BFO: a simple "brute-force" optimizer and its self optimization

Philippe Toint

Department of Mathematics, University of Namur, Belgium

(philippe.toint@fundp.ac.be)

EWMINLP 2010, Marseille, April 2010

The problem

We consider the unconstrained nonlinear programming problem:

$$\text{minimize} \quad f(x)$$

for $x \in \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}$.

Important special case: the nonlinear least-squares problem

Additional constraints:

- there may be bounds on the variables
- derivatives are unavailable
- objective function possibly nonsmooth
- some variables can only assume discrete values

Motivation: parameter tuning in algorithm design (Audet-Orban), but many other examples...

Direct methods for optimization

A long history of direct search methods, including:

- Hookes-Jeeves, Nelder-Mead
- Coope-Price
- Dennis, Torczon, Lewis, Trosset (GPS)
- Dennis, Audet, Abramson (MADS).

This is one more of them

A very simple version:

BFO, a Brute-Force Optimizer

BFO basics

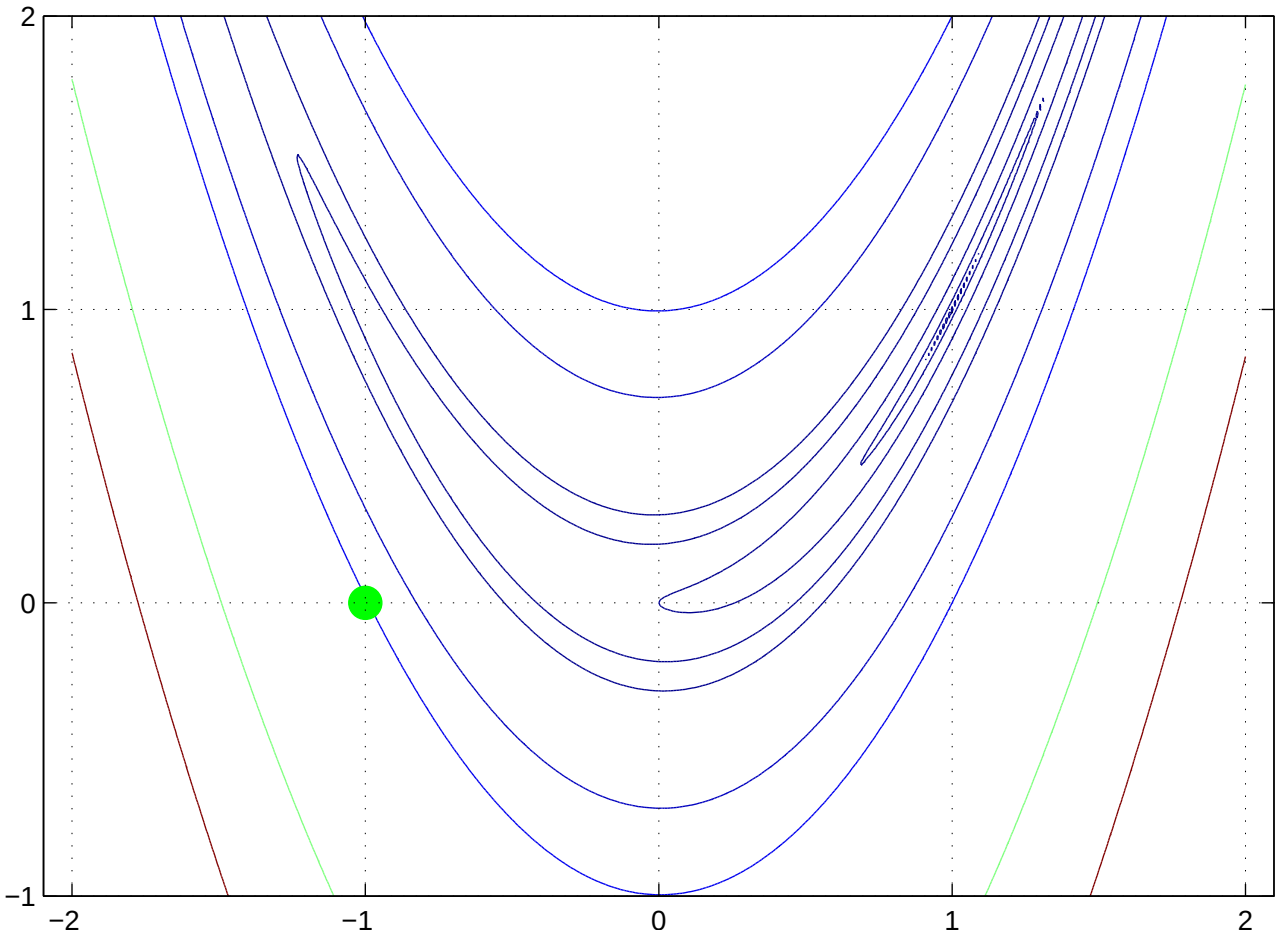
Main features: (not really original)

- Minimizes on progressively finer grids
- Some grid spacing remains fixed to user-specified discrete values
- Only for Local minimization...

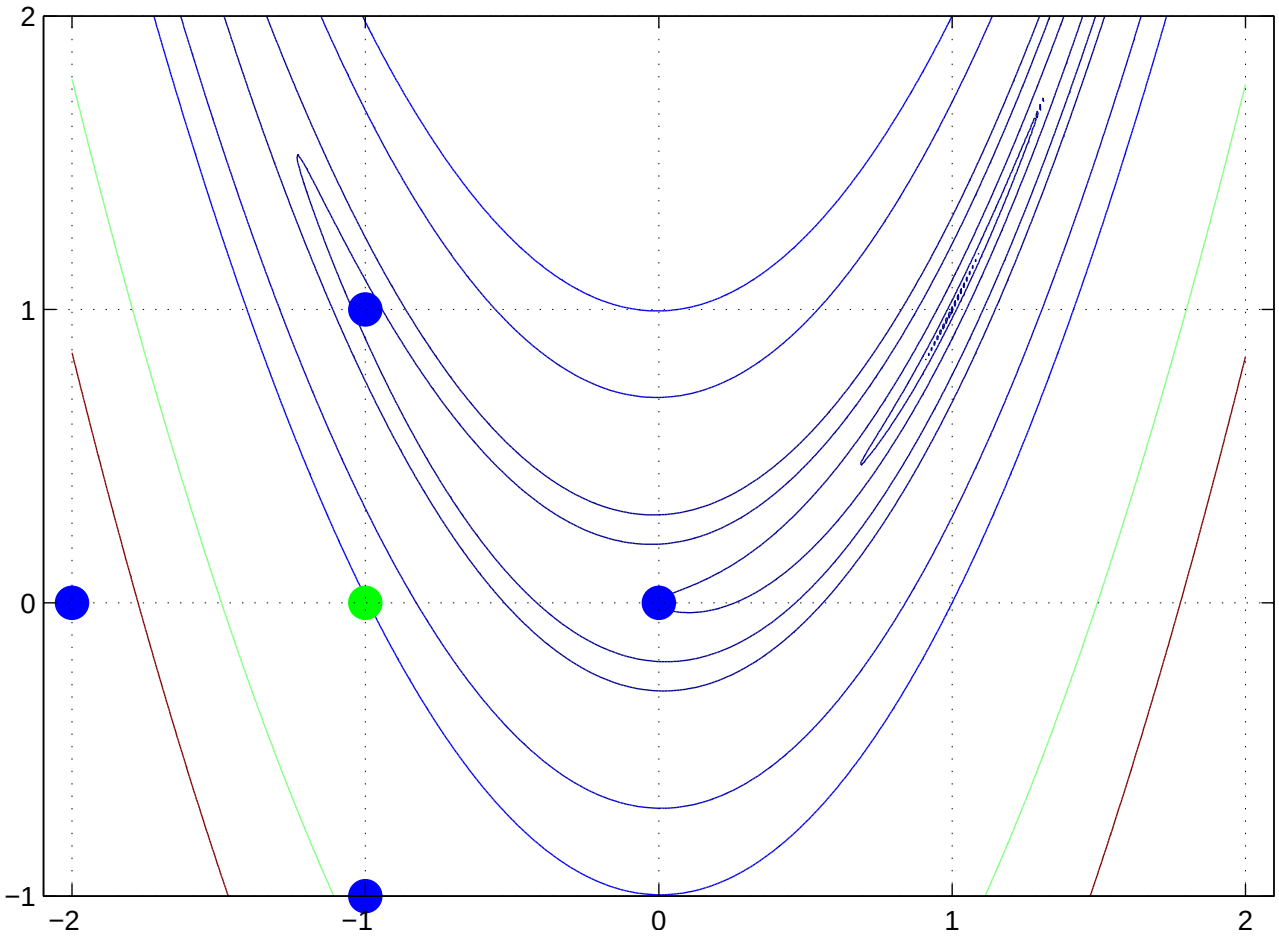
But also

- Attempts to learn from convergence history
- Includes some elements of random search
- User-friendly interface

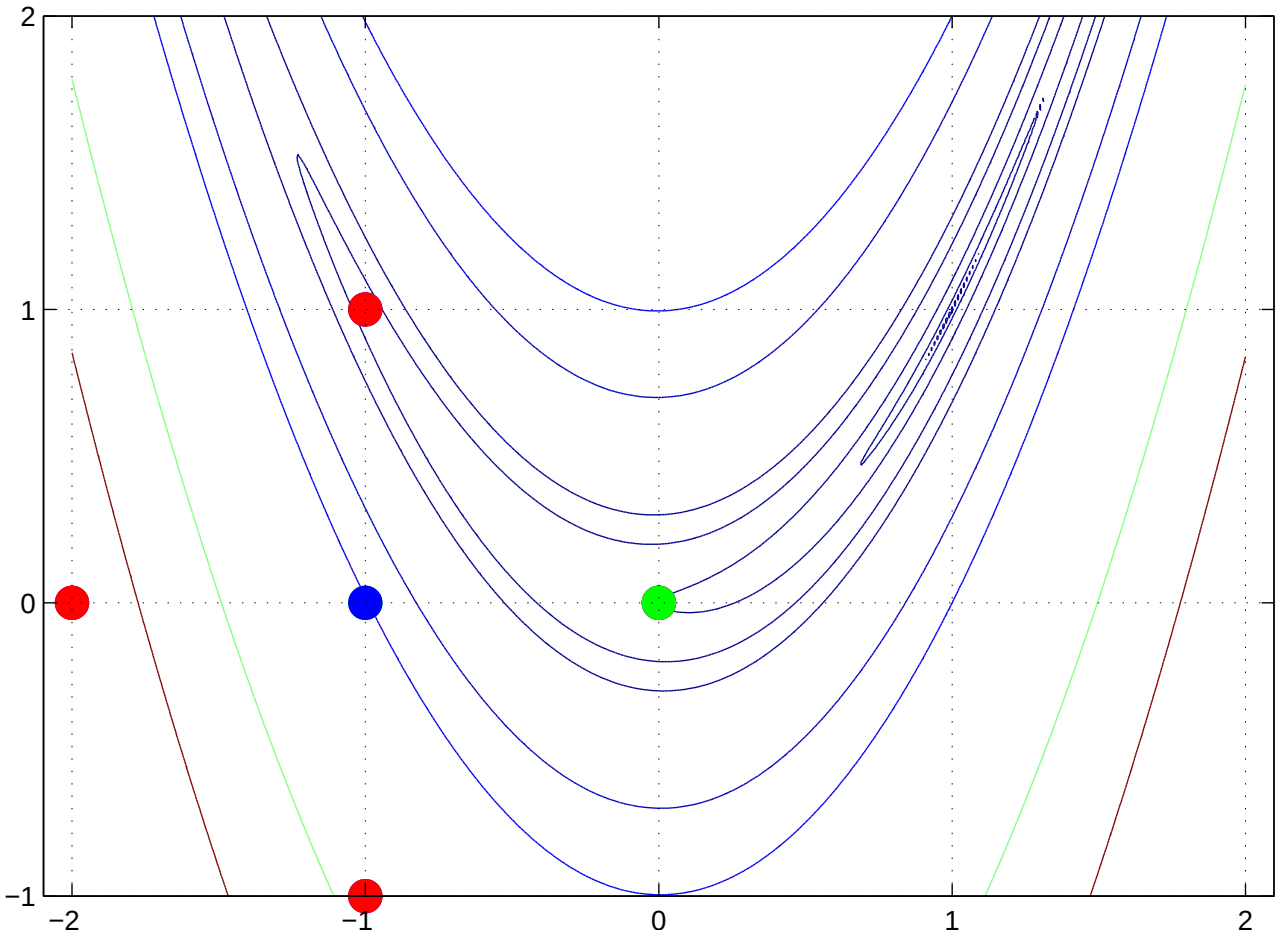
The compass-search on Rosenbrock's function



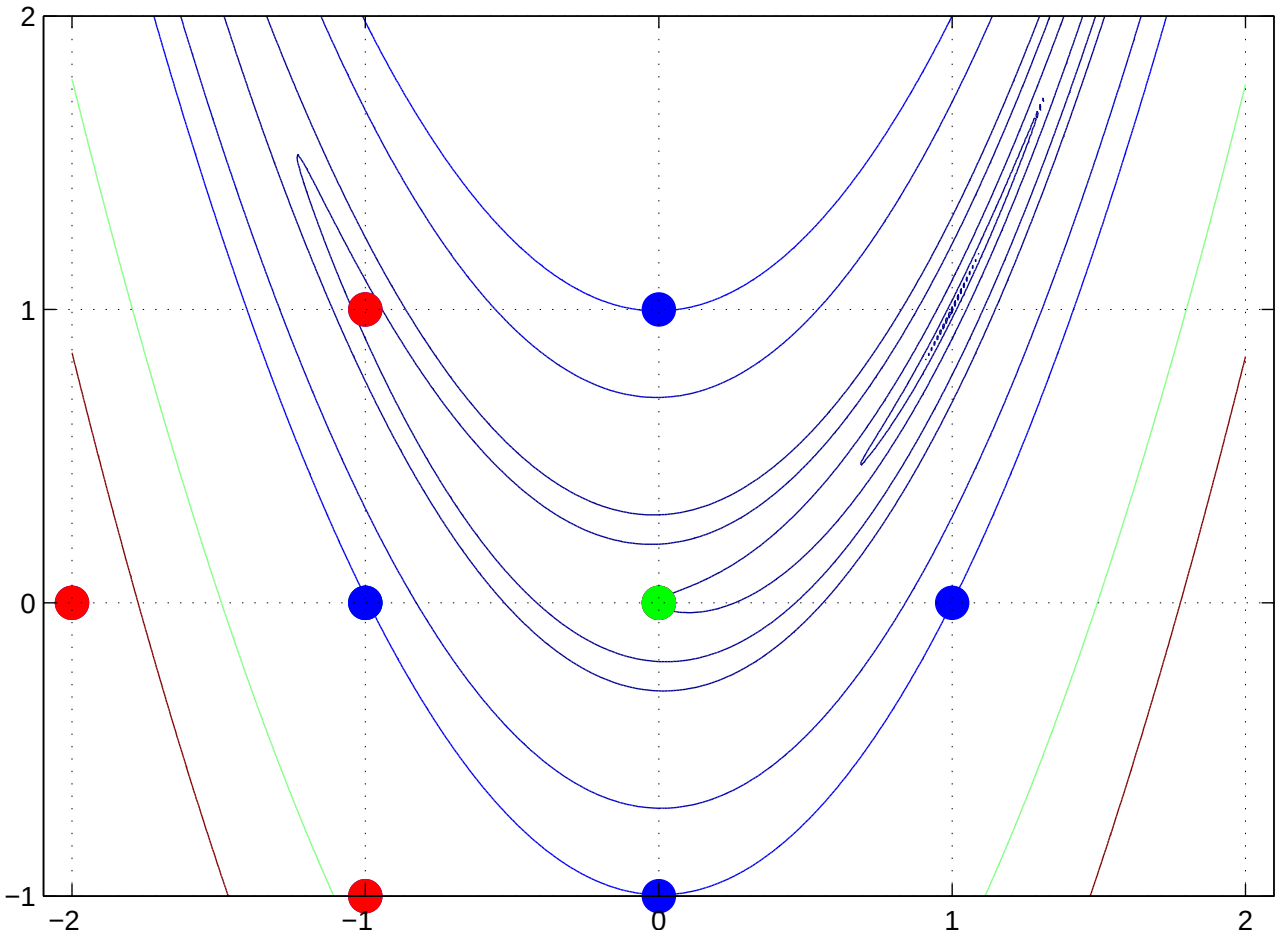
The compass-search on Rosenbrock's function



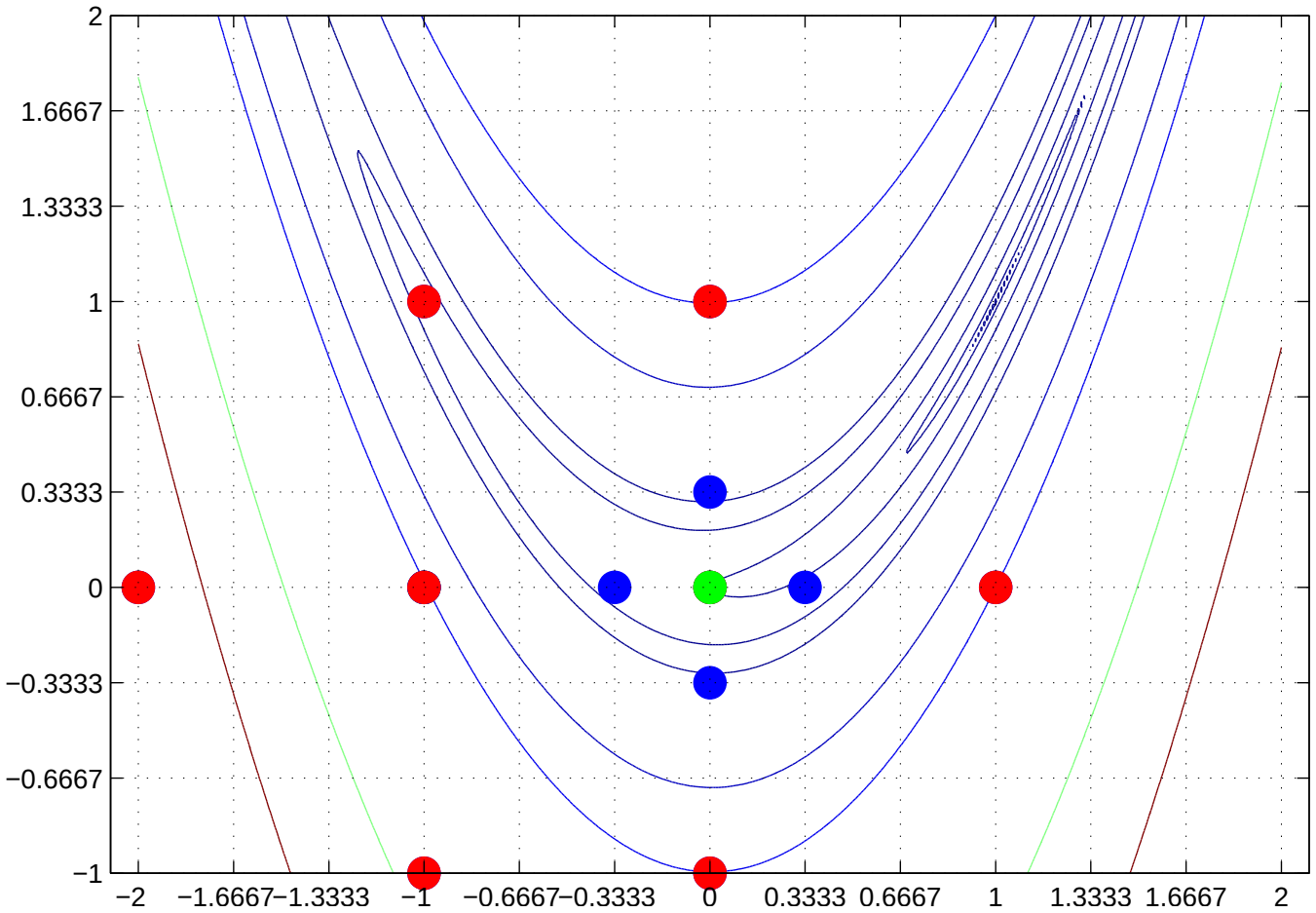
The compass-search on Rosenbrock's function



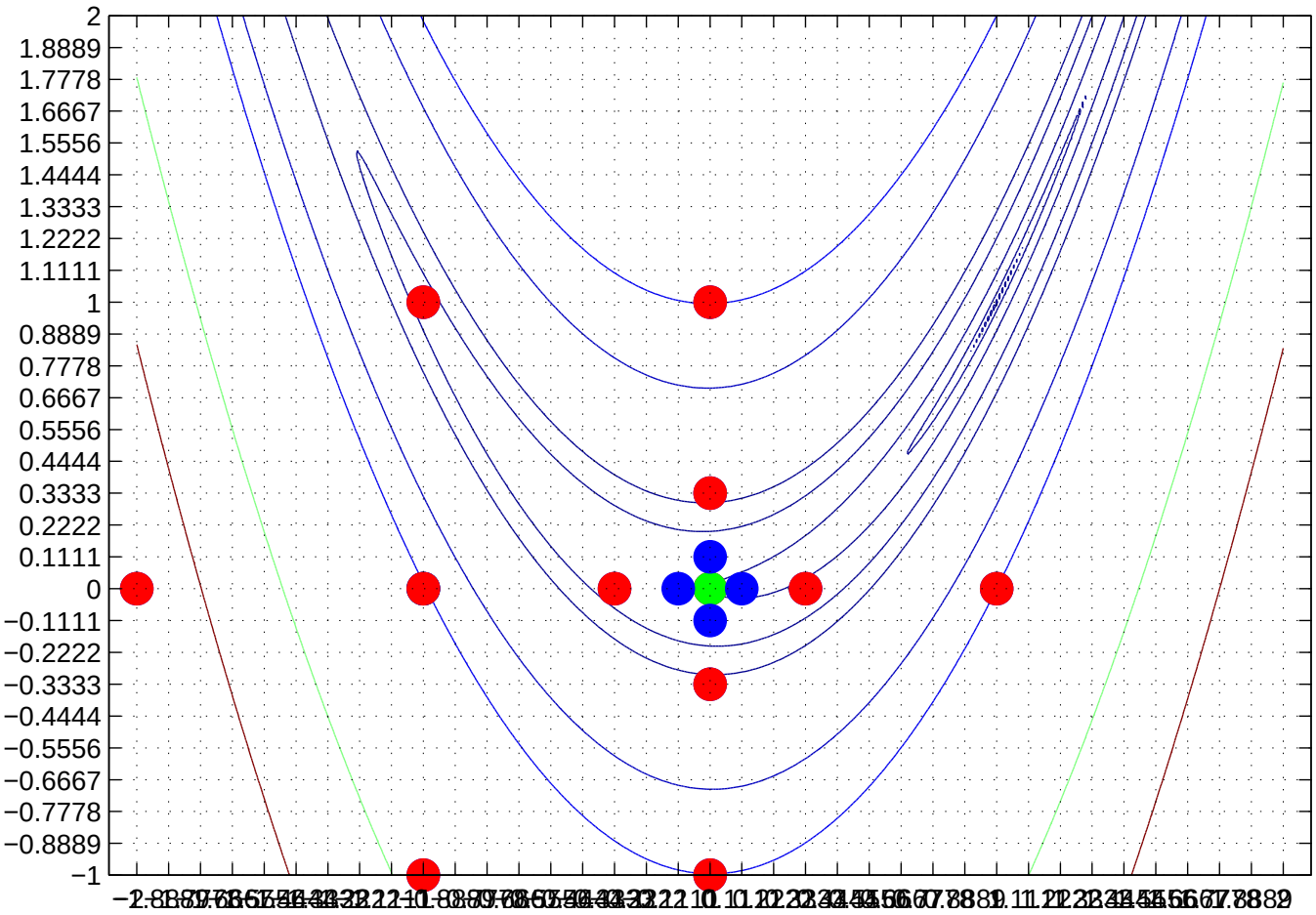
The compass-search on Rosenbrock's function



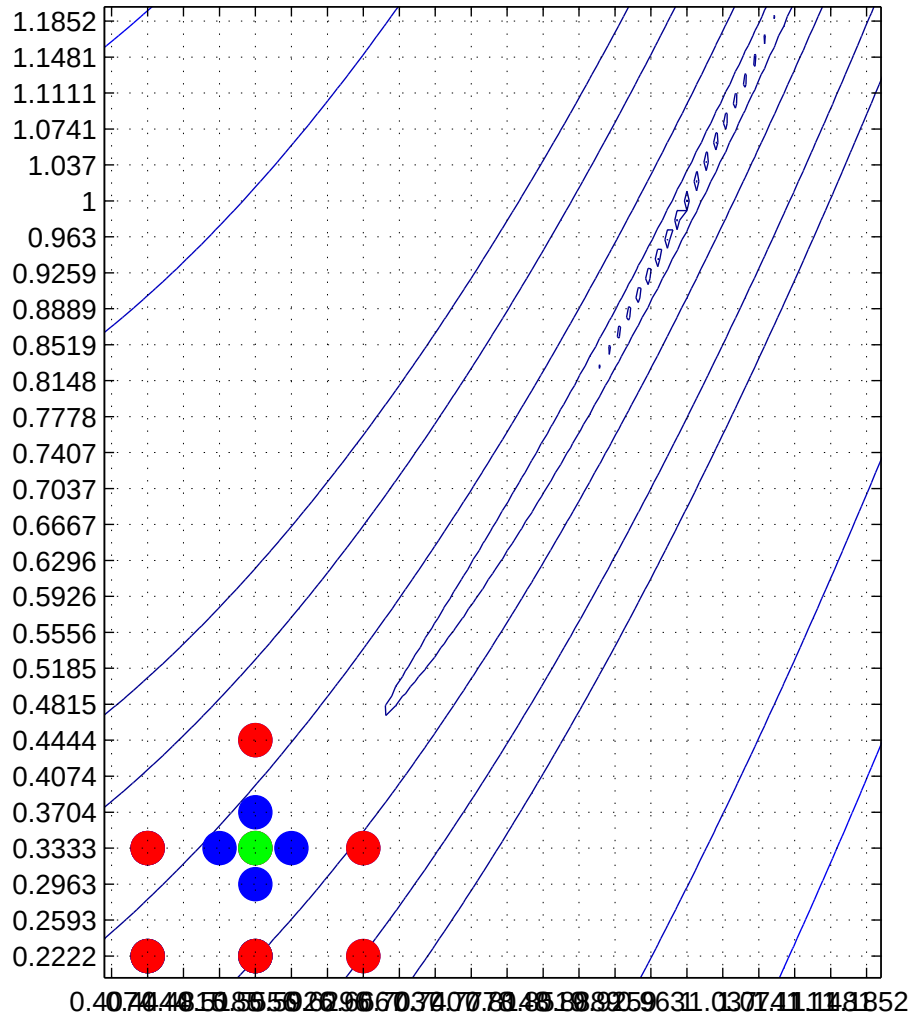
The compass-search on Rosenbrock's function



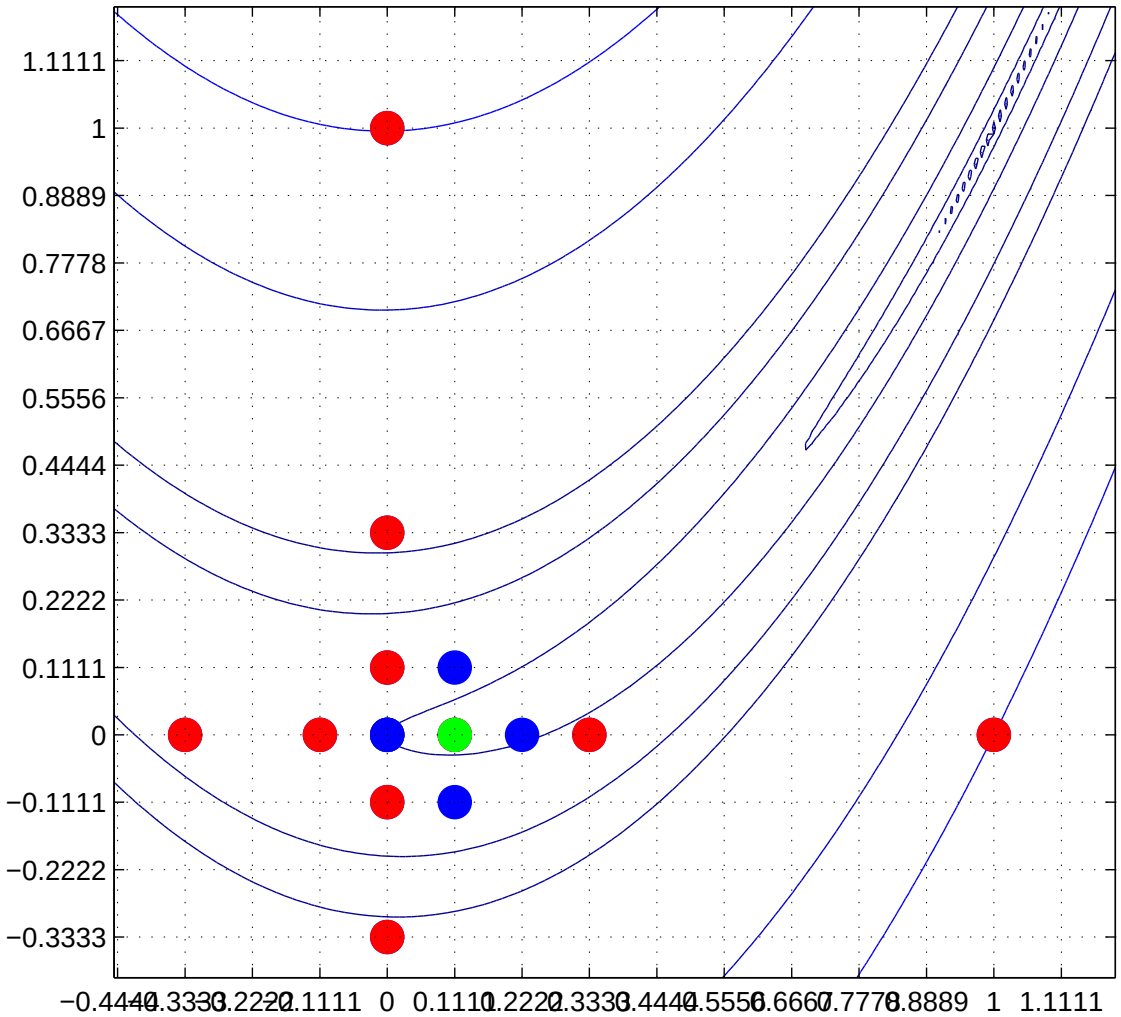
The compass-search on Rosenbrock's function



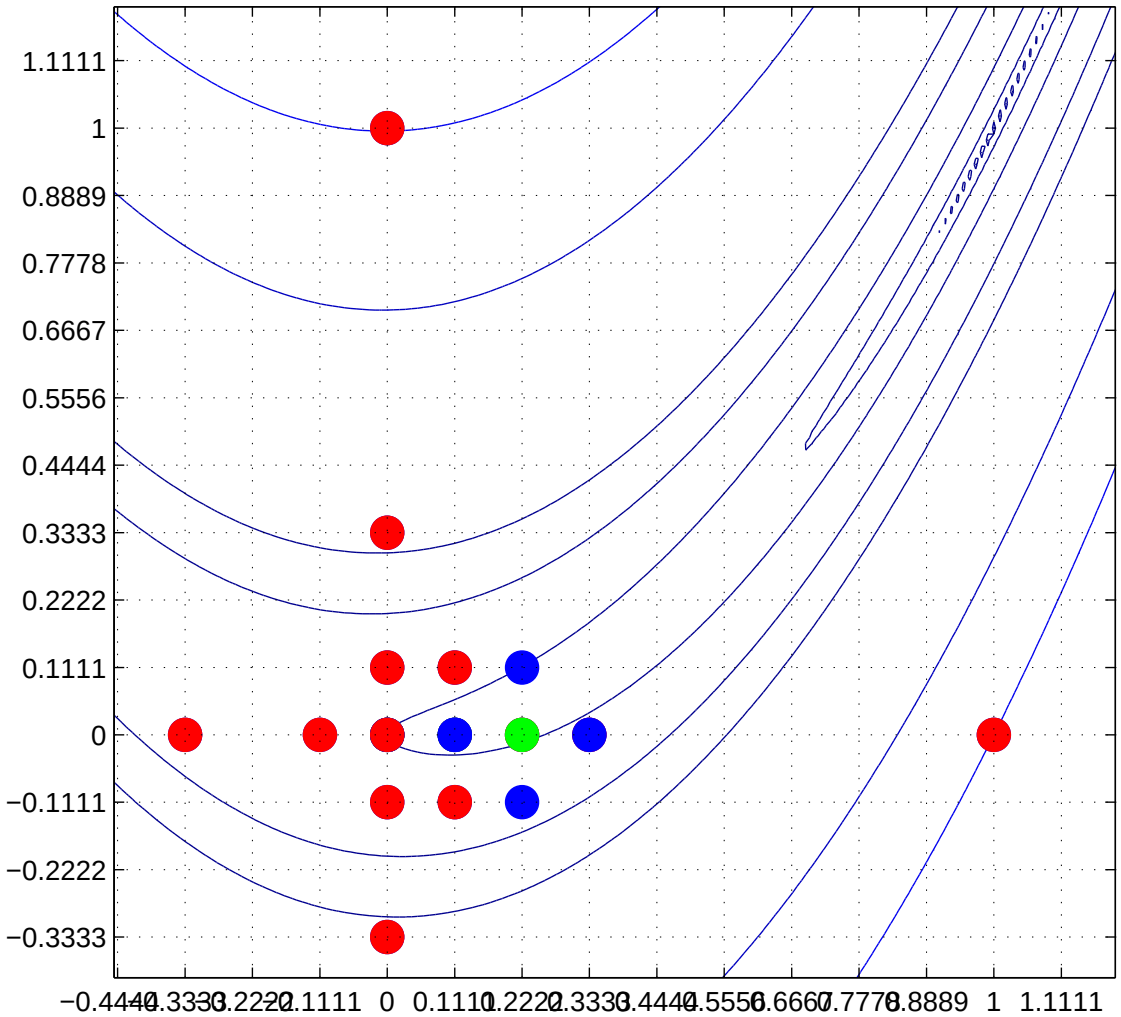
The compass-search on Rosenbrock's function



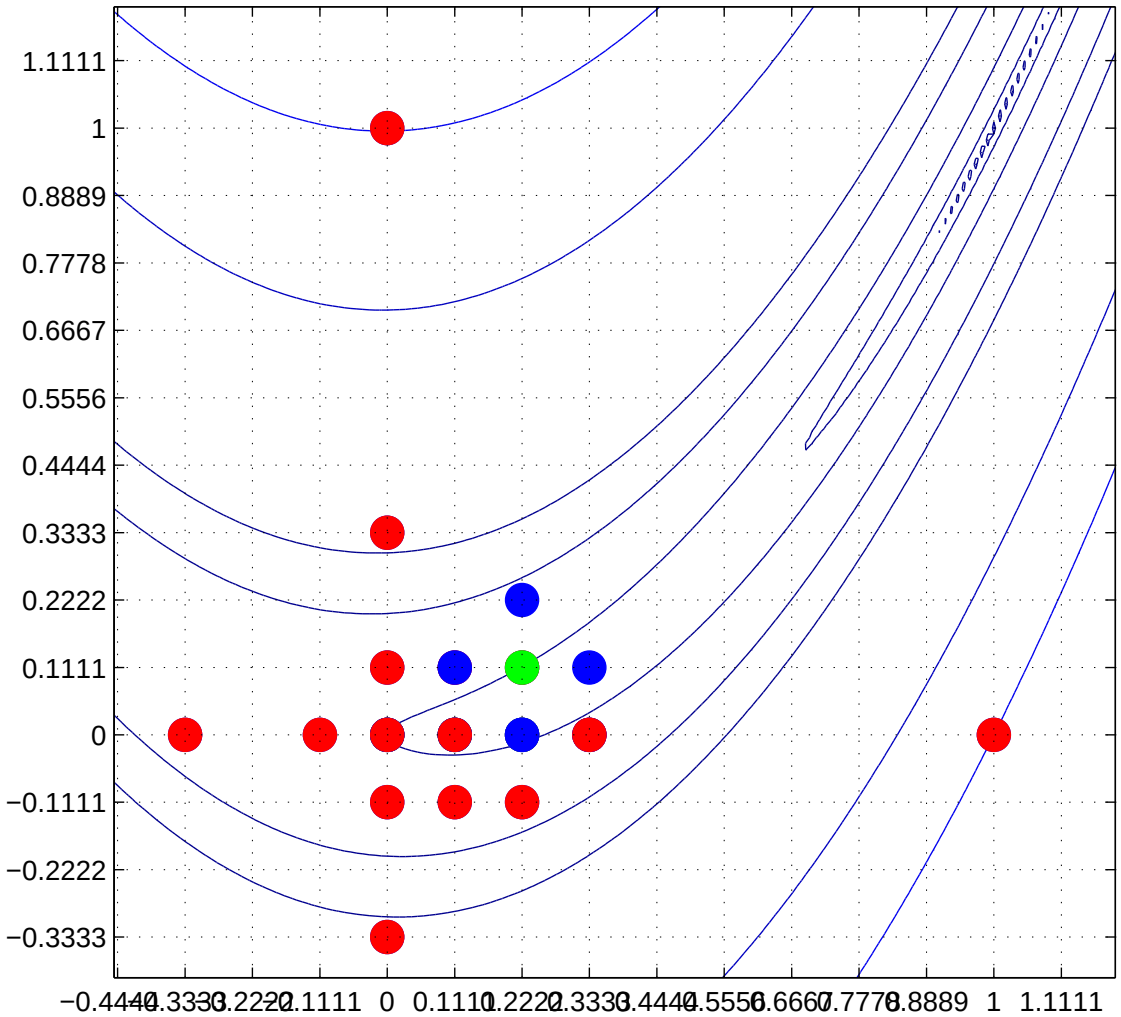
The compass-search on Rosenbrock's function



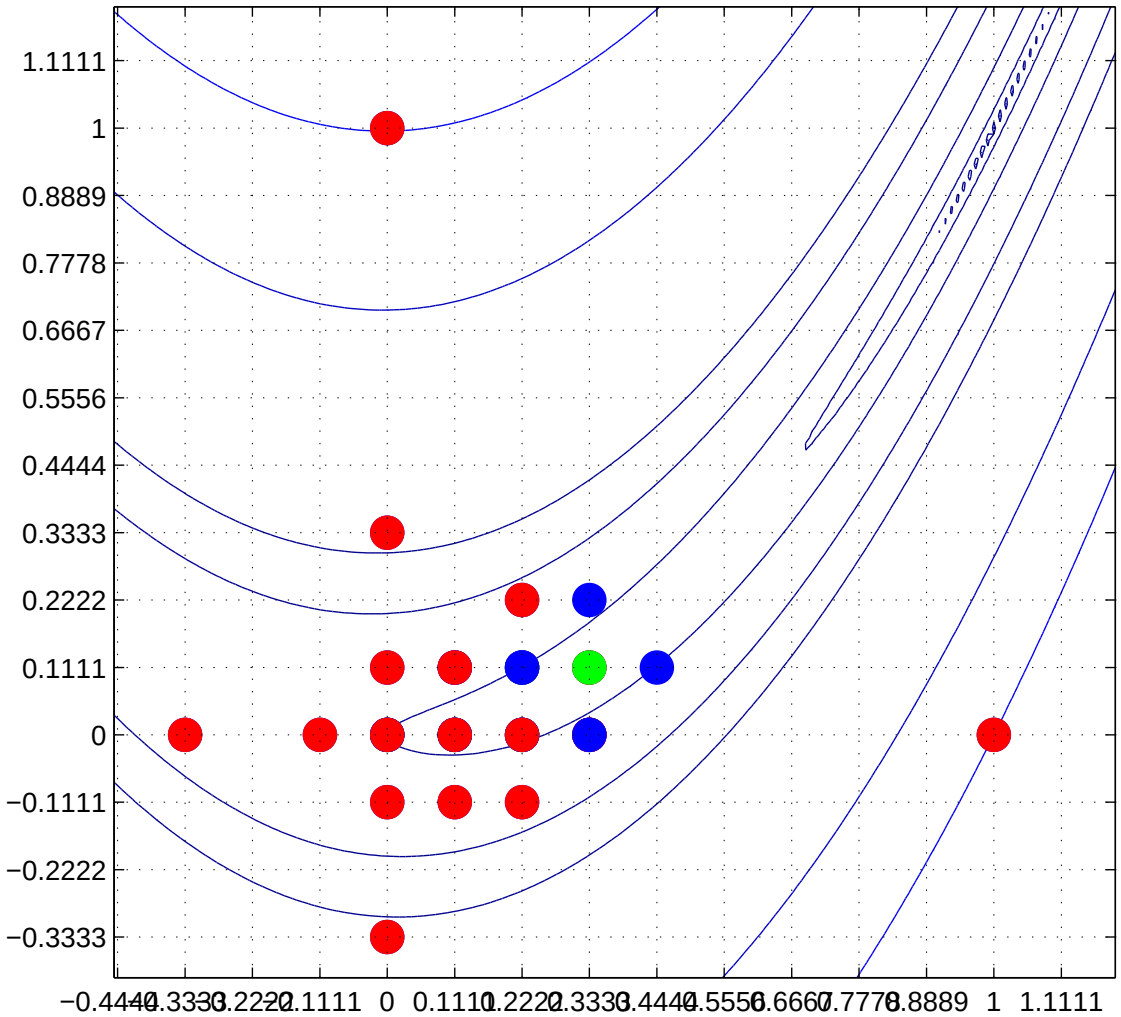
The compass-search on Rosenbrock's function



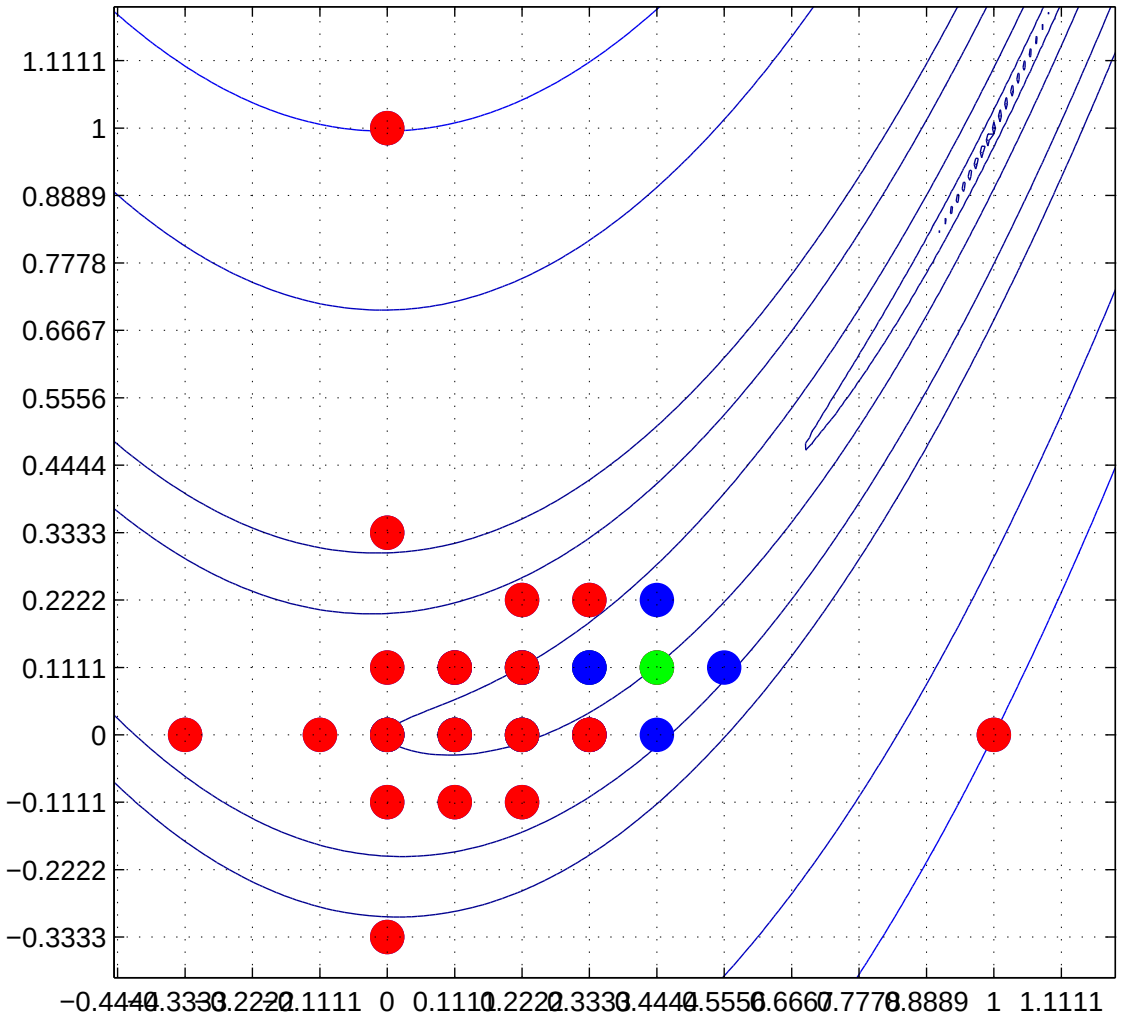
The compass-search on Rosenbrock's function



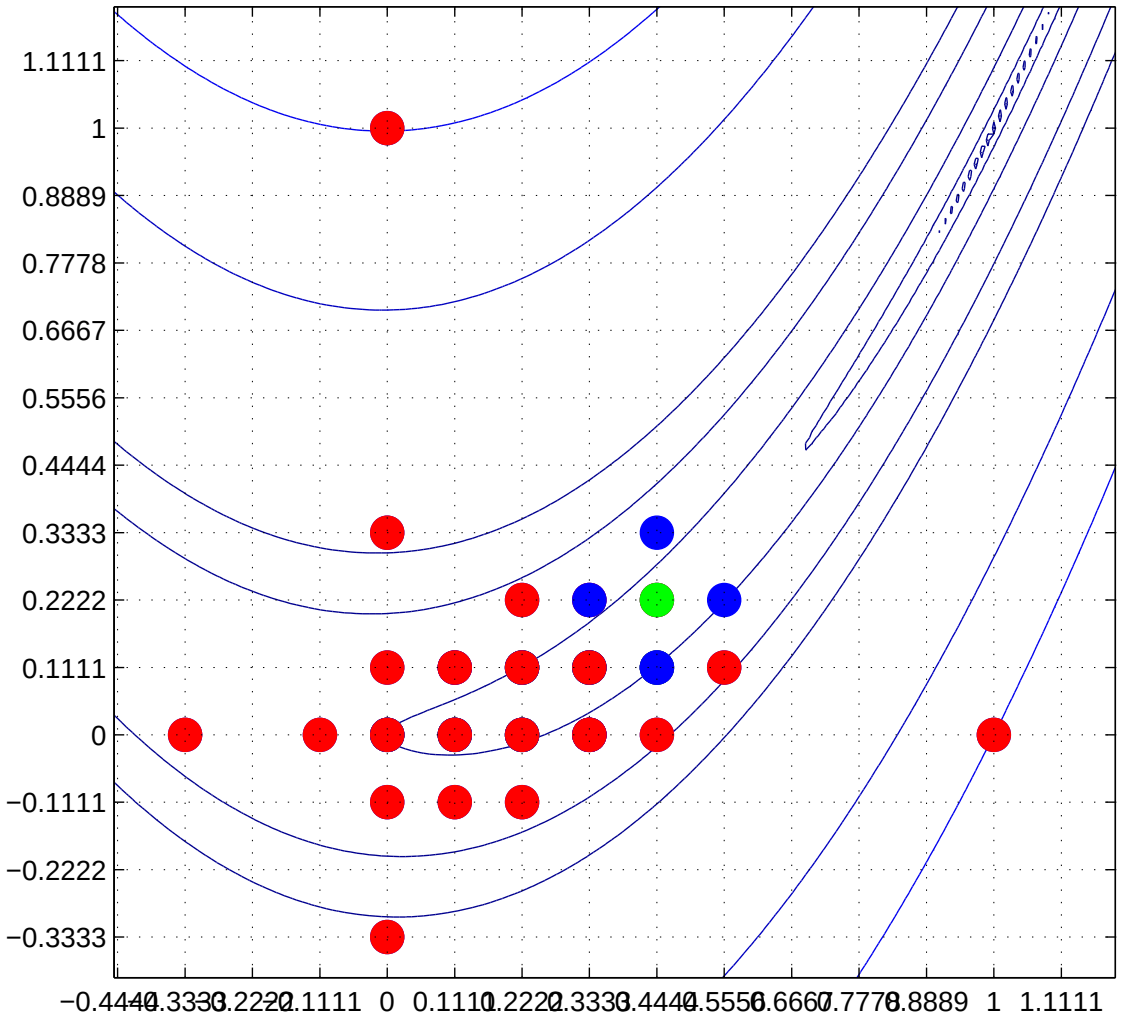
The compass-search on Rosenbrock's function



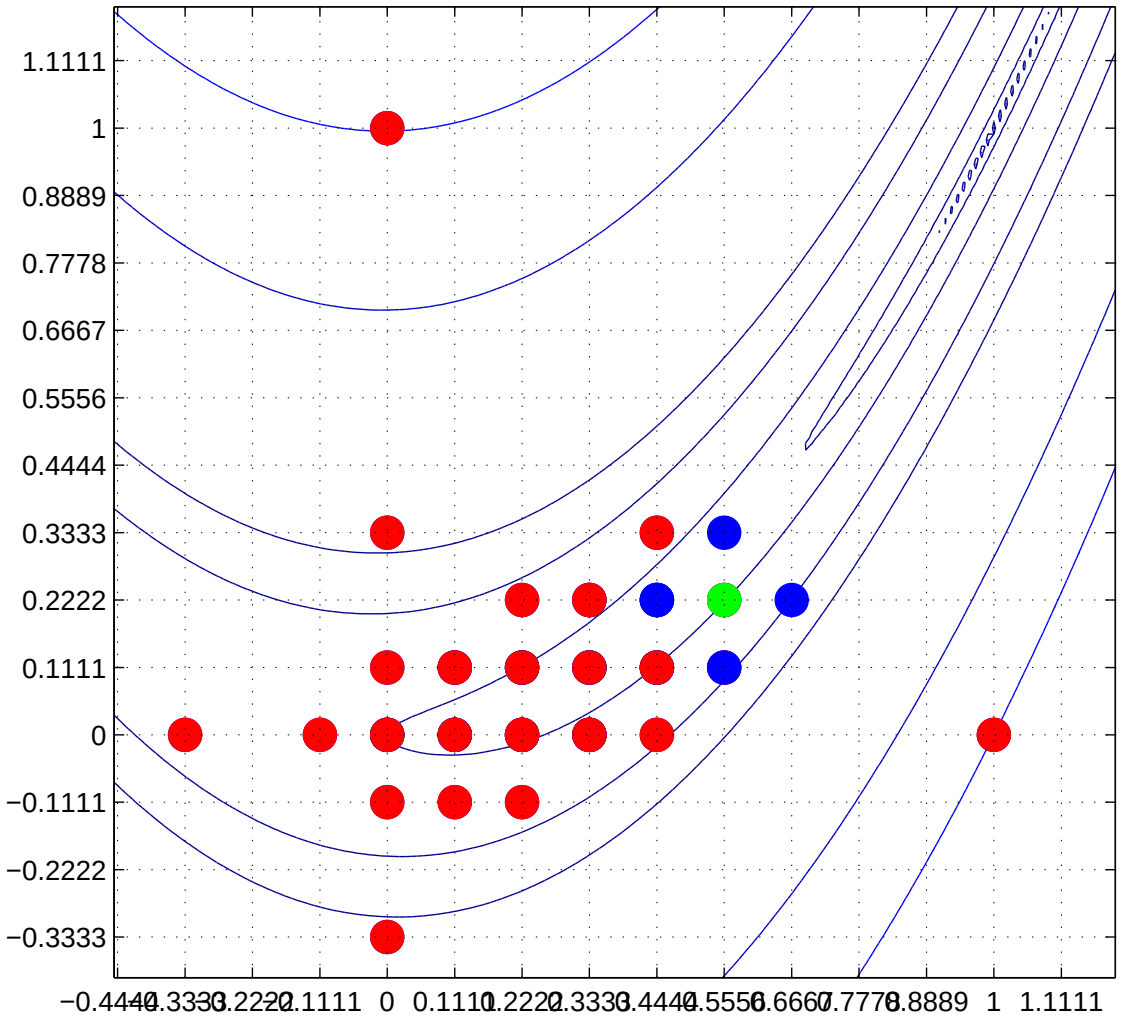
The compass-search on Rosenbrock's function



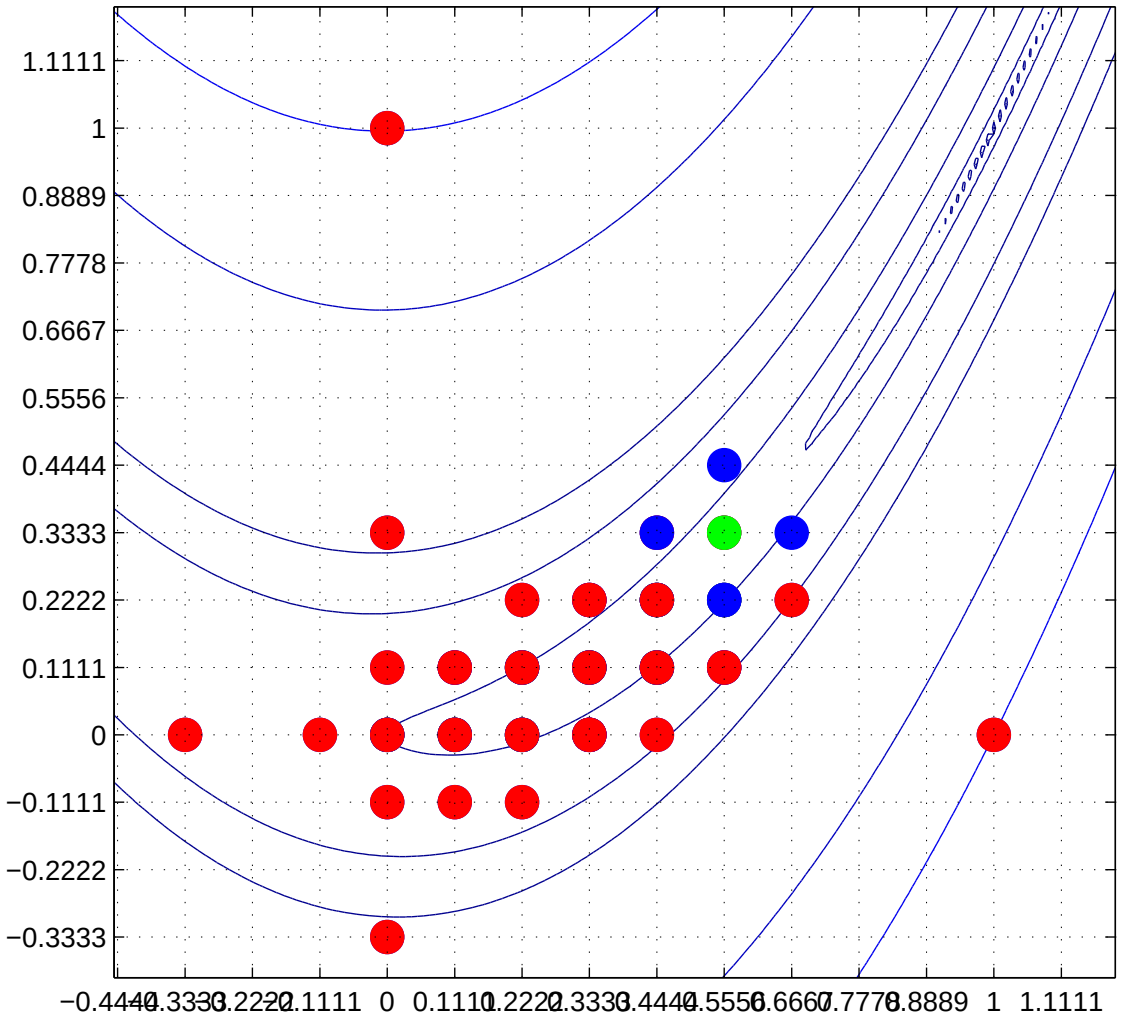
The compass-search on Rosenbrock's function



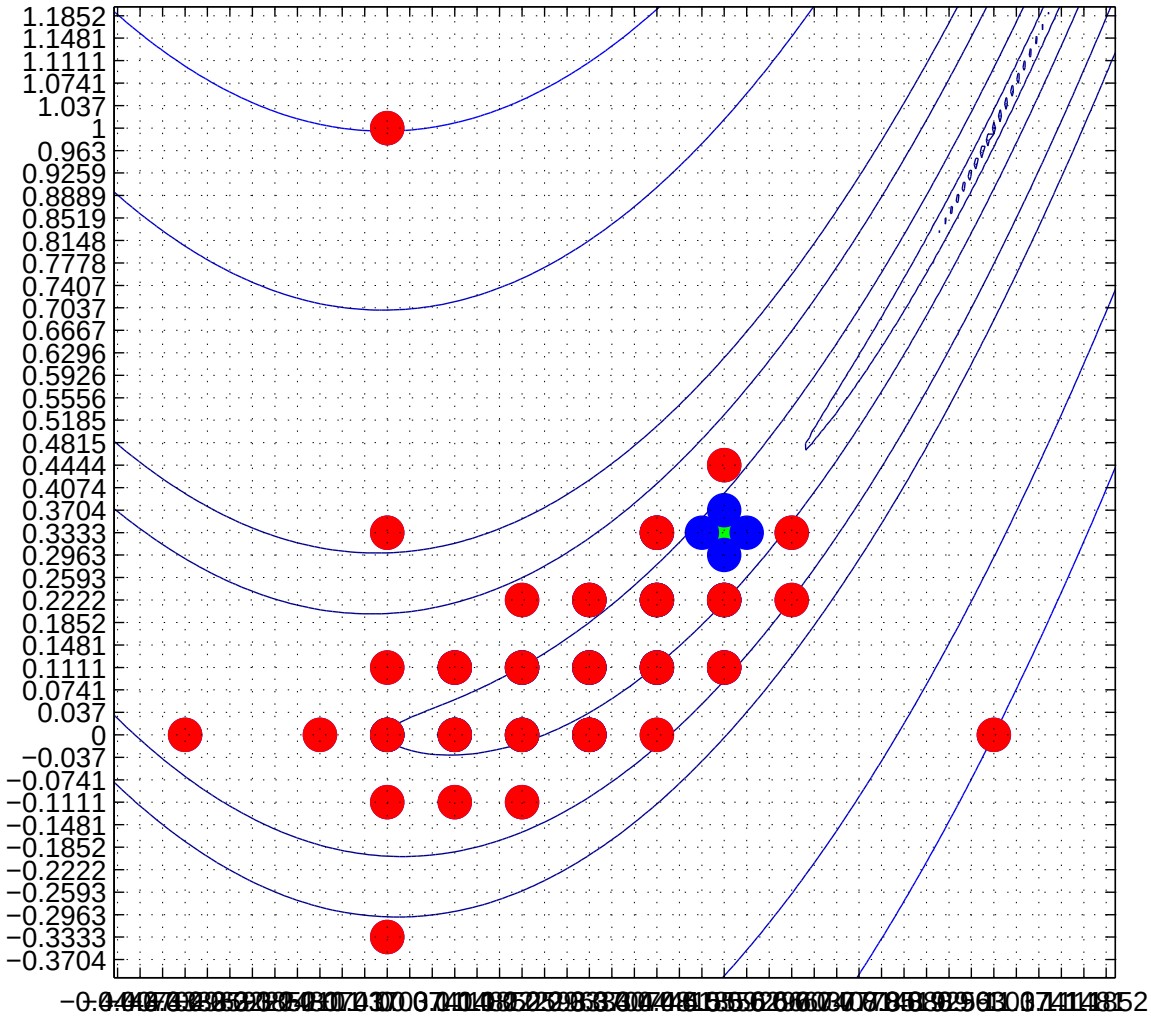
The compass-search on Rosenbrock's function



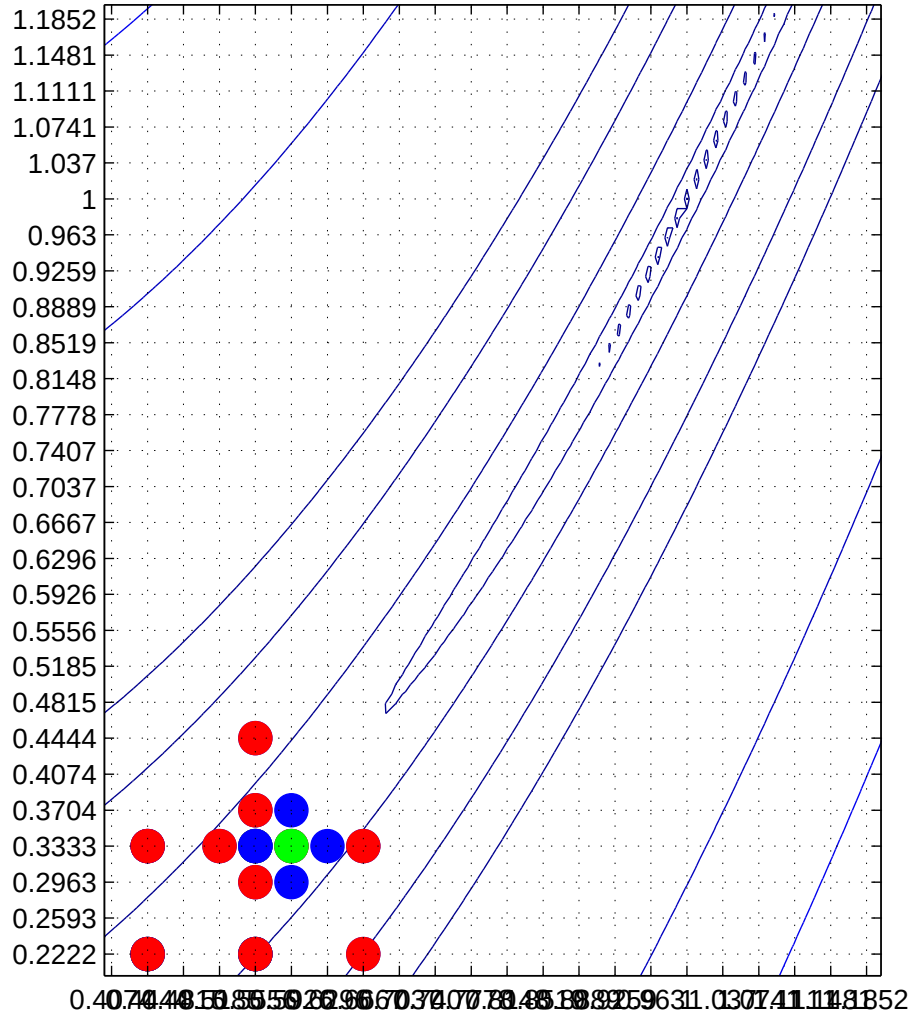
The compass-search on Rosenbrock's function



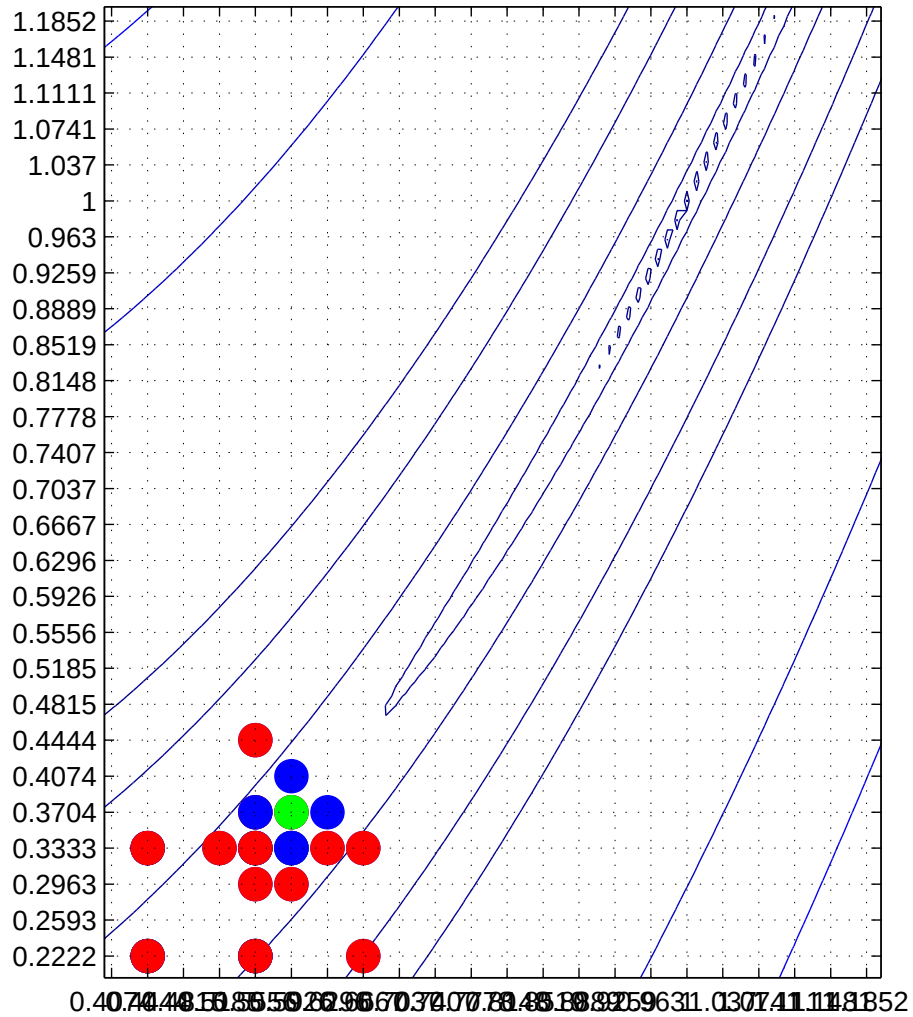
The compass-search on Rosenbrock's function



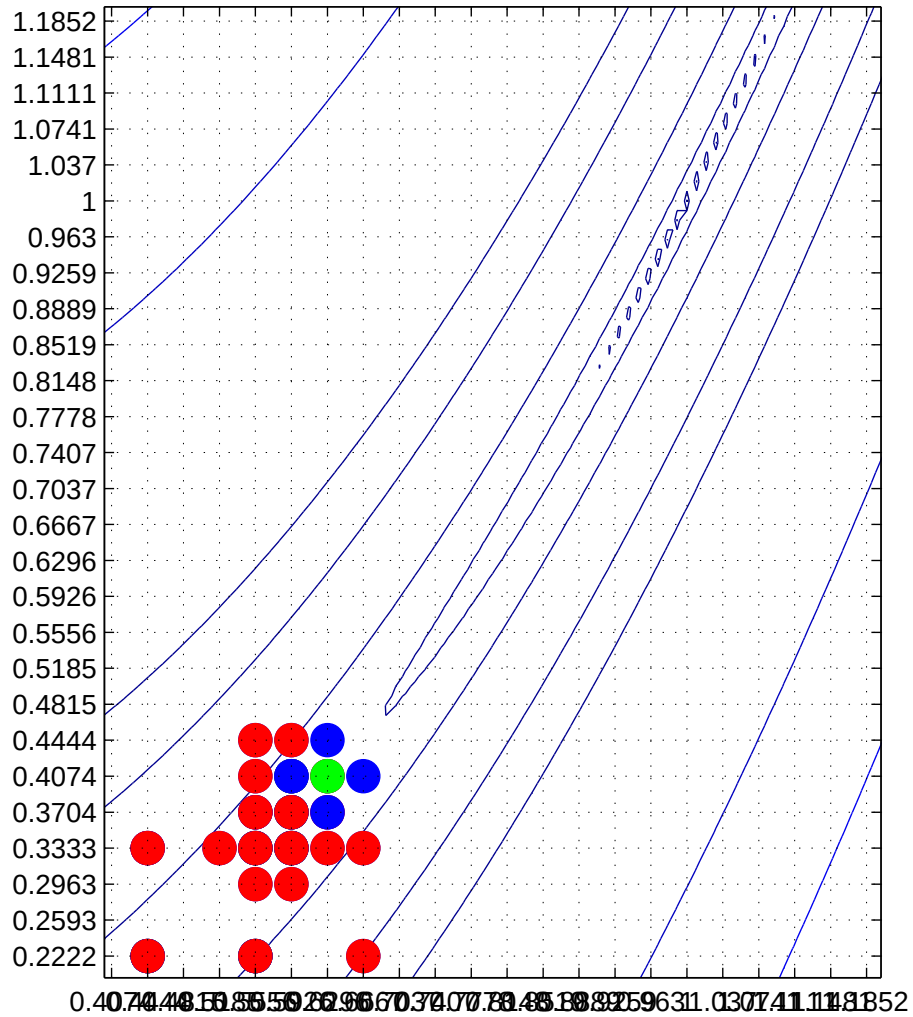
The compass-search on Rosenbrock's function



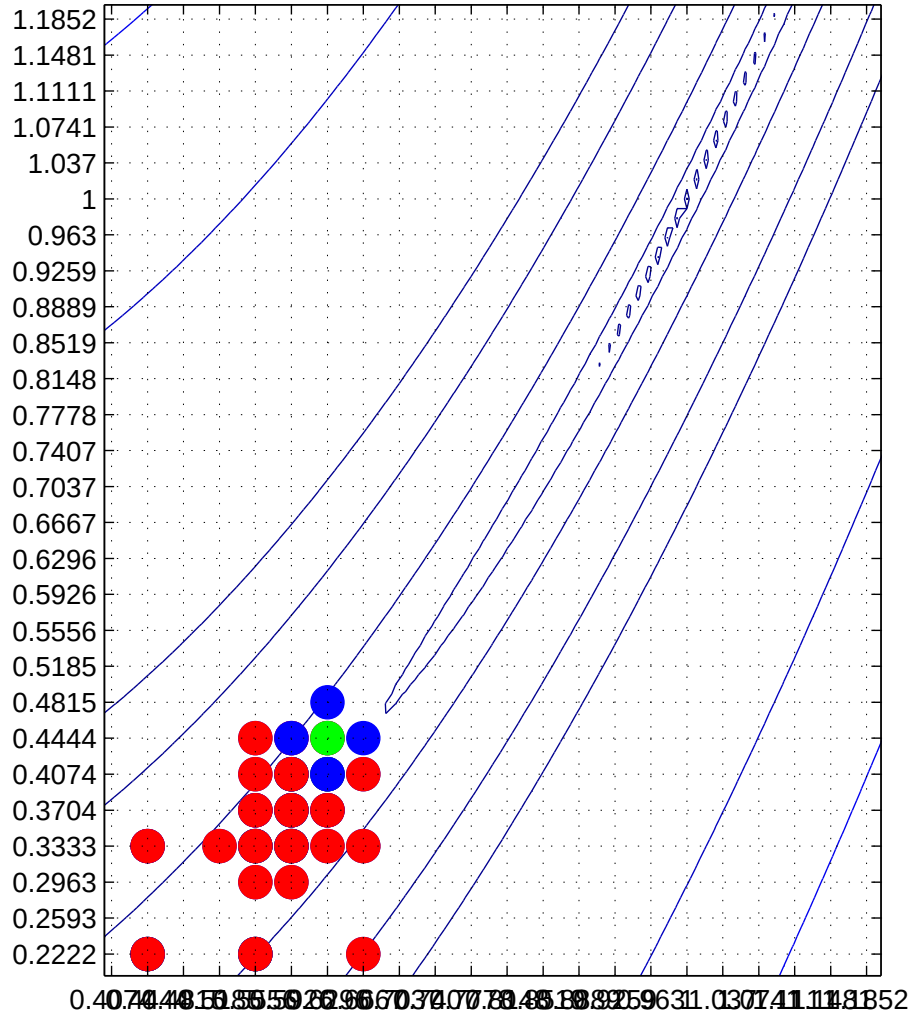
The compass-search on Rosenbrock's function



The compass-search on Rosenbrock's function



The compass-search on Rosenbrock's function



Weaknesses and algorithmic "solutions" (1)

- All compass points evaluated at each iteration
  - stop evaluating as soon as sufficient reduction is obtained
  - (suitable ordering of search directions)
- Painful zigzag crawling when axis not suitable
  - align axis on a grid with progress direction on previous grid (+ random complement)
  - define progress on a user-defined number of past iterations (inertia)
  - (independent scale for each variable)

Weaknesses and algorithmic "solutions" (2)

- Compute function at infeasible points
  - keep track of active and nearly-active bound constraints
- Slow progress on fine grids
  - expand the (continuous) grid on "successful iterations"
- Minima in neighbouring discrete subspaces not close to each other
  - recursively explore a local tree of discrete subspaces
  - keep track of record value in each such subspace to avoid re-exploration

Additional package features

Implementation features:

- check-pointing at user-specified frequency
- norms of vector functions t(f(x))
- allows objective functions with user-defined parameters
- allows partial objective computations
- allows randomized termination test
- provides a FD estimate of the (projected) gradient's norm (in the continuous variables) at termination
- very flexible keyword-based calling sequence

Algorithmic parameters

BFO: the Brute-Force Optimizer

User may specify (amongst others):

- grid shrinking/expansion factors
- inertia for defining progress directions
- initial scale in continuous variables
- local tree-search strategy (depth-first vs breadth-first)
- max conditioning for sampling gradient descent

A first setting for algorithm tuning

How best choose the algorithmic parameters?

- a set of 64 (uncs) + 64 (box) + 47 (mixed) test problems
- define objective function value

  $\phi_{BFO}(\text{params})$ = total number of evaluations to solve all problems

- choose (reasonable) initial default values
- simple idea: let BFO optimize for finding better values!
  (problem in 6 continuous and 2 discrete variables)

(D. Orban and Ch. Audet)

A robust approach (1)

DANGER: overfitting...

Investigate other formulations inspired by robust optimization (in continuous variables)

(Idea also considered by A. Conn?)

- local sample $\mathcal{S}(\text{params})$:
  Consider varying each continuous algorithmic parameter by

$$-5\% \quad -1\% \quad 0\% \quad +1\% \quad +5\%$$

  around each tested value (percentage tunable)
- local box

$$\mathcal{B}(\text{params}) = \prod_{1}^{\#\text{params}} [0.95\,\text{params}_i, 1.05\,\text{params}_i]$$

A robust approach (2)

Three "robust" formulations:

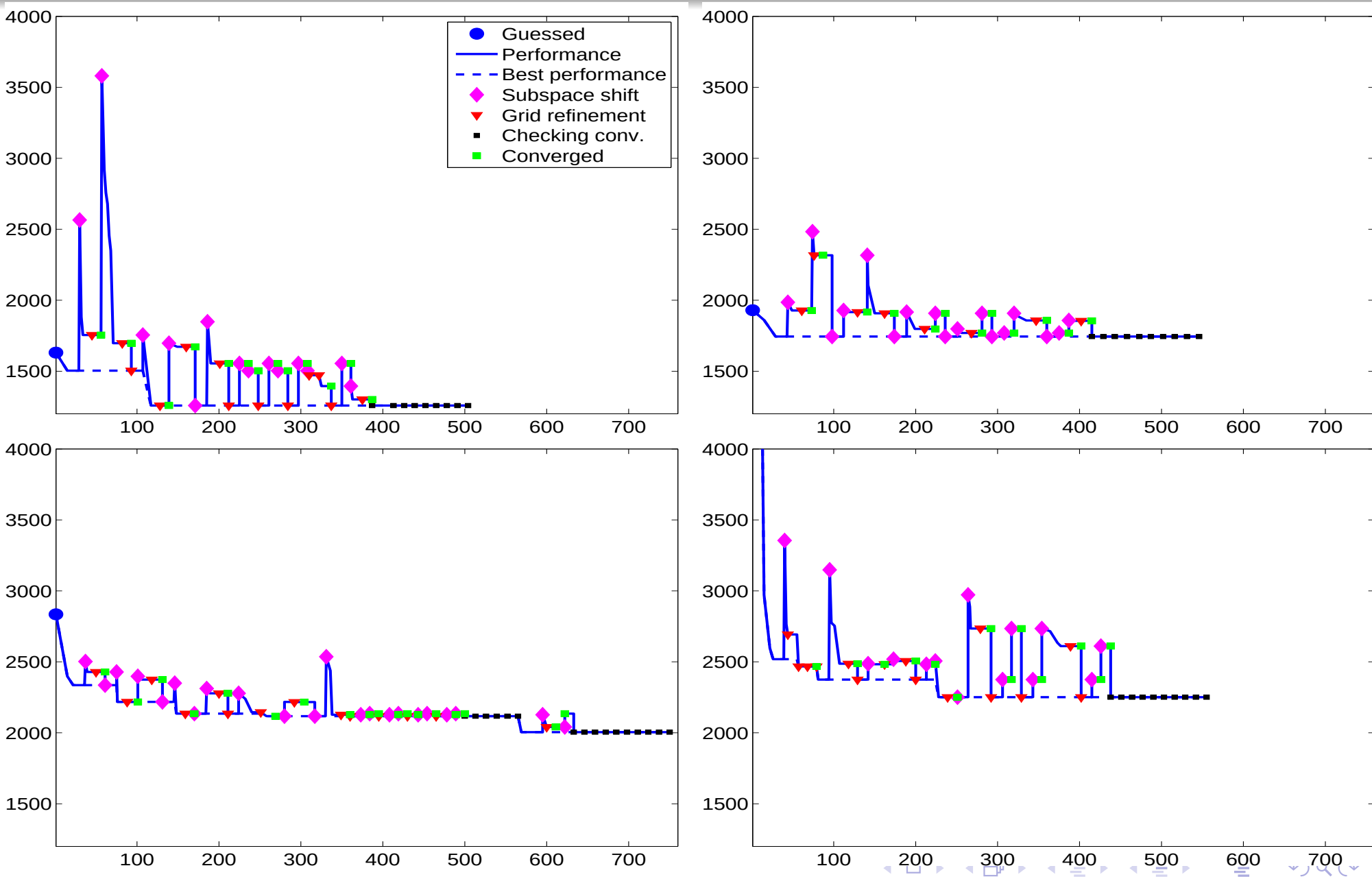
$$\min_{\text{params}} \sum_{\text{pars}\in\mathcal{S}(\text{params})} \phi_{BFO}(\text{pars})$$

$$\min_{\text{params}} \max_{\text{pars}\in\mathcal{S}(\text{params})} \phi_{BFO}(\text{pars})$$

$$\min_{\text{params}} \max_{\text{pars}\in\mathcal{B}(\text{params})} \phi_{BFO}(\text{pars})$$

- progressively (and considerably) more expansive
- BFO may be used for both optimization levels (form. 2 & 3)
- maximization problem possibly approximate (moderately coarse stopping rule)

On 3 test problems only: the optimization history



On 3 test problems only: the optimal parameters

	shrink	decr.	max exp.	in.	scale	exp.	condlim
guess	0.1	0.00001	3	5	1	2.1	1.2
simple	0.05	0.56469	3	6	1	2.1	1.147526
sum	0.1	0.00001	3.200578	5	1.25	2.114963	1.0734592
max ($\mathcal{S}$)	0.05	0.00001	4.0626793	9	1.1787	1.99772	1.4533078
max ($\mathcal{B}$)	0.1	0.000001	3.258	6	1.25	2.1	1.2

Which is the most sensible?

Maybe max ($\mathcal{B}$) ?

More computations are on the way...

Conclusions

BFO

- yet another direct method for mixed integer local optimization
- further direct improvements under development (ordering, sample gradients, etc)
- simple compact implementation
- freely-available easy-to-use Matlab package
- useable for algorithm tuning
- more to do (constraints, use of structure, ...)

Optimization of algorithmic parameters

- four formulations proposed and (nearly) tested
- fully robust formulation most reasonable?

Thank you for your attention!