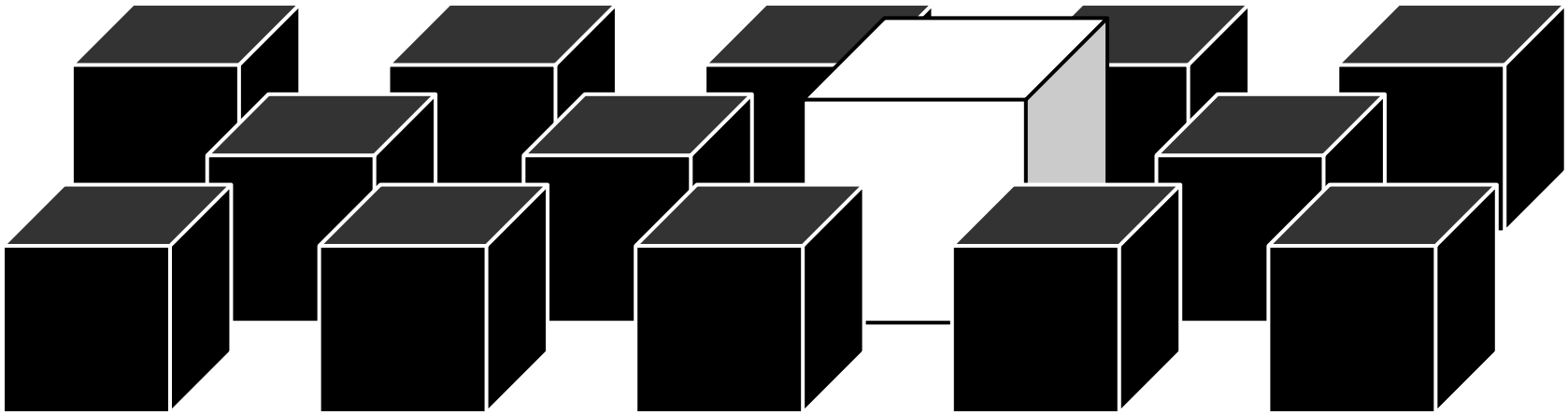


Panobbgo: Framework for Parallel Black-Box Global Optimization

Mag. Harald Schilly

CST2014, 2014-09-05

Overview

“Panobbgo” - software framework for optimizing black-box functions leveraging parallelization.

1. **Motivation**: Why, which problems does it tackle?
2. **Idea**: Which techniques drive its fruition.
3. **Implementation**: Architectural overview.

Global Black-Box Opt.

Minimization of $f(\hat{x}) \rightarrow \mathbb{R}$

with the parameter vector: $\hat{x} \in \mathbb{R}^n$

bounded in a **finite box**: $\hat{x}_i \in [\underline{x_i}, \overline{x_i}]$

optional constraints: $\hat{g}(\hat{x}) \leq 0$

... and almost nothing is known about f and g.

Existing Derivative Free Solvers

- Tomlab, Global, MCS, Snobfit, CMA-ES, BOBYAO, PSWARM, DFO, ...
- They constitute a diverse set of software.
- Build on similar ideas, but are independent.

Ideas ⇒ Heuristics

- **Local**: Simplex, Pattern Search, Local Model-based & Trust-region methods, Implicit filtering, ...
- **Global**: Lipschitzian-based methods, Branch&Bound partitioning techniques, Global model-based methods, ...
- **Stochastic**: Simulated Annealing, Genetic Algorithms, Particle Swarms, ...

(according to Rios&Sahinidis, 2012)

Heuristics /2

In practice, almost all solvers use a mix of techniques.

e.g.

- global deterministic sampling + local search
- branch&bound + particle swarm
- ...

Parallelization

Almost all existing solvers are single-threaded.

Parallelization: running tasks at the same time, especially focus on expensive objective functions.

Parallelization /2

Parallelization is usually done naively:

e.g. running several instances of the same solver with different starting points.

Drawback: limited flow of information between instances!

Opportunity

Concurrent programming:

How to actually conduct parallelization.

These methods allow sophisticated control of parallelization and richer exchange of information.

Opportunity /2

An open **modularized approach** has potential to:

- foster experimentation
- rapid innovation
- improved leveraging of parallel hardware.

Opportunity /2

Foster experimentation: modification of an existing - or introduction of a new - **module** is a small localized change.

Rapid innovation: Reuse benchmark, tests & reporting tools, ...

Leveraging of parallelization: the underlying concurrent programming model is the driver for all modules.

Components

During the design of an initial version and development, three main types of components crystallized:

- **Analyzers**
- **Heuristics**
- **Strategies**

… held together by a threaded **EventBus**

EventBus

A threaded **EventBus** coordinates all components via a **double-indirection scheme**.

Register your interest, you will be called back.

Publish information without defining recipients.

EventBus /2

API

- **publish**: send objects to all listeners
  `eventbus.publish('<key>', <payload>)`

- **listen** via `on_<key>()` callbacks.
  `def on_<key>(self, <payload>): ...`

Component: Analyzer

Analyzers loosely rely on each other.

- **reacts** to new events;
- modifies its **internal state** based on the new input (e.g. calculating statistics);
- emits new data objects as **events**;
- **exposes information** via methods.

Analyzer: Best

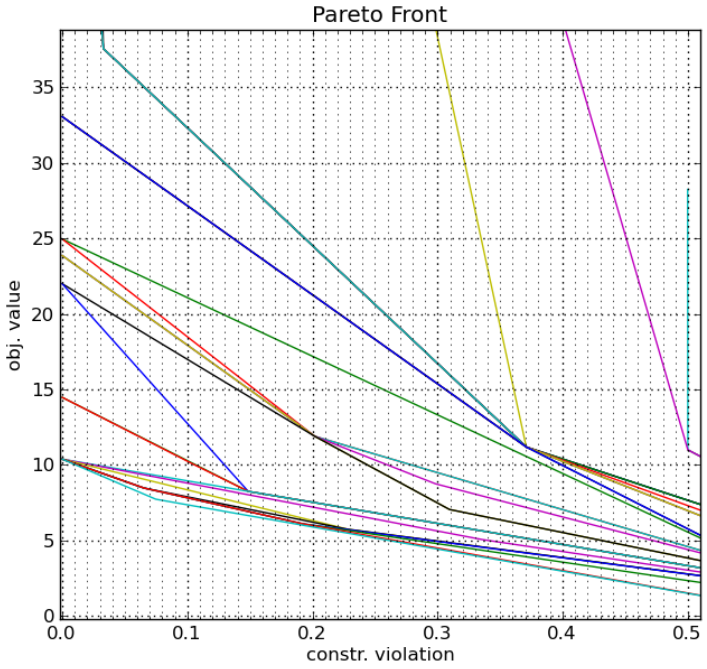
The "Best"-Analyzer is very simple.

It filters all new results and stores the best point.

New best points are published in the '**`new_best`**' channel.

Analyzer: Best /2

- **pareto-front** of obj. f. vs. constr. viol.



Analyzer: Grid

The "Grid"-Analyzer collects all results in an n-dimensional grid.

Statistics for each grid are tabulated, ...

Component: Heuristic

A ***Heuristic*** is a special Analyzer!

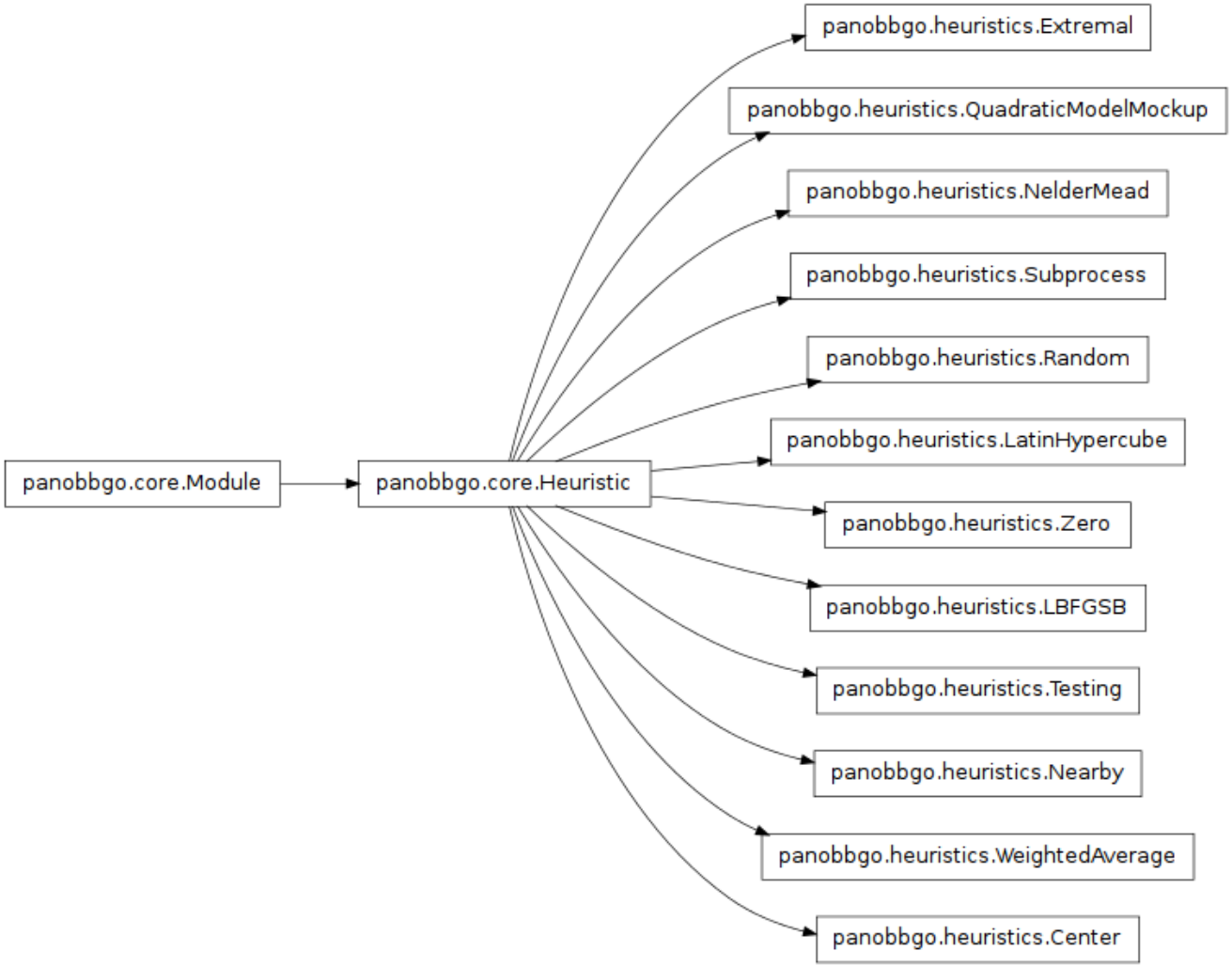
Has the ability to **emit <u>new search points</u>**.

For parallelization, it is essential to have several new and flavoured search points available at the same time.

Heuristic: Overview

- **Zero, Center**
- **Random**
- **Extremal**: searches the borders/edges
- **WeightedAverage**: estimation based on best point's neighbours.
- **WeightedLeastSquares**: Quadratic Model
- **NelderMead**: based on NM method
- **LatinHypercube**: random sampling method
- ...

Heuristic: Classes



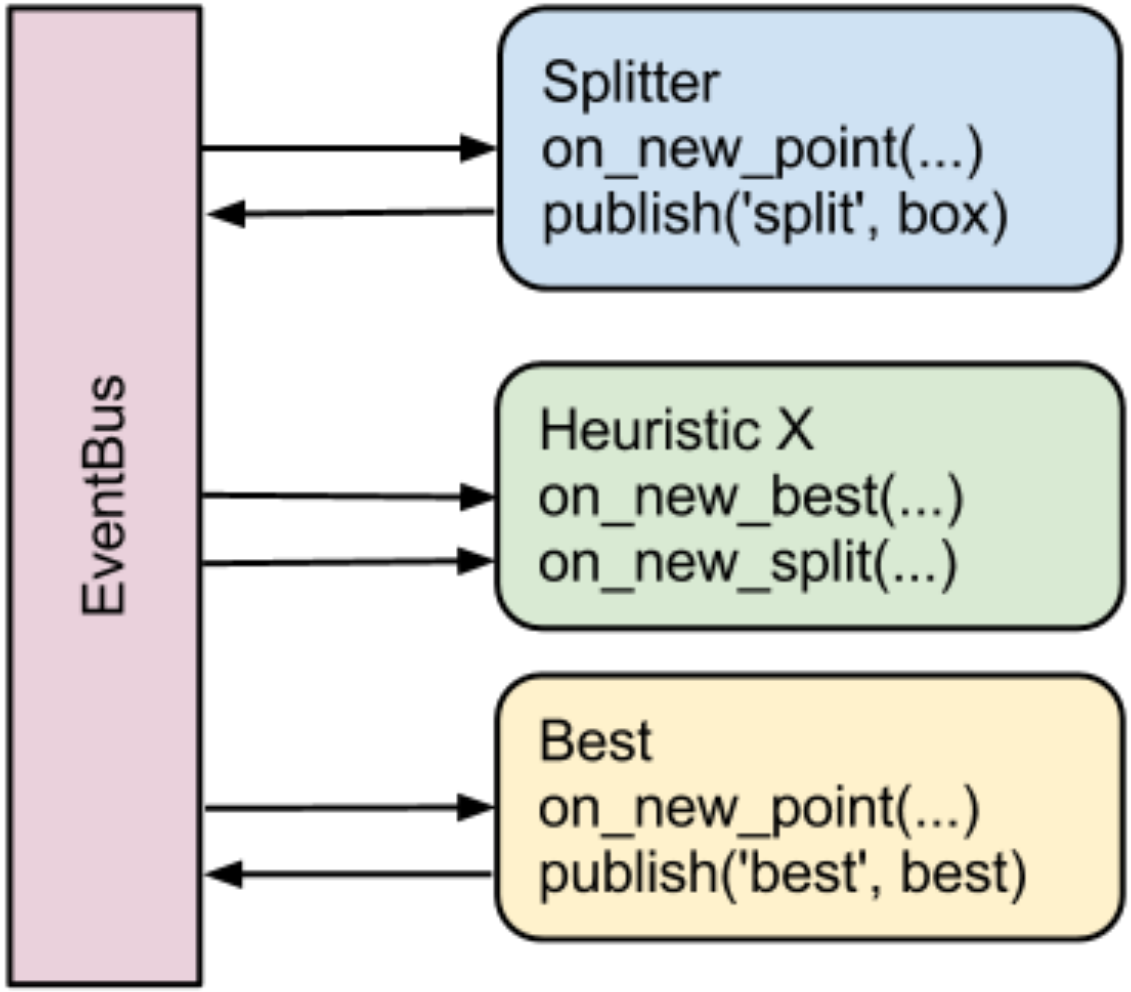
Component: Strategy

Selecting new points is a "**Strategy**".

It requests queued points from the heuristics.

E.g. **Rewarding**: heuristics are selected proportional to their recent relative improvements.

Components: Overview



API: High-Level Interface

① `problem = Himmelblau()`

② `strategy = StrategyRewarding(problem)`

```
strategy.add(Random)
strategy.add(LatinHypercube, div=5)
strategy.add(Nearby, radius=.1, new=3)
strategy.add(Extremal)
strategy.add(Center)
strategy.add(WeightedAverage)
```

} Heuristics ③

④ `strategy.start()`

⑤ `print "best point:", strategy.best`

High-Level Output Logging w/ focus

```
0.001657 config:114      CONFG INFO     cmdln options: Namespace(capacity=None,...)
0.003300 config:178      CONFG INFO    config.ini: {'core.smooth': '0.5', ...}
0.082599 config:196      CONFG INFO     IPython profile: default
0.082702 config:197      CONFG INFO     Environment: {'IPython': '0.13.1', ...}
0.000642 core:539        STRAT INFO    Init of 'StrategyRewarding' w/ 7 heuristics.
0.001013 core:541        STRAT INFO    Problem 'RosenbrockConstraint': 3 dims, params:
                                       {'par2': 0.5, 'par1': 10}, box: [[0.00 2.00], [-2.00 2.00], [-2.00 2.00]]
0.228702 core:748        STATS INFO        0 (   4) pnts | Tasks:   0 pend,   4 finished |
                                        0.470 [s] cpu,  0.228 [s] wall,  0.044 [s/task]
0.229909 strategies:90   STRAT INFO    ⌘  -39.000000 ≷  0.5000 @ [   1.000000    0.000000    0.000000] |
                                        Δ 1.0000000 Center
0.232063 strategies:90   STRAT INFO    ⌘   18.226507 ≷  1.3873 @ [   0.525005   -1.168365   -0.695210] |
                                        Δ 0.0000000 Latin Hypercube
0.232723 strategies:90   STRAT INFO    ⌘  -48.000000 ≷  0.7071 @ [   0.000000    0.000000    0.000000] |
                                        Δ 1.0000000 Zero
...
1.431347 strategies:90   STRAT INFO    ⌘  -36.561401 ≷  0.0000 @ [   0.450626    1.196734    2.000000] |
                                        Δ 0.2476225 Nearby 0.10
1.443912 core:715        STRAT INFO     Strategy 'StrategyRewarding' finished after 1.268 [s] and 12 loops.
best:  -36.561401 ≷  0.0000 @ [   0.450626    1.196734    2.000000]
```

High-Level Output Statistics

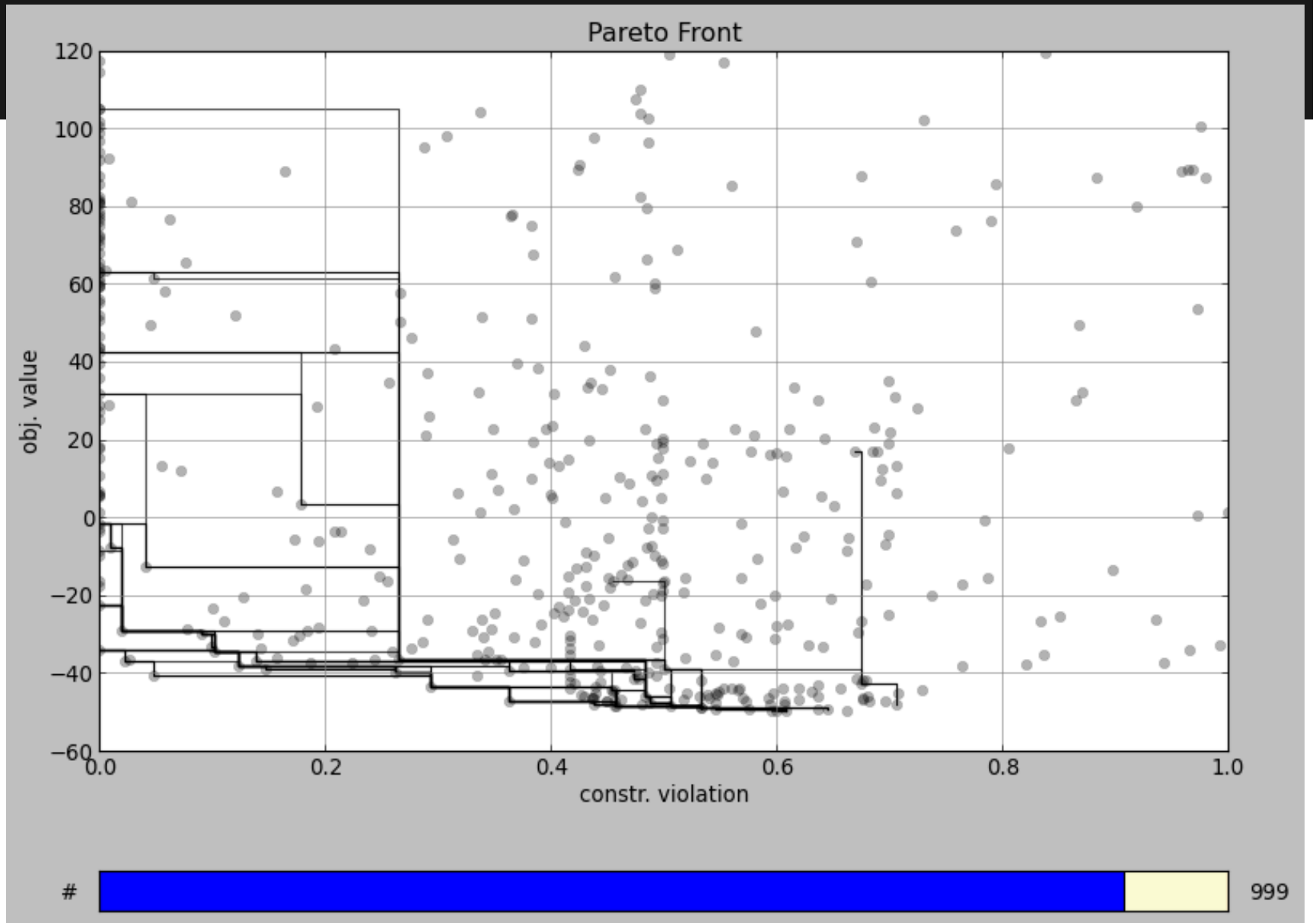
~1000 evaluations in **59** concurrent tasks:

- total execution time: **4.0955 [s]**
- Evaluation/Task: **0.025 [s/t]** up to **0.031 [s/t]**
- number of strategy evaluations: **11** (main loops)

~5000 evaluations in **200** concurrent tasks:

- **5.182 [s]** total
- **0.036 [s/task]** maximum
- **37** strategy evaluations

UI / constr-violat. vs. f(x)



Deliverables

- Open-source **framework** for global black-box optimization
- Well-**documented** functionality;
  written with **expressive code**
- Deep **modularization**
  - Easy to **extend** and **experiment**
- Flexible parallelization for **distributed computing**

Additional Parts

- **Dependency Management**: define explicit and implicit dependencies.
- **Problems**: several classical examples are already included, ...

Outlook

- **Hot-start mechanism**: it's desirable to be able to continue with a previous optimization.

- **Result Database**: storing optimization progress persistently and add utilities to analyze them (plot of performance progress, ...)

Thank you for your attention

Mag. Harald Schilly
harald.schilly@univie.ac.at
2014, University of Vienna, Austria