Numerical Experience with a Class of Trust-Region Algorithms for Unconstrained Smooth Optimization

XI Brazilian Workshop on Continuous Optimization

Abel Soares Siqueira
Universidade Federal do Paraná

Geovani Nunes Grapiglia
Universidade Federal do Paraná

May 23, 2016

Unconstrained Optimization

$$\min f(x),$$

$f : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable.

Classical Trust-Region Method (Powell [1])

1. $q_k(d) = f(x^k) + \nabla f(x^k)^T d + \frac{1}{2} d^T B_k d$
2. d^k such that $\|d^k\| \leq \delta_k$ and
$$q_k(0) - q_k(d^k) \geq \kappa \|\nabla f(x^k)\| \min \left\{ \frac{\|\nabla f(x^k)\|}{1 + \|B_k\|}, \delta_k \right\}.$$
3. $\rho_k = \dfrac{f(x^k) - f(x^k + d^k)}{q_k(0) - q_k(d^k)}$
4. If $\rho_k \geq \eta_1$, do $x^{k+1} = x^k + d^k$. Otherwise $x^{k+1} = x^k$.
5. Choose δ_{k+1}

Modified Trust-Region Method (Fan and Yuan [2])

1. $q_k(d) = f(x^k) + \nabla f(x^k)^T d + \frac{1}{2} d^T B_k d$
2. d^k such that $\|d^k\| \leq \delta_k \|\nabla f(x^k)\|$ and
$$q_k(0) - q_k(d^k) \geq \kappa \|\nabla f(x^k)\| \min\left\{\frac{\|\nabla f(x^k)\|}{1 + \|B_k\|}, \delta_k \|\nabla f(x^k)\|\right\}.$$
3. $\rho_k = \dfrac{f(x^k) - f(x^k + d^k)}{q_k(0) - q_k(d^k)}$
4. If $\rho_k \geq \eta_1$, do $x^{k+1} = x^k + d^k$. Otherwise $x^{k+1} = x^k$.
5. Choose δ_{k+1}

ARC Method (Cartis, Gould, and Toint [3], [4])

1. $q_k(d) = f(x^k) + \nabla f(x^k)^T d + \frac{1}{2} d^T B_k d + \dfrac{1}{3\delta_k}\|d\|^3$
2. d^k such that $\|d^k\| \leq \delta_k^{\frac{1}{2}} \|\nabla f(x^k)\|^{\frac{1}{2}}$ and
$$q_k(0) - q_k(d^k) \geq \kappa \|\nabla f(x^k)\| \min\left\{ \frac{\|\nabla f(x^k)\|}{1 + \|B_k\|}, \delta_k^{\frac{1}{2}} \|\nabla f(x^k)\|^{\frac{1}{2}} \right\}.$$
3. $\rho_k = \dfrac{f(x^k) - f(x^k + d^k)}{q_k(0) - q_k(d^k)}$
4. If $\rho_k \geq \eta_1$, do $x^{k+1} = x^k + d^k$. Otherwise $x^{k+1} = x^k$.
5. Choose δ_{k+1}

NSC Method

- "Nonlinear stepsize control, trust regions and regularizations for unconstrained optimization", Toint (2013) [5]
- Generalizes trust-region and regularization methods;
- Provides unified convergence theory;
- Suggests new methods.

Let $\phi, \psi, \chi : \mathbb{R}^n \to \mathbb{R}$ be nonnegative functions such that

$$\min\{\phi(x), \psi(x), \chi(x)\} = 0 \Rightarrow x \text{ is a critical point}$$

NSC Method

1. Find a model $q_k(d)$ such that $q_k(0) = f(x^k)$ and $f(x^k + d) - q_k(d) \leq \kappa_m \|d\|^2$
2. d^k such that $\|d^k\| \leq \Delta(\delta_k, \chi_k) = \delta_k^\alpha \chi_k^\beta$ and
$$q_k(0) - q_k(d^k) \geq \kappa\psi_k \min\left\{\frac{\phi_k}{1 + \|B_k\|}, \Delta(\delta_k, \chi_k)\right\}.$$
3. $\rho_k = \dfrac{f(x^k) - f(x^k + d^k)}{q_k(0) - q_k(d^k)}$
4. If $\rho_k \geq \eta_1$, do $x^{k+1} = x^k + d^k$. Otherwise $x^{k+1} = x^k$.
5. $$\delta_{k+1} \in \begin{cases} [\gamma_1\delta_k, \gamma_2\delta_k] & \rho_k < \eta_1 \\ [\gamma_2\delta_k, \delta_k] & \eta_1 \leq \rho_k < \eta_2 \\ [\delta_k, +\infty) & \rho_k \geq \eta_2 \end{cases}$$

NSC Method (Particular cases)

- Classical Trust-Region Method

$$\begin{cases} \alpha = 1 \text{ and } \beta = 0 \\ \phi_k = \psi_k = \chi_k = \|\nabla f(x^k)\| \end{cases} \Longrightarrow \Delta(\delta_k, \chi_k) = \delta_k$$

- Modified Trust-Region Method

$$\begin{cases} \alpha = \beta = 1 \\ \phi_k = \psi_k = \chi_k = \|\nabla f(x^k)\| \end{cases} \Longrightarrow \Delta(\delta_k, \chi_k) = \delta_k \|\nabla f(x^k)\|$$

- ARC Method

$$\begin{cases} \alpha = \beta = 1/2 \\ \phi_k = \psi_k = \chi_k = \|\nabla f(x^k)\| \end{cases} \Longrightarrow \Delta(\delta_k, \chi_k) = \delta_k^{\frac{1}{2}} \|\nabla f(x^k)\|^{\frac{1}{2}}$$

How α and β affect the method?

Theorem

Suppose that

- $\phi_k \geq \chi_k$ *and* $\psi_k \geq \chi_k$*;*
- $\{f(x_k)\}$ *is bounded below; and*
- $\|B_k\| \leq \kappa$.

Then the NSC method takes at most $\mathcal{O}(\epsilon^{-2})$ *iterations to achieve* $\chi_k \leq \epsilon$.

Nonlinear stepsize control algorithms: complexity bounds for first and second order optimality **(TR) - Grapiglia, Yuan, and Yuan ([6])**

How does α and β affect the method

- Similar to what Gould, Orban, Sartenaer, *et al.* [7] did;
- Discretize $(0, 1] \times [0, 1]$ to a 50×51 grid;
- Define algorithm for each (α, β);
- Run algorithm for 58 CUTEst problems (small);
- Analyze how sensitive it is;
- Anything better than traditional.

Implementation

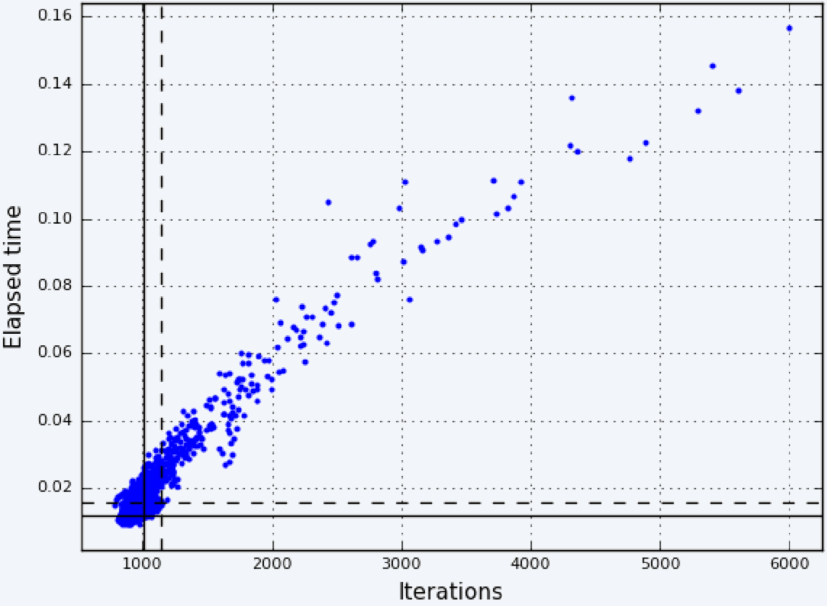
- $q(d) = f(x^k) + \nabla f(x^k)^T d + \frac{1}{2} d^T \nabla^2 f(x^k) d$
- Find d^k by Steihaug-Toint
- $\epsilon = 10^{-8}$, maximum number of iteration 1000
- $\eta_1 = \frac{1}{4}$, $\eta_2 = \frac{3}{4}$
- $\sigma_1 = \frac{1}{6}$, $\sigma_2 = 4$
- $\delta_{k+1} = \begin{cases} \sigma_1 \delta_k & \rho_k < \eta_1 \\ \delta_k & \eta_1 \leq \rho_k < \eta_2 \\ \sigma_2 \delta_k & \rho_k \geq \eta_2 \end{cases}$

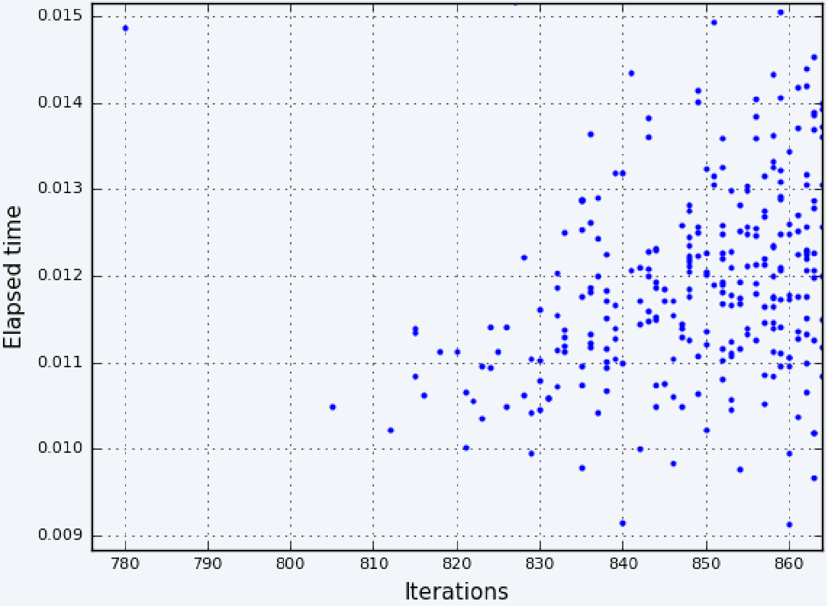
Measurement of time

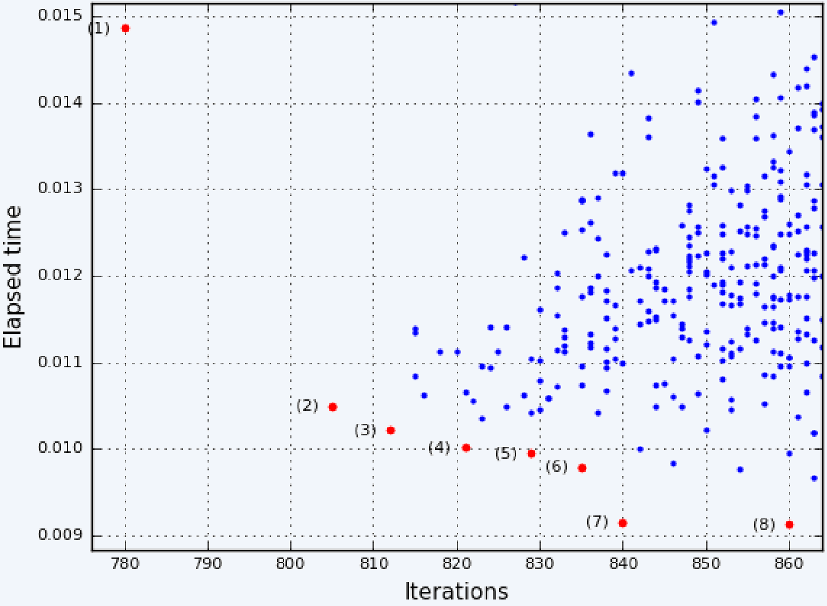
- Some problems have very small elapsed time (smallest: 1.32×10^{-6});
- Run problem many times (how many?);
- First time: t_0, define $N = \left\lceil \dfrac{0.1}{t_0} \right\rceil$;
- Run N times, get average;
- Get 3 averages, keep the best.

Elapsed time and number of iterations

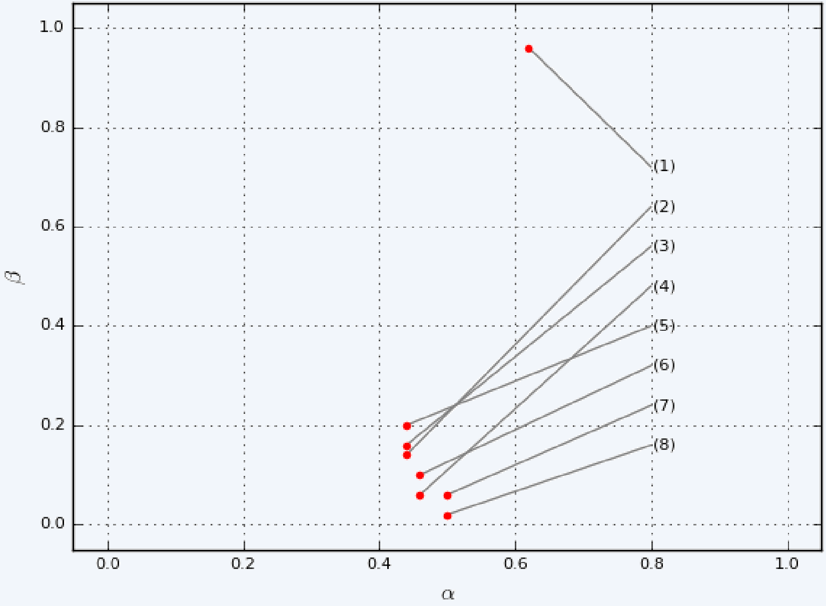
- 30 problems converged for every choice of (α, β);
- Number of iterations and elapsed time for these problems;

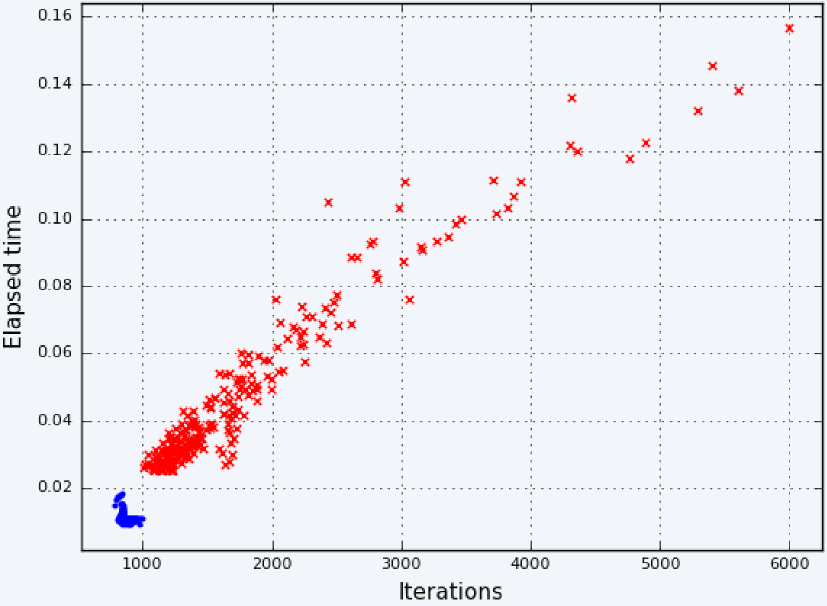

Elapsed time
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
1000
2000
3000
4000
5000
6000
Iterations

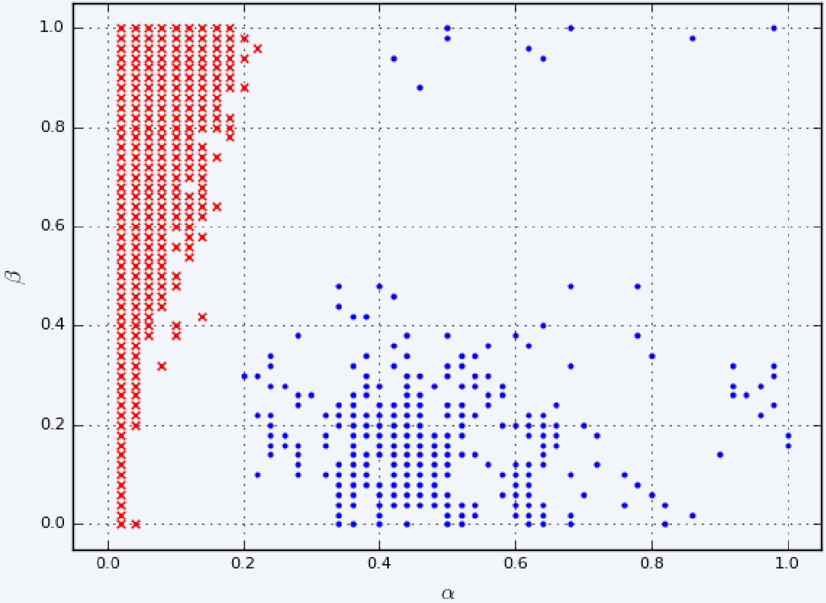

0.015
0.014
0.013
0.012
0.011
0.010
0.009
Elapsed time
780
790
800
810
820
830
840
850
860
Iterations



Elapsed time
Iterations
0.015
0.014
0.013
0.012
0.011
0.010
0.009
780
790
800
810
820
830
840
850
860
(1)
(2)
(3)
(4)
(5)
(6)
(7)
(8)

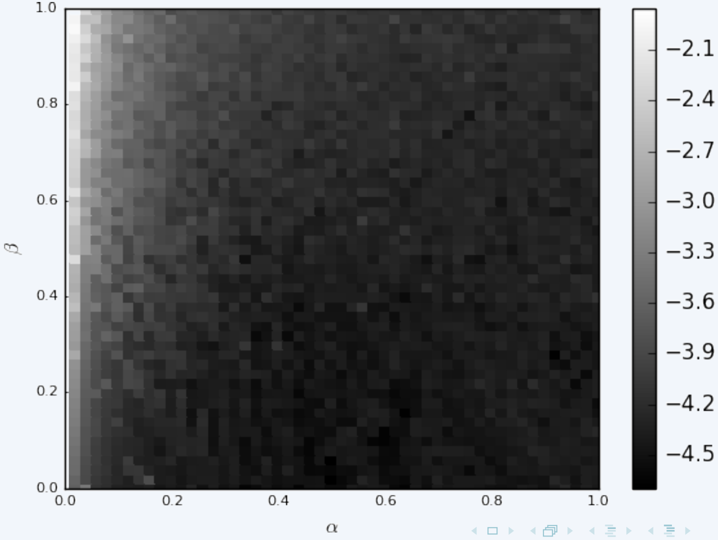

1.0
0.8
0.6
0.4
0.2
0.0
β
0.0
0.2
0.4
0.6
0.8
1.0
α
(1)
(2)
(3)
(4)
(5)
(6)
(7)
(8)


0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
Elapsed time
1000
2000
3000
4000
5000
6000
Iterations

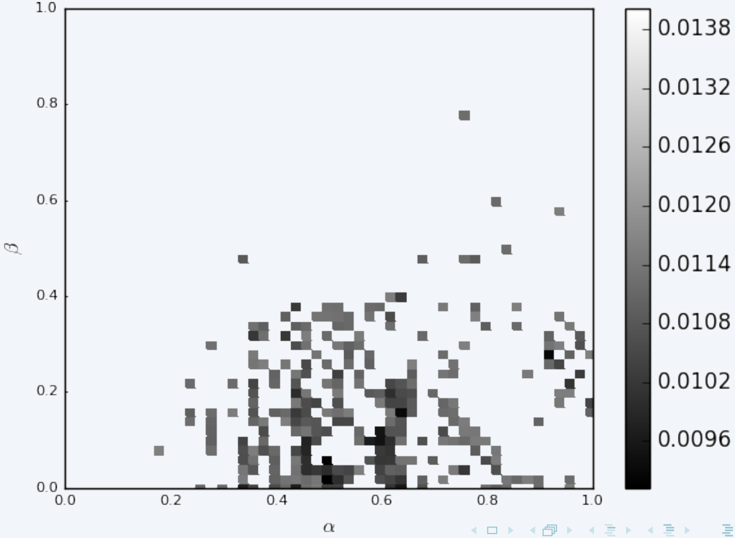

1.0
0.8
0.6
0.4
0.2
0.0
β
0.0
0.2
0.4
0.6
0.8
1.0
α

Time and iterations individually

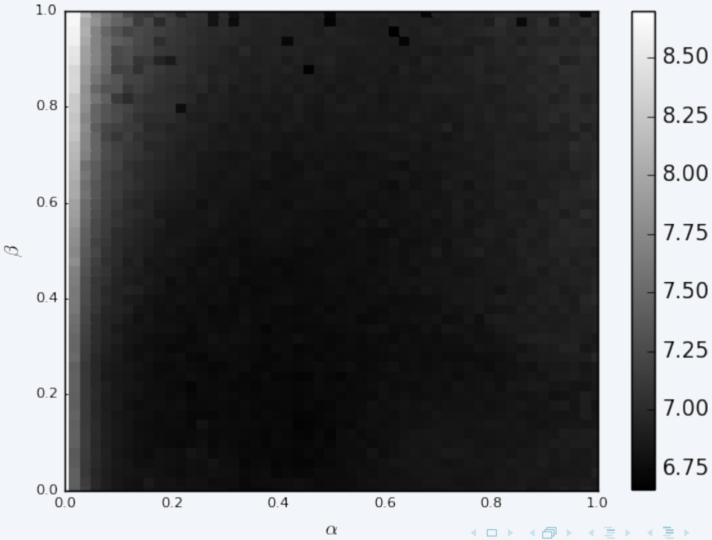
time



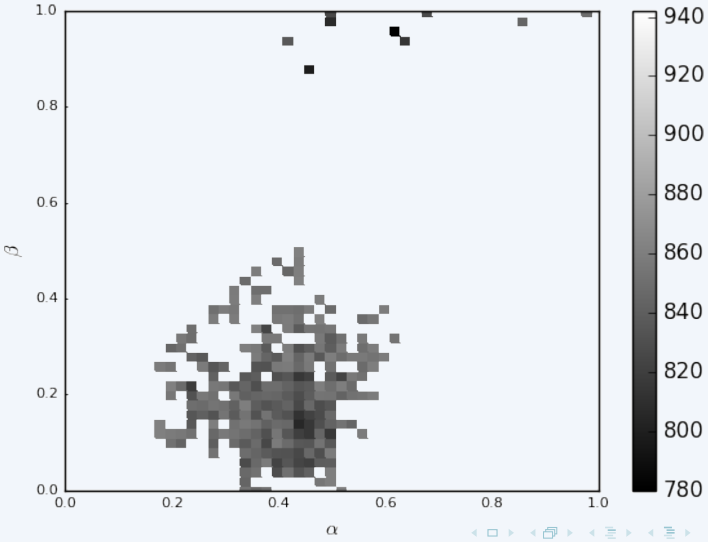
time


1.0
0.8
0.6
0.4
0.2
0.0
β
0.0
0.2
0.4
0.6
0.8
1.0
α
0.0138
0.0132
0.0126
0.0120
0.0114
0.0108
0.0102
0.0096

iter

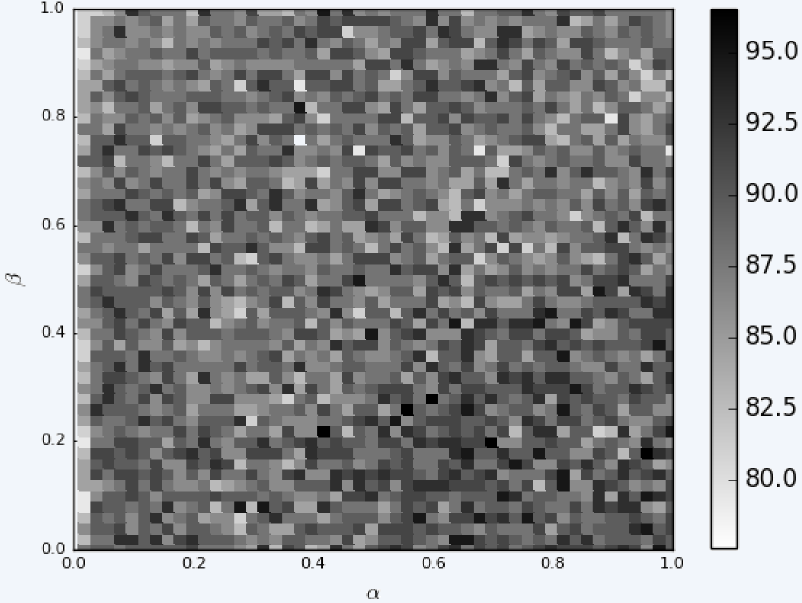

1.0
0.8
0.6
0.4
0.2
0.0
β
0.0
0.2
0.4
0.6
0.8
1.0
α
8.50
8.25
8.00
7.75
7.50
7.25
7.00
6.75

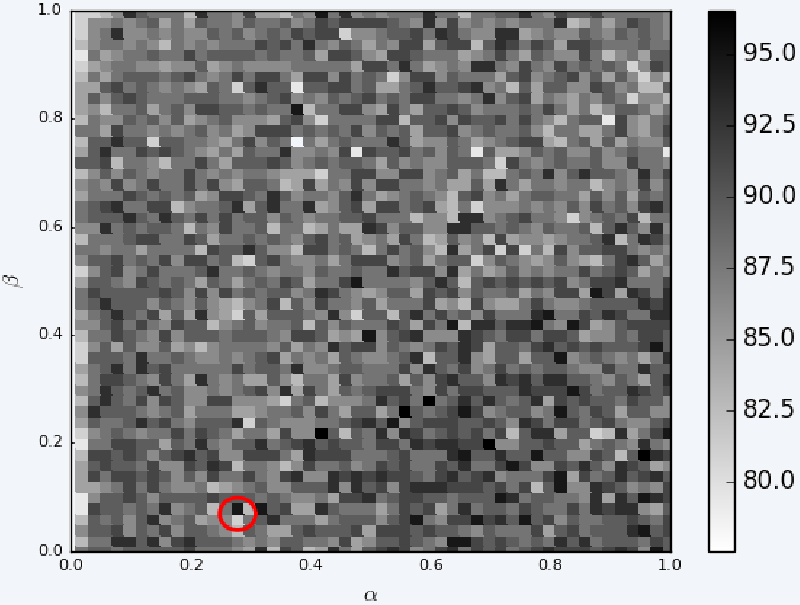
iter


1.0
0.8
0.6
0.4
0.2
0.0
β
0.0
0.2
0.4
0.6
0.8
1.0
α
940
920
900
880
860
840
820
800
780

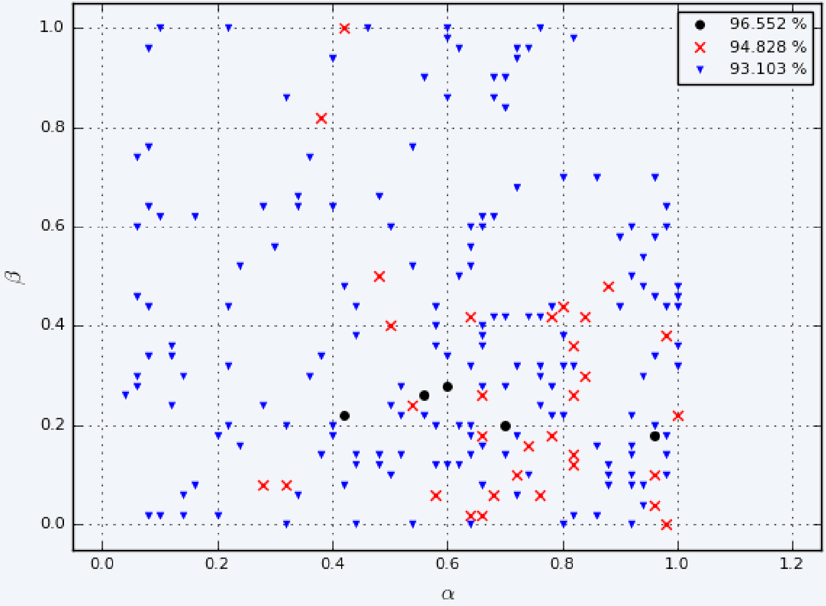
Robustness and efficiency

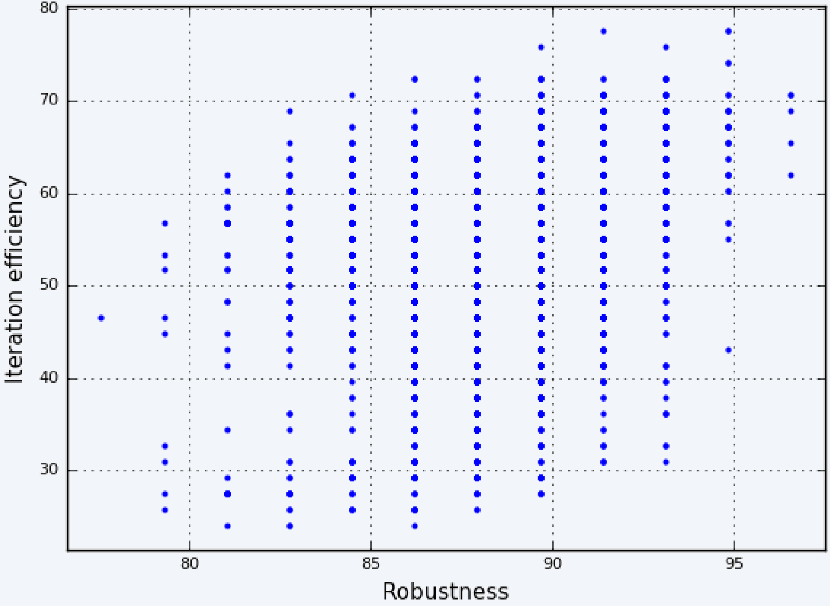
- Robustness varies between 45 and 56 problems;
- Performance profile
  - $r_{s,p} = \dfrac{c_{s,p}}{\min\{c_{s,p} \mid s \in \mathcal{S}\}} \quad s \in \mathcal{S}, p \in \mathcal{P}$;
  - $\rho_s(t) = \dfrac{\#\{r_{s,p} \leq t \mid p \in \mathcal{P}\}}{\#\mathcal{P}}$;
  - $\rho_s(1)$ is an efficiency measure;
  - $\rho_s(+\infty)$ is the robustness (independent of $\mathcal{S}$);
- Run $(1, 0)$ vs $s = (\alpha, \beta)$, get $\rho_s(1)$ and $\rho_s(+\infty)$.

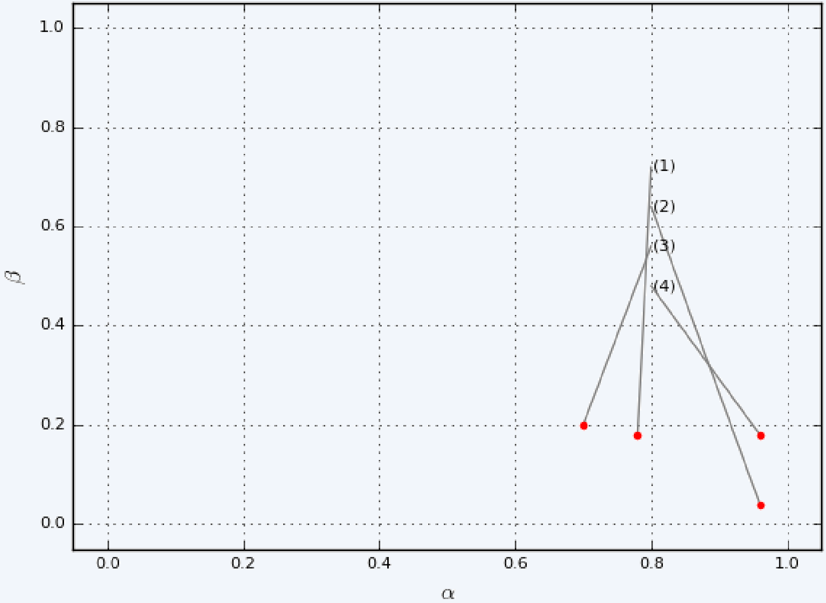

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
β
α
95.0
92.5
90.0
87.5
85.0
82.5
80.0



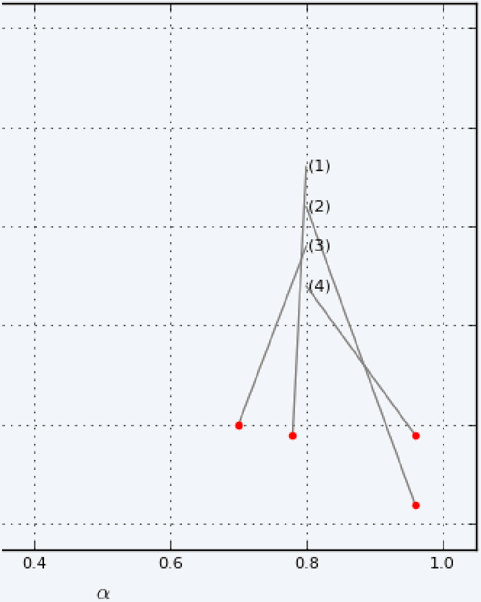
1.0
0.8
0.6
0.4
0.2
0.0
β
0.0
0.2
0.4
0.6
0.8
1.0
α
95.0
92.5
90.0
87.5
85.0
82.5
80.0


96.552 %
94.828 %
93.103 %
β
α
0.0
0.2
0.4
0.6
0.8
1.0
1.2


80
70
60
50
40
30
Iteration efficiency
80
85
90
95
Robustness



1.0
0.8
0.6
0.4
0.2
0.0
β
(1)
(2)
(3)
(4)
0.0
0.2
0.4
0.6
0.8
1.0
α

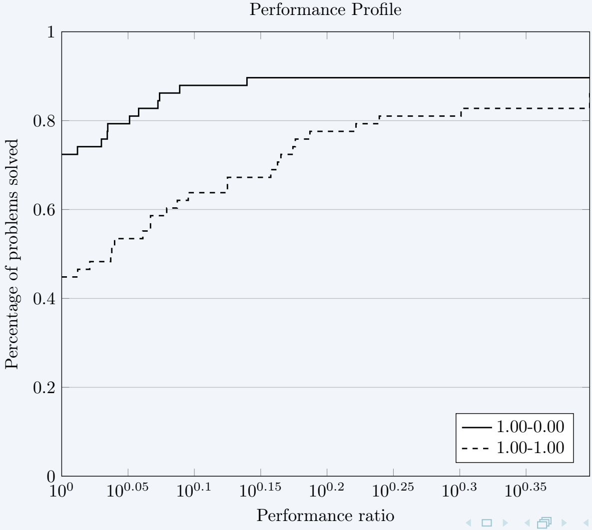


	Rob	Eff
(1), (2)	94.8%	77.6%
(3), (4)	96.6%	70.7%

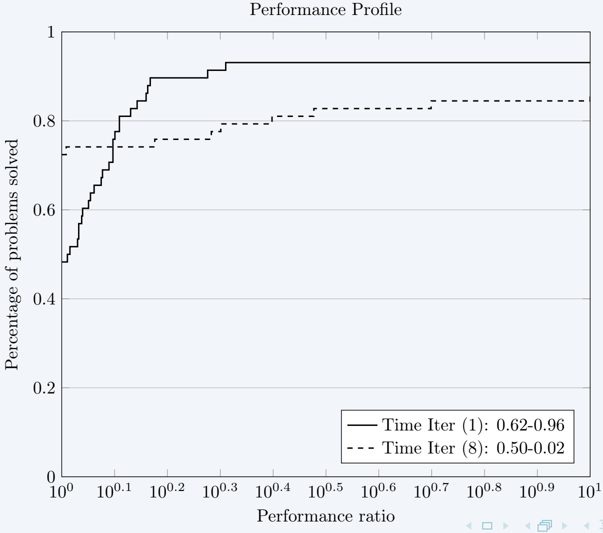
Performance Profiles

- Number of iterations;
- $(1, 0)$ and $(1, 1)$;
- Best of iterations and elapsed time;
- Best of robustness and efficiency (vs $(1, 0)$);
- Full set of 58 problems;
- Used Perprof-py [8].

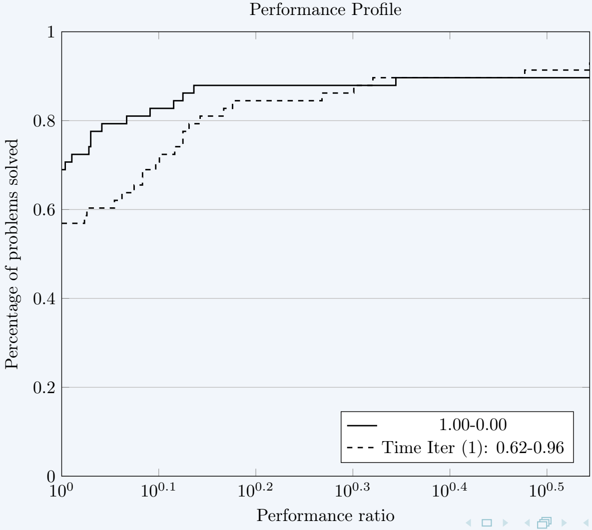
(1,0) vs (1,1)


Performance Profile
1
0.8
0.6
0.4
0.2
0
Percentage of problems solved
10^0
$10^{0.05}$
$10^{0.1}$
$10^{0.15}$
$10^{0.2}$
$10^{0.25}$
$10^{0.3}$
$10^{0.35}$
Performance ratio
1.00-0.00
1.00-1.00

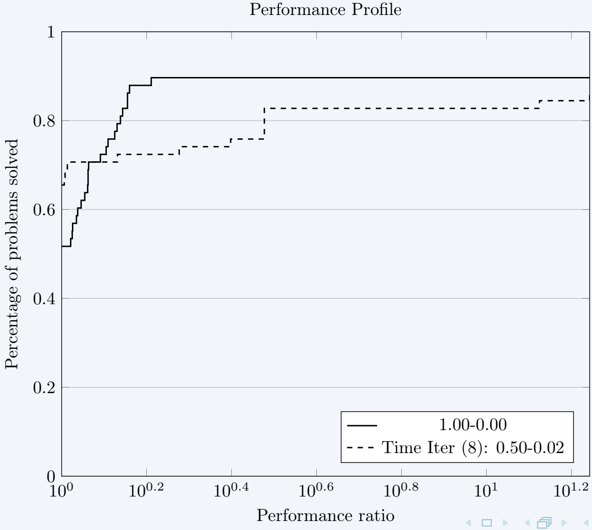
Iter best vs Time best



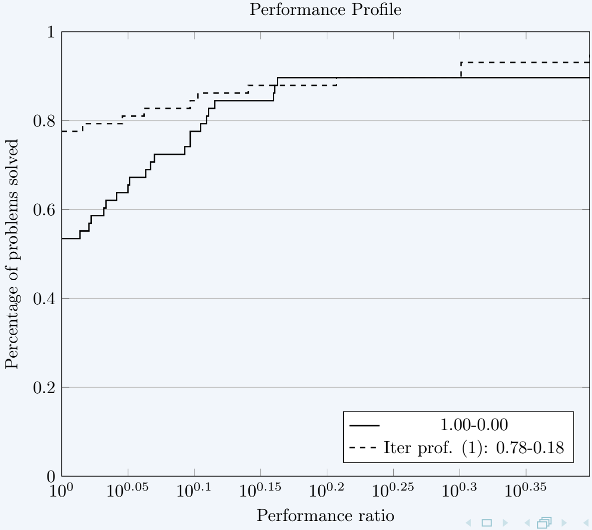
(1,0) vs Best of iter



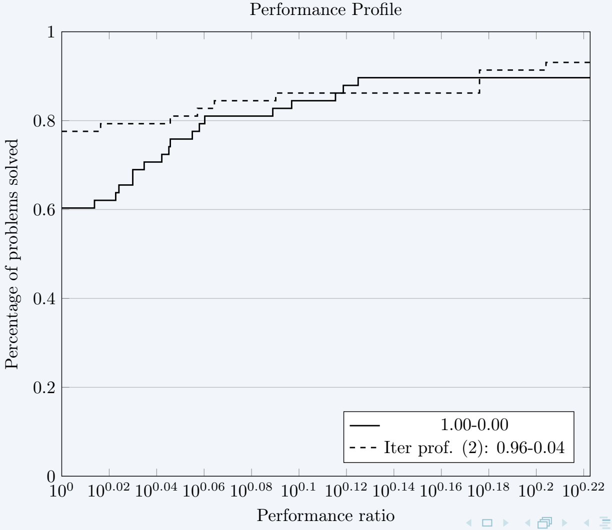
(1,0) vs Best of time



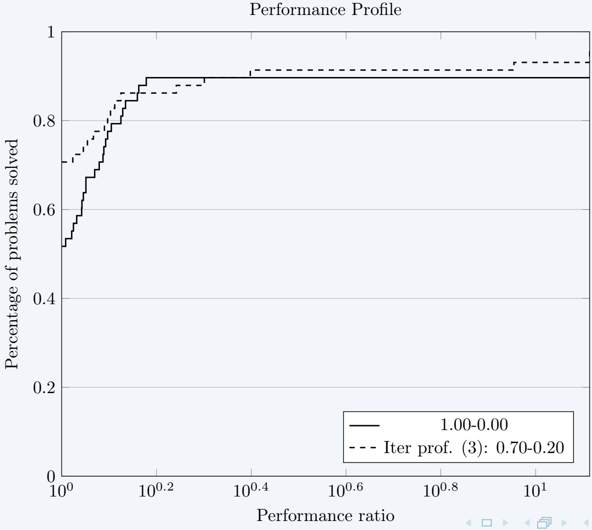
(1,0) vs Profile top



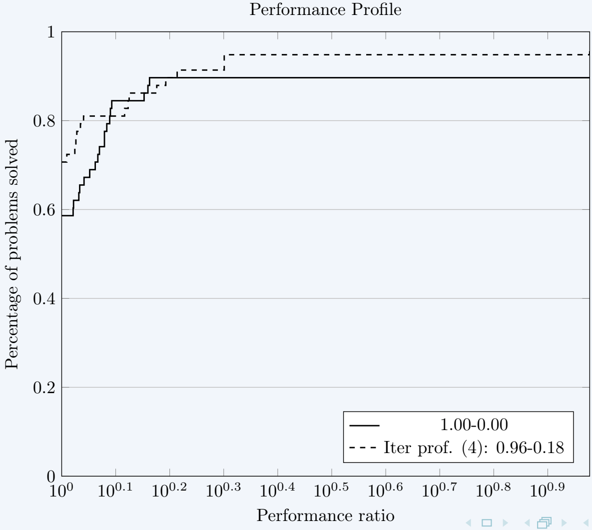
(1,0) vs Profile top



(1,0) vs Profile top

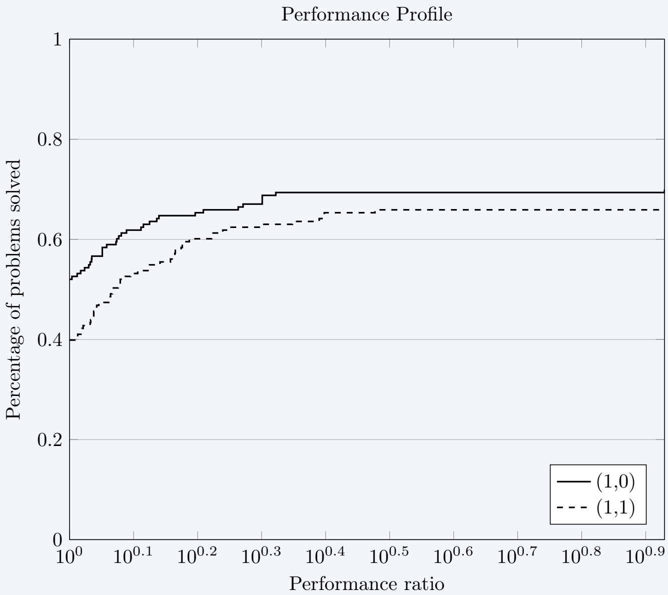


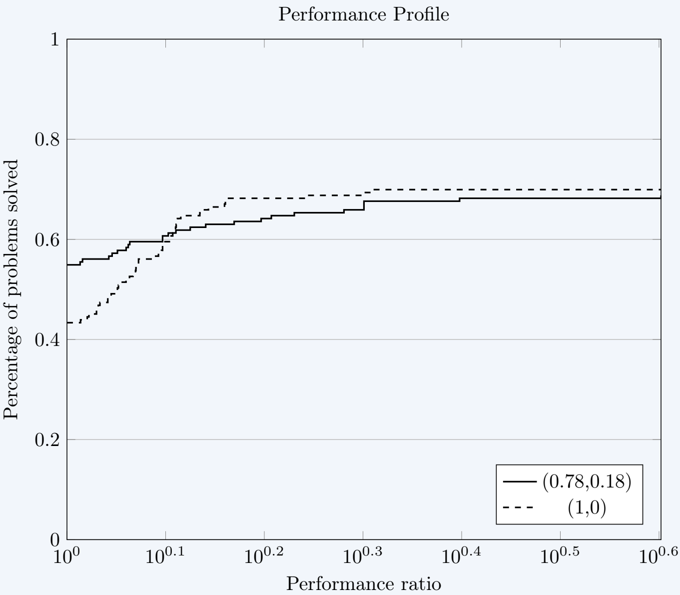
(1,0) vs Profile top

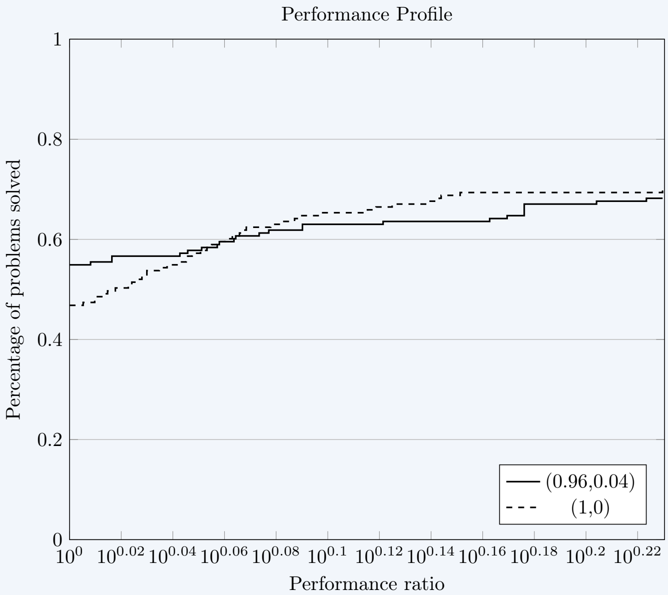


Best on full unconstrained set

- All 173 unconstrained problems without bounds.
- (1,0) vs (1,1);
- (1,0) vs (0.78,0.18), (1)
- (1,0) vs (0.96,0.04), (3)
- 1 minute, 1000 iterations.


Performance Profile
Percentage of problems solved
1
0.8
0.6
0.4
0.2
0
10^0
$10^{0.1}$
$10^{0.2}$
$10^{0.3}$
$10^{0.4}$
$10^{0.5}$
$10^{0.6}$
$10^{0.7}$
$10^{0.8}$
$10^{0.9}$
Performance ratio
(1,0)
(1,1)

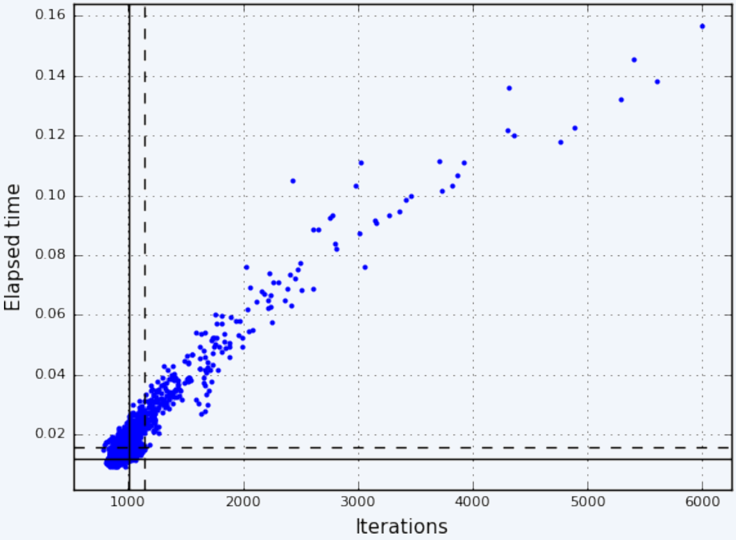

Performance Profile
Percentage of problems solved
1
0.8
0.6
0.4
0.2
0
10^0
$10^{0.1}$
$10^{0.2}$
$10^{0.3}$
$10^{0.4}$
$10^{0.5}$
$10^{0.6}$
Performance ratio
(0.78,0.18)
(1,0)


Performance Profile
Percentage of problems solved
0
0.2
0.4
0.6
0.8
1
10^0 $10^{0.02}$ $10^{0.04}$ $10^{0.06}$ $10^{0.08}$ $10^{0.1}$ $10^{0.12}$ $10^{0.14}$ $10^{0.16}$ $10^{0.18}$ $10^{0.2}$ $10^{0.22}$
Performance ratio
(0.96,0.04)
(1,0)

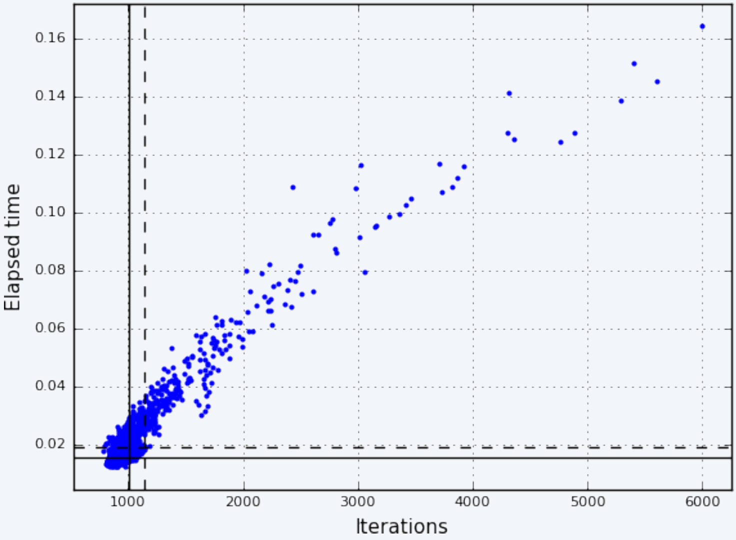
How reproducible are these results?

- Ran the complete grid again;

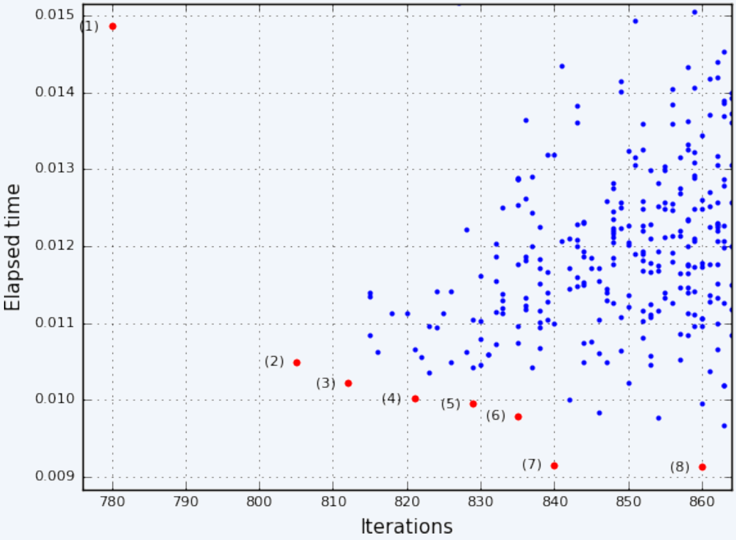
Run 1


Elapsed time
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
1000
2000
3000
4000
5000
6000
Iterations

Run 2

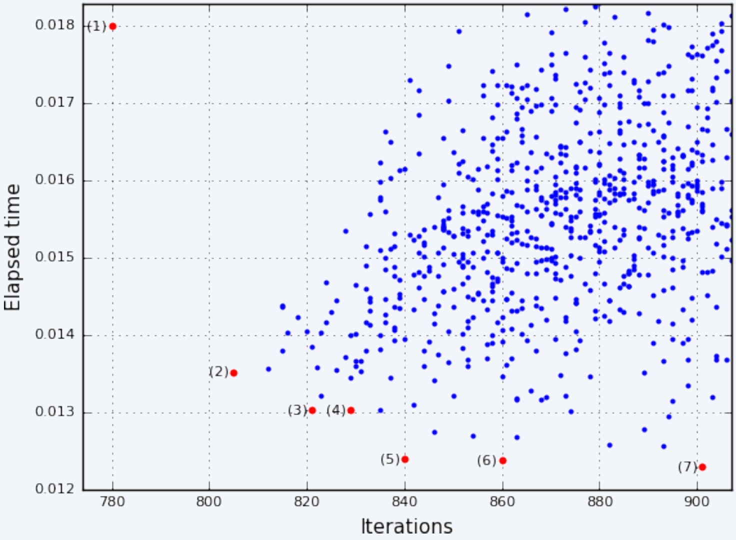

0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
Elapsed time
1000
2000
3000
4000
5000
6000
Iterations

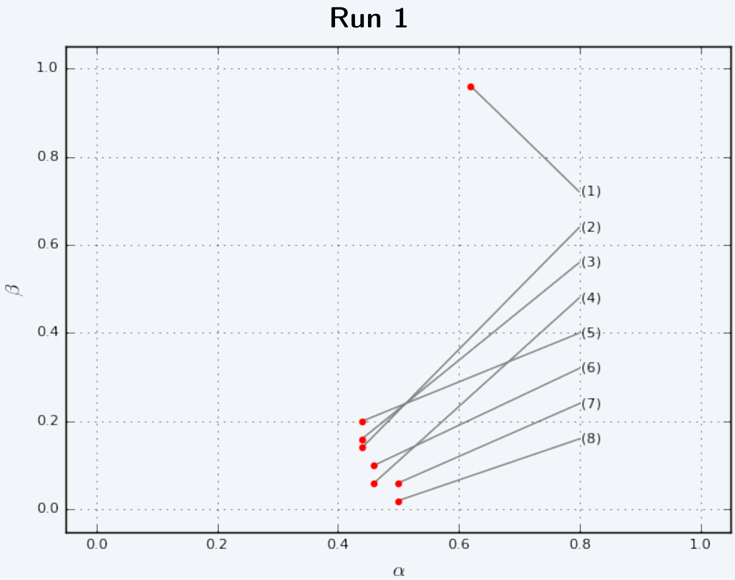
Run 1



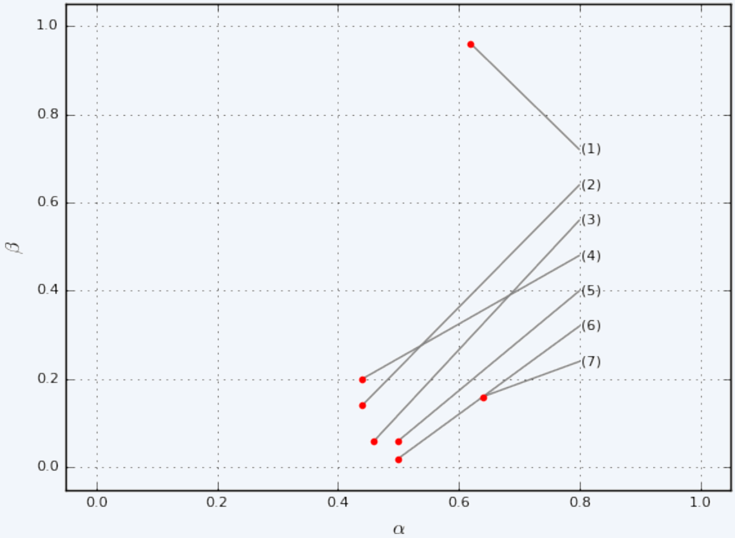
Elapsed time
Iterations
0.015
0.014
0.013
0.012
0.011
0.010
0.009
780
790
800
810
820
830
840
850
860
(1)
(2)
(3)
(4)
(5)
(6)
(7)
(8)

Run 2

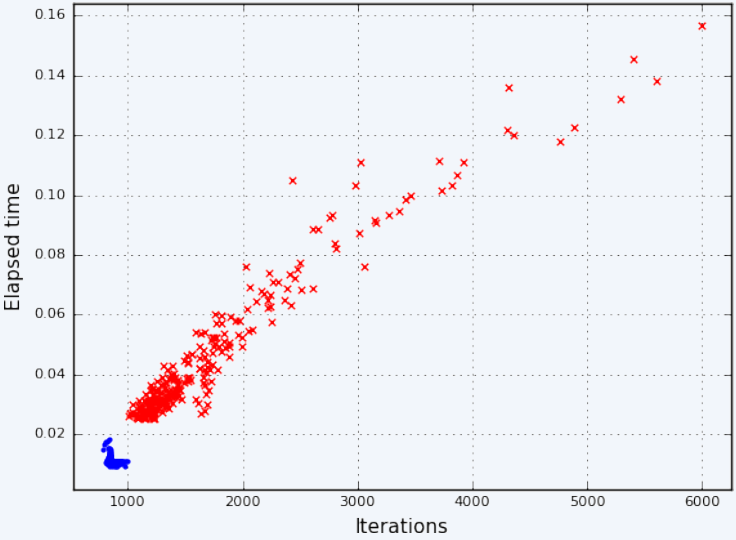

0.018
0.017
0.016
0.015
0.014
0.013
0.012
Elapsed time
780
800
820
840
860
880
900
Iterations
(1)
(2)
(3)
(4)
(5)
(6)
(7)


Run 1
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
β
α
(1)
(2)
(3)
(4)
(5)
(6)
(7)
(8)

Run 2

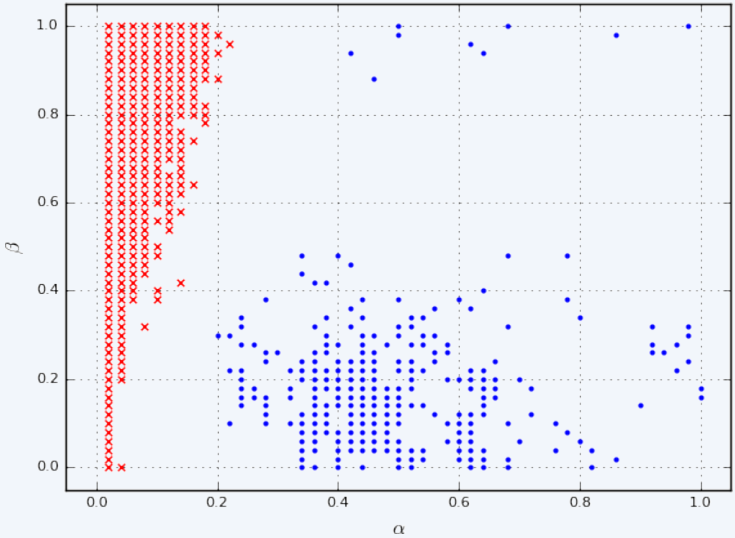


Run 1

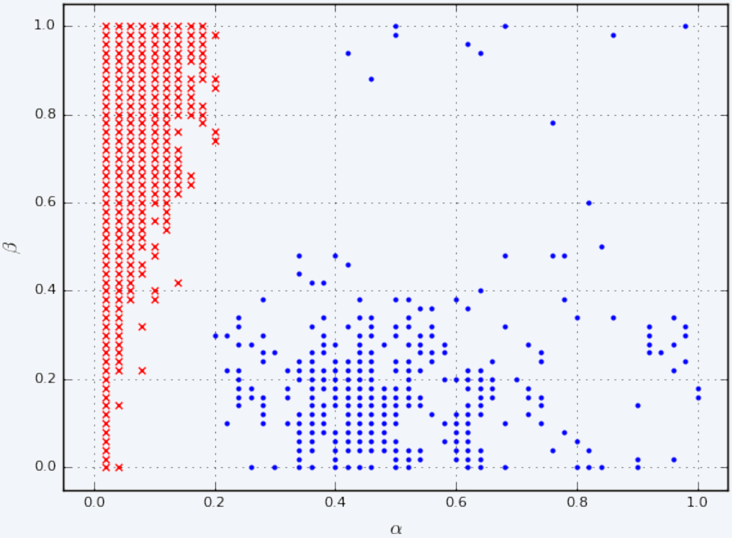

0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
Elapsed time
1000
2000
3000
4000
5000
6000
Iterations


Run 2
Elapsed time
Iterations
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
1000
2000
3000
4000
5000
6000

Run 1



Run 2



Conclusions

- The algorithm is indeed very dependent on (α, β);
- There are appears to be superior choices to be made;
- There are many better choices than $(1, 0)$ in the small set;
- The results, naturally, also depend on $\mathcal{P}$.

Future work

- Optimize other parameters for each (α, β);
- Optimize (α, β)? (Too many local minima);
- Sensitivity of this analysis regards the set of problems;
- Best choices for specific class of problems;
- Some algorithm modification that reduces sensitivity? (non-monotone);

M. J. D. Powell, "Convergence properties of a class of minimization algorithms", in *Nonlinear Programming 2*, O. L. Mangasarian, R. R. Meyer, and S. M. Robinson, Eds., Academic Press, New York, 1975.

J. Fan and Y. Yuan, "A new trust region algorithm with trust region radius converging to zero", in *Proceedings of the 5th International Conference on Optimization: Techniques and Applications (ICOTA 2001, Hong Kong)*, D. Li, Ed., 2001, pp. 786–794.

C. Cartis, N. I. M. Gould, and P. L. Toint, "Adaptive cubic overestimation methods for unconstrained optimization. part I: Motivation, convergence and numerical results.", ,

- ——,"Adaptive cubic overestimation methods for unconstrained optimization. part II: Worst-case function - and derivative - evaluation complexity", *Mathematical Programming*, vol. 130, no. 2, pp. 295–319, 2011. DOI: 10.1007/s10107-009-0337-y.
- P. L. Toint, "Nonlinear stepsize control, trust regions and regularizations for unconstrained optimization", *Optimization Methods and Software*, vol. 28, no. 1, pp. 82–95, 2013. DOI: 10.1080/10556788.2011.610458.
- G. N. Grapiglia, J. Yuan, and Y. Yuan, "Nonlinear stepsize control algorithms: Complexity bounds for first and second order optimality", UFPR, Tech. Rep., 2016.

N. I. M. Gould, D. Orban, A. Sartenaer, and P. L. Toint, "Sensitivity of trust-region algorithms to their parameters", *4OR*, vol. 3, no. 3, pp. 227–241, 2005. DOI: 10.1007/s10288-005-0065-y.

A. S. Siqueira, R. G. C. da Silva, and L.-R. Santos, "Perprof-py: A python package for performance profile of mathematical optimization software", *Journal of Open Research Software*, vol. 4, no. 1, e12, 2016. DOI: 10.5334/jors.81.

Thanks