowhere too dense or too sparse?
- Should the y-axis be transformed to make the interesting part of the data more visible?
- Should the y-axis have a logarithmic scale?
- Is it misleading to start the y-range at the smallest measured value? (if not too much space wasted start from 0)
- Clip the range of y-values to exclude useless parts of curves?
- Can we use banking to 45^o?
- Are all curves sufficiently well separated?
- Can noise be reduced using more accurate measurements?
- Are error bars needed? If so, what should they indicate? Remember that measurement errors are usually not random variables.

- Connect points belonging to the same curve.
- Only use splines for connecting points if interpolation is sensible.
- Do not connect points belonging to unrelated problem instances.
- Use different point and line styles for different curves.
- Use the same styles for corresponding curves in different graphs.
- Place labels defining point and line styles in the right order and without concealing the curves.
- Give axis units
- Captions should make figures self contained.
- Give enough information to make experiments reproducible.
- Golden ratio rule: make the graph wider than higher [Tufte 1983].
- Rule of 7: show at most 7 curves (omit those clearly irrelevant).
- Avoid: explaining axes, connecting unrelated points by lines, cryptic abbreviations, microscopic lettering, pie charts

Suggested Reading

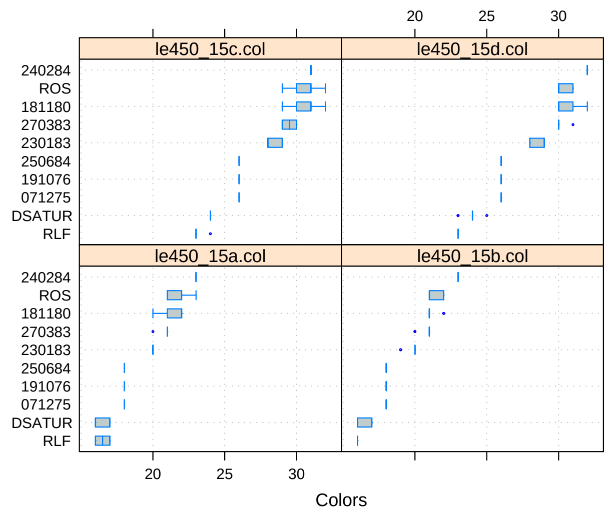
Outline

2007 competition

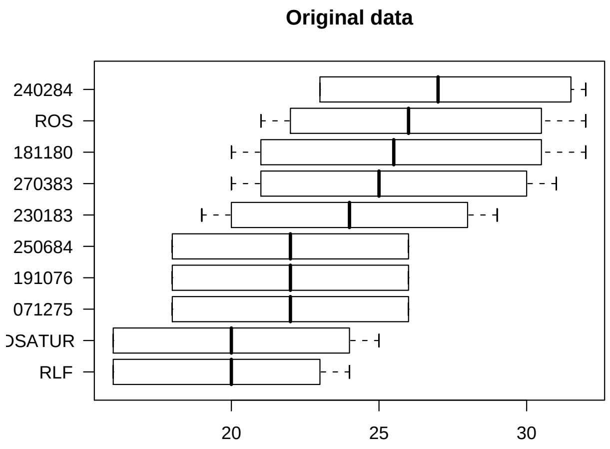
- Graph Coloring Problem
- Task 1: submit a construction heuristic
  Set of instances A: 4 instances
- Task 2: submit an algorithm derived from the use of a metaheuristic for construction heuristics
  Time limit for each single run: 90 seconds
  Set of instances B: 15 instances
- Task 3: a peak performance algorithm
  Time limit for each single run: 360 seconds
  Set of instance C: The instances in the set are generated in order to admit different kind of colorings, ranging from equi-partite classes to highly variable classes.

Comparative Analysis

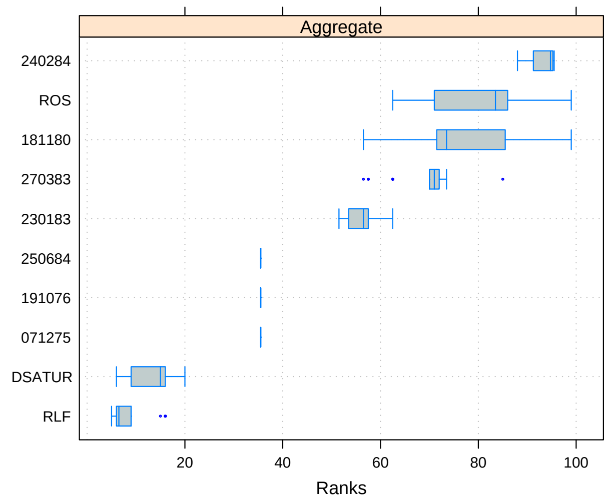
View of raw data within each instance



View of raw data aggregated for the 4 instances

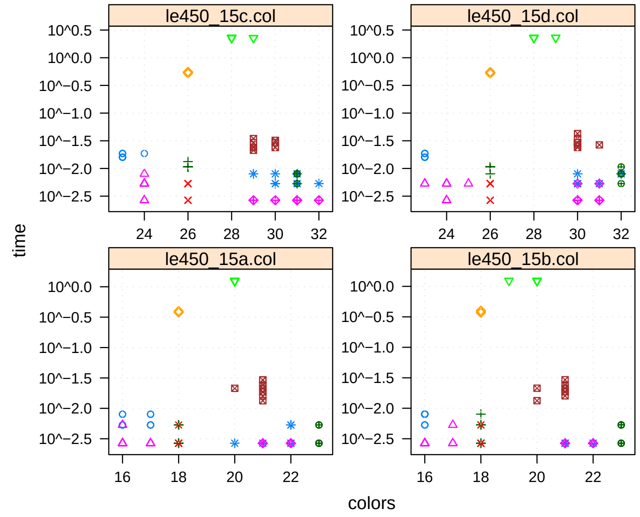


View of raw data ranked within instances and aggregated for the 4 instances

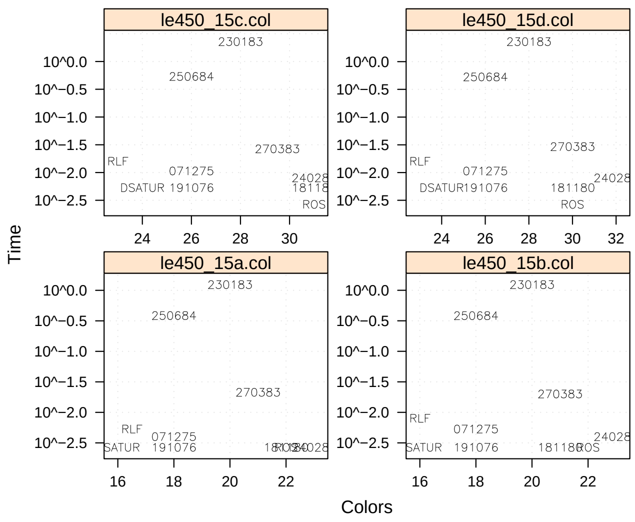


Trade off Solution-Quality *vs* Run-Time

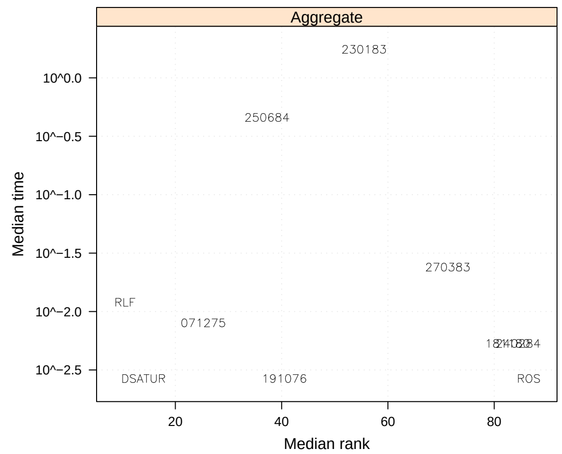
The trade off computation time *vs* sol quality. Raw data.



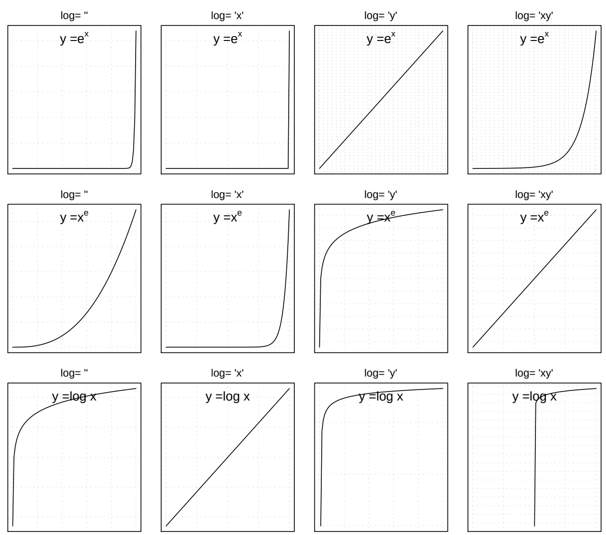
The trade off computation time *vs* sol quality. Raw data.



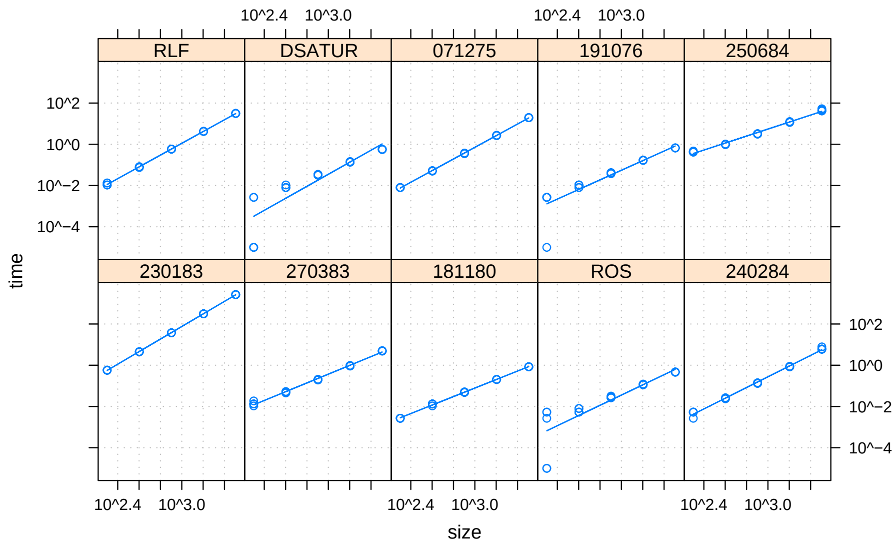
The trade off computation time *vs* sol quality.
Solution quality ranked within the instances and computation time in raw terms



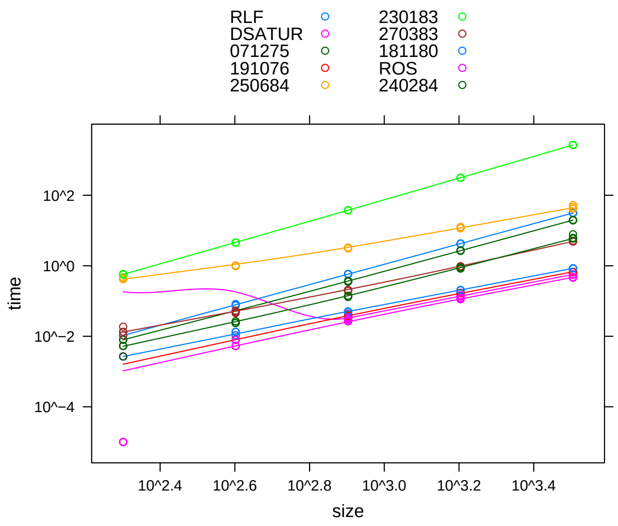
Scaling Analysis



Linear regression in log-log plots ⇒ polynomial growth



Comparative visualization



Numerical data

Size	071275	181180	191076	230183	240284	250684	270383
200	0.008	0.00267	0.00267	0.5787	0.00533	0.42933	0.01333
400	0.05067	0.01333	0.01067	4.5443	0.024	0.98667	0.05067
800	0.36002	0.05067	0.04	37.68	0.13868	3.2313	0.2
1600	2.7175	0.20268	0.16801	313.27	0.85339	11.709	0.96267
3200	19.711	0.84805	0.66937	2674.8	6.1524	42.287	4.9413

Size	DSATUR	RLF	ROS
200	0	0.01067	0.00267
400	0.008	0.07734	0.00533
800	0.032	0.58404	0.02667
1600	0.13601	4.2563	0.11467
3200	0.5627	31.519	0.46936

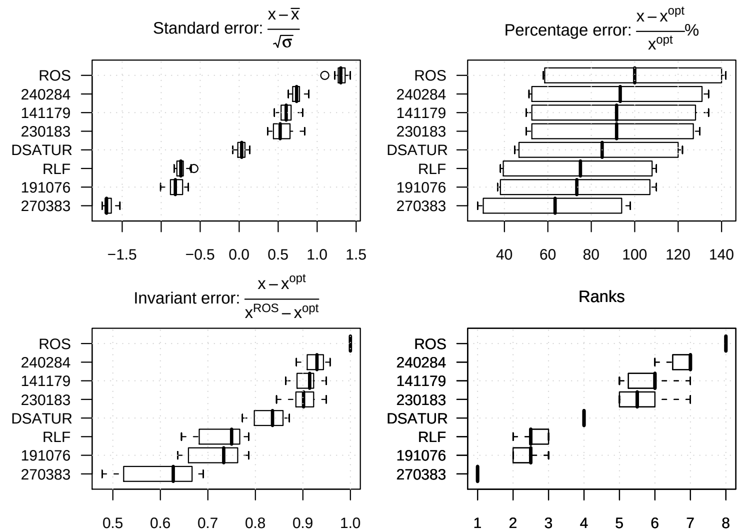
Experimental Setup

- 15 new flat instances created

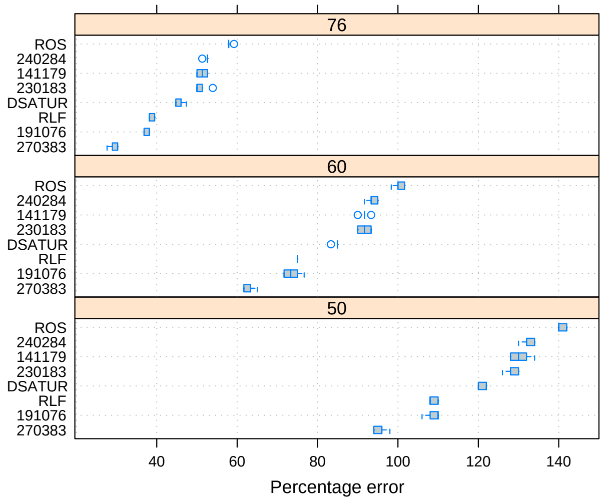
Type	# instances	Upper bound
flat-1000-50-0-?.col	5	50
flat-1000-60-0-?.col	5	60
flat-1000-76-0-?.col	5	76

- each algorithm run once on each of the 15 new instances
- fairness principle: same computational resources to all algorithms
  $\Rightarrow$ 90 seconds on Intel(R) Celeron(R) CPU 2.40GHz, 1GB RAM (120 seconds for 230183)
- restart ROS heuristic used as reference algorithm
- restart RLF and DSATUR also included

Results



Results



Results

Algorithm	flat-1000-50	flat-1000-60	flat-1000-76
270383	98	98	99
191076	105	104	105
RLF	104	105	105
DSATUR	111	111	111
230183	114	115	114
141179	115	115	115
240284	116	116	116
ROS	120	120	120

Results