A BFGS-SQP Method for Nonsmooth, Nonconvex, Constrained Optimization and its evaluation using Relative Minimization Profiles

Frank E. Curtis, Lehigh University
Tim Mitchell, Max Planck Institute, Magdeburg
Michael L. Overton, Courant Institute, NYU

Merida, January 2016

Under review by Opt. Meth. Software (revision submitted)

Second half of talk based on Tim Mitchell's ISMP talk

Constrained Nonsmooth Optimization

Given continuous, nonconvex and nonsmooth functions :

$$f : \mathbb{R}^n \to \mathbb{R} \quad \text{and} \quad c_i : \mathbb{R}^n \to \mathbb{R},\ i = 1, \ldots, m$$

Consider:

$$\min_x f(x) \quad \text{s.t.} \quad c_i(x) \leq 0,\ i = 1, \ldots, m$$

CONSTRAINED NONSMOOTH OPTIMIZATION

Given continuous, nonconvex and nonsmooth functions :

$$f : \mathbb{R}^n \to \mathbb{R} \quad \text{and} \quad c_i : \mathbb{R}^n \to \mathbb{R},\ i = 1, \ldots, m$$

Consider:

$$\min_x f(x) \quad \text{s.t.} \quad c_i(x) \leq 0,\ i = 1, \ldots, m$$

If f and the c_i are (locally) Lipschitz functions, they are differentiable almost everywhere, but often, at minimizers, f and the active constraints are not differentiable.

CONSTRAINED NONSMOOTH OPTIMIZATION

Given continuous, nonconvex and nonsmooth functions :

$$f : \mathbb{R}^n \to \mathbb{R} \quad \text{and} \quad c_i : \mathbb{R}^n \to \mathbb{R},\ i = 1, \ldots, m$$

Consider:

$$\min_x f(x) \quad \text{s.t.} \quad c_i(x) \leq 0,\ i = 1, \ldots, m$$

If f and the c_i are (locally) Lipschitz functions, they are differentiable almost everywhere, but often, at minimizers, f and the active constraints are not differentiable.

There are almost no published methods for such general problems, even when n is small.

Unconstrained Nonsmooth Optimization

Failure of steepest descent: known for decades. Alternatives:

Unconstrained Nonsmooth Optimization

Failure of steepest descent: known for decades. Alternatives:

Bundle methods [Lemaréchal, Kiwiel, 1980s]: collect bundle of historical gradient information to overcome discontinuity in gradients. Search direction obtained by solving a QP. Philosophy: if gradient is not defined at a point, user returns a "subgradient" instead. Guaranteed convergence to nonsmooth stationary point if f is Lipschitz.

UNCONSTRAINED NONSMOOTH OPTIMIZATION

Failure of steepest descent: known for decades. Alternatives:

Bundle methods [Lemaréchal, Kiwiel, 1980s]: collect bundle of historical gradient information to overcome discontinuity in gradients. Search direction obtained by solving a QP. Philosophy: if gradient is not defined at a point, user returns a "subgradient" instead. Guaranteed convergence to nonsmooth stationary point if f is Lipschitz.

Gradient sampling (GS) [Burke, Lewis, O. 2005, Kiwiel 2006]: sample gradients randomly near current iterate to overcome discontinuity in gradients. Search direction obtained by solving a QP, with an Armijo line search. Philosophy: user does not try to estimate whether f is differentiable at a given iterate before computing gradients. It will be with probability one, and this fails only in the limit. Guaranteed convergence to nonsmooth stationary point if f is Lipschitz.

UNCONSTRAINED NONSMOOTH OPTIMIZATION

Failure of steepest descent: known for decades. Alternatives:

Bundle methods [Lemaréchal, Kiwiel, 1980s]: collect bundle of historical gradient information to overcome discontinuity in gradients. Search direction obtained by solving a QP. Philosophy: if gradient is not defined at a point, user returns a "subgradient" instead. Guaranteed convergence to nonsmooth stationary point if f is Lipschitz.

Gradient sampling (GS) [Burke, Lewis, O. 2005, Kiwiel 2006]: sample gradients randomly near current iterate to overcome discontinuity in gradients. Search direction obtained by solving a QP, with an Armijo line search. Philosophy: user does not try to estimate whether f is differentiable at a given iterate before computing gradients. It will be with probability one, and this fails only in the limit. Guaranteed convergence to nonsmooth stationary point if f is Lipschitz.

BFGS [Broyden, Fletcher, Goldfarb, Shanno 1970; Lewis, O. 2013]: use the BFGS update H_k to the full Hessian approximation, with a weak Wolfe line search. Philosophy: H_k becomes very ill conditioned because of discontinuities in gradient, but this is desirable, and leads to automatic identification of U and V spaces on which f is smooth/nonsmooth. No theory except in very special cases, but extremely reliable in practice, and much faster than Bundle and GS.

I : SEQUENTIAL FIXED PENALTY PARAMETER (SFPP)

Consider the exact nonsmooth penalty function
[Conn et al, 1977, for smooth NLP]

$$\phi(x;\mu) = \mu f(x) + v(x)$$

where μ is a penalty parameter and

$$v(x) = \|\max\{c(x), 0\}\|_1 = \sum_{c_i(x)>0} c_i(x).$$

I : SEQUENTIAL FIXED PENALTY PARAMETER (SFPP)

Consider the exact nonsmooth penalty function
[Conn et al, 1977, for smooth NLP]

$$\phi(x;\mu) = \mu f(x) + v(x)$$

where μ is a penalty parameter and

$$v(x) = \|\max\{c(x), 0\}\|_1 = \sum_{c_i(x)>0} c_i(x).$$

Use a method for unconstrained nonsmooth optimization to repeatedly minimize penalty function, adjusting μ after each minimization if necessary until feasible.

I : SEQUENTIAL FIXED PENALTY PARAMETER (SFPP)

Consider the exact nonsmooth penalty function
[Conn et al, 1977, for smooth NLP]

$$\phi(x;\mu) = \mu f(x) + v(x)$$

where μ is a penalty parameter and

$$v(x) = \|\max\{c(x), 0\}\|_1 = \sum_{c_i(x)>0} c_i(x).$$

Use a method for unconstrained nonsmooth optimization to repeatedly minimize penalty function, adjusting μ after each minimization if necessary until feasible.

Well known disadvantages: don't know how to choose μ or how accurately to do the unconstrained minimizations.

II : SQP - GRADIENT SAMPLING (SQP-GS)

A successive quadratic programming method based on the unconstrained Gradient Sampling method.

[Curtis, O. SIOPT 2013.]

II : SQP - Gradient Sampling (SQP-GS)

A successive quadratic programming method based on the unconstrained Gradient Sampling method.
[Curtis, O. SIOPT 2013.]

Samples gradients of c_i as well as f.

II : SQP - GRADIENT SAMPLING (SQP-GS)

A successive quadratic programming method based on the unconstrained Gradient Sampling method.
[Curtis, O. SIOPT 2013.]

Samples gradients of c_i as well as f.

Basic search direction computed as in a penalty-SQP method [Fletcher 1987], but with additional constraints in the QP corresponding to the sampled constraint gradients.

II : SQP - GRADIENT SAMPLING (SQP-GS)

A successive quadratic programming method based on the unconstrained Gradient Sampling method.
[Curtis, O. SIOPT 2013.]

Samples gradients of c_i as well as f.

Basic search direction computed as in a penalty-SQP method [Fletcher 1987], but with additional constraints in the QP corresponding to the sampled constraint gradients.

Convergence guarantees to "nonsmooth stationary points" when f and c_i are Lipschitz.

II : SQP - GRADIENT SAMPLING (SQP-GS)

A successive quadratic programming method based on the unconstrained Gradient Sampling method.
[Curtis, O. SIOPT 2013.]

Samples gradients of c_i as well as f.

Basic search direction computed as in a penalty-SQP method [Fletcher 1987], but with additional constraints in the QP corresponding to the sampled constraint gradients.

Convergence guarantees to "nonsmooth stationary points" when f and c_i are Lipschitz.

Computationally intensive: requires function and constraint gradient evaluations at $n + 1$ points per iteration.

III : BFGS-SQP: A New Method

A successive quadratic programming method based on the unconstrained BFGS method.

III : BFGS-SQP: A NEW METHOD

A successive quadratic programming method based on the unconstrained BFGS method.

As in SQP-GS, basic search direction computed as in a penalty-SQP method [Fletcher 1987], solving the QP

$$\min_{d\in\mathbb{R}^n, s\in\mathbb{R}^m} \mu(f(x_k) + \nabla f(x_k)^\mathsf{T} d) + e^\mathsf{T} s + \frac{1}{2} d^\mathsf{T} H_k d$$

$$\text{s.t. } c(x_k) + \nabla c(x_k)^\mathsf{T} d \leq s, \quad s \geq 0.$$

III : BFGS-SQP: A NEW METHOD

A successive quadratic programming method based on the unconstrained BFGS method.

As in SQP-GS, basic search direction computed as in a penalty-SQP method [Fletcher 1987], solving the QP

$$\min_{d \in \mathbb{R}^n, s \in \mathbb{R}^m} \mu(f(x_k) + \nabla f(x_k)^\mathsf{T} d) + e^\mathsf{T} s + \frac{1}{2} d^\mathsf{T} H_k d$$

$$\text{s.t. } c(x_k) + \nabla c(x_k)^\mathsf{T} d \leq s, \quad s \geq 0.$$

Uses $\phi(x; \mu)$ as a merit function, but unlike in SQP-GS, uses a "steering strategy" to update μ.

III : BFGS-SQP: A NEW METHOD

A successive quadratic programming method based on the unconstrained BFGS method.

As in SQP-GS, basic search direction computed as in a penalty-SQP method [Fletcher 1987], solving the QP

$$\min_{d\in\mathbb{R}^n, s\in\mathbb{R}^m} \mu(f(x_k) + \nabla f(x_k)^\mathsf{T} d) + e^\mathsf{T} s + \frac{1}{2} d^\mathsf{T} H_k d$$
$$\text{s.t. } c(x_k) + \nabla c(x_k)^\mathsf{T} d \le s, \quad s \ge 0.$$

Uses $\phi(x;\mu)$ as a merit function, but unlike in SQP-GS, uses a "steering strategy" to update μ.

No convergence theory at all.

III : BFGS-SQP: A NEW METHOD

A successive quadratic programming method based on the unconstrained BFGS method.

As in SQP-GS, basic search direction computed as in a penalty-SQP method [Fletcher 1987], solving the QP

$$\min_{d\in\mathbb{R}^n, s\in\mathbb{R}^m} \mu(f(x_k) + \nabla f(x_k)^\mathsf{T} d) + e^\mathsf{T} s + \frac{1}{2} d^\mathsf{T} H_k d$$
$$\text{s.t. } c(x_k) + \nabla c(x_k)^\mathsf{T} d \le s, \quad s \ge 0.$$

Uses $\phi(x;\mu)$ as a merit function, but unlike in SQP-GS, uses a "steering strategy" to update μ.

No convergence theory at all.

Computationally very efficient compared to SQP-GS.

III : BFGS-SQP: A NEW METHOD

A successive quadratic programming method based on the unconstrained BFGS method.

As in SQP-GS, basic search direction computed as in a penalty-SQP method [Fletcher 1987], solving the QP

$$\min_{d\in\mathbb{R}^n, s\in\mathbb{R}^m} \mu(f(x_k) + \nabla f(x_k)^\mathsf{T} d) + e^\mathsf{T} s + \frac{1}{2} d^\mathsf{T} H_k d$$
$$\text{s.t. } c(x_k) + \nabla c(x_k)^\mathsf{T} d \leq s, \quad s \geq 0.$$

Uses $\phi(x;\mu)$ as a merit function, but unlike in SQP-GS, uses a "steering strategy" to update μ.

No convergence theory at all.

Computationally very efficient compared to SQP-GS.

Point of this talk: not to explain the details of the algorithm, but to evaluate how well it works in practice, compared to other methods, on some challenging applications.

SUMMARIZING: COMPARISON OF FOUR METHODS

All codes run with default parameters except with very small termination tolerances and a maximum number of 500 iterations.

SUMMARIZING: COMPARISON OF FOUR METHODS

All codes run with default parameters except with very small termination tolerances and a maximum number of 500 iterations.

- SFPP (Sequential Fixed Penalty Parameter)
  - repeatedly apply BFGS to minimize nonsmooth penalty function
  - BFGS is reliable and efficient for nonsmooth unconstrained problems, although theory is very limited
  - disadvantages of sequential penalty methods are well known

Summarizing: Comparison of Four Methods

All codes run with default parameters except with very small termination tolerances and a maximum number of 500 iterations.

- SFPP (Sequential Fixed Penalty Parameter)
  - repeatedly apply BFGS to minimize nonsmooth penalty function
  - BFGS is reliable and efficient for nonsmooth unconstrained problems, although theory is very limited
  - disadvantages of sequential penalty methods are well known
- SQP-GS (SQP Gradient Sampling) [Curtis and O. 2012]
  - guaranteed convergence for nonsmooth, nonconvex, Lipschitz constrained optimization
  - gradient sampling requires $n+1$ gradient evaluations per iteration

SUMMARIZING: COMPARISON OF FOUR METHODS

All codes run with default parameters except with very small termination tolerances and a maximum number of 500 iterations.

- SFPP (Sequential Fixed Penalty Parameter)
  - repeatedly apply BFGS to minimize nonsmooth penalty function
  - BFGS is reliable and efficient for nonsmooth unconstrained problems, although theory is very limited
  - disadvantages of sequential penalty methods are well known
- SQP-GS (SQP Gradient Sampling) [Curtis and O. 2012]
  - guaranteed convergence for nonsmooth, nonconvex, Lipschitz constrained optimization
  - gradient sampling requires $n+1$ gradient evaluations per iteration
- BFGS-SQP [our new method]
  - uses a nonsmooth penalty function with a steering strategy
  - no theoretical guarantees

SUMMARIZING: COMPARISON OF FOUR METHODS

All codes run with default parameters except with very small termination tolerances and a maximum number of 500 iterations.

- SFPP (Sequential Fixed Penalty Parameter)
  - repeatedly apply BFGS to minimize nonsmooth penalty function
  - BFGS is reliable and efficient for nonsmooth unconstrained problems, although theory is very limited
  - disadvantages of sequential penalty methods are well known
- SQP-GS (SQP Gradient Sampling) [Curtis and O. 2012]
  - guaranteed convergence for nonsmooth, nonconvex, Lipschitz constrained optimization
  - gradient sampling requires $n+1$ gradient evaluations per iteration
- BFGS-SQP [our new method]
  - uses a nonsmooth penalty function with a steering strategy
  - no theoretical guarantees
- SNOPT [Gill, Murray and Saunders 2002]
  - a well regarded code for nonlinearly constrained problems
  - not intended for nonsmooth objective or constraints
  - only one of the four solvers that is compiled code
  - suggested by OMS editor as a benchmark/sanity check

Spectral Radius Optimization

The spectral radius of a matrix $M \in \mathbb{C}^{N \times N}$ is

$$\rho(M) := \max\{|\lambda| : \lambda \in \sigma(M)\}$$

where the spectrum, or set of eigenvalues of M, is

$$\sigma(M) = \{\lambda \in \mathbb{C} : \det(M - \lambda I) = 0\}.$$

We say that M is (marginally) stable if $\rho(M) \leq 1$.

Spectral Radius Optimization

The spectral radius of a matrix $M \in \mathbb{C}^{N \times N}$ is

$$\rho(M) := \max\{|\lambda| : \lambda \in \sigma(M)\}$$

where the spectrum, or set of eigenvalues of M, is

$$\sigma(M) = \{\lambda \in \mathbb{C} : \det(M - \lambda I) = 0\}.$$

We say that M is (marginally) stable if $\rho(M) \leq 1$.
Consider:

$$\min_{X \in \mathbb{R}^{M \times P}} \max\{\rho(A_i + B_i X C_i) : i \in \{m+1, \ldots, m+q\}\}$$
$$\text{s.t. } \rho(A_i + B_i X C_i) \leq 1, \; i \in \{1, \ldots, m\}$$

A nonsmooth, nonconvex, non-Lipschitz optimization problem arising in control design via "static output feedback".

Spectral Radius Optimization

The spectral radius of a matrix $M \in \mathbb{C}^{N \times N}$ is

$$\rho(M) := \max\{|\lambda| : \lambda \in \sigma(M)\}$$

where the spectrum, or set of eigenvalues of M, is

$$\sigma(M) = \{\lambda \in \mathbb{C} : \det(M - \lambda I) = 0\}.$$

We say that M is (marginally) stable if $\rho(M) \leq 1$.
Consider:

$$\min_{X \in \mathbb{R}^{M \times P}} \max\{\rho(A_i + B_i X C_i) : i \in \{m+1, \ldots, m+q\}\}$$

$$\text{s.t. } \rho(A_i + B_i X C_i) \leq 1, \; i \in \{1, \ldots, m\}$$

A nonsmooth, nonconvex, non-Lipschitz optimization problem arising in control design via "static output feedback".

Data matrices A_i, B_i, C_i generated randomly and scaled such that $\rho(A_i) < 1, i = 1, \ldots, m$, and hence $X = 0$ is a feasible initial point.

SPECTRAL RADIUS OPTIMIZATION

The spectral radius of a matrix $M \in \mathbb{C}^{N \times N}$ is

$$\rho(M) := \max\{|\lambda| : \lambda \in \sigma(M)\}$$

where the spectrum, or set of eigenvalues of M, is

$$\sigma(M) = \{\lambda \in \mathbb{C} : \det(M - \lambda I) = 0\}.$$

We say that M is (marginally) stable if $\rho(M) \leq 1$.
Consider:

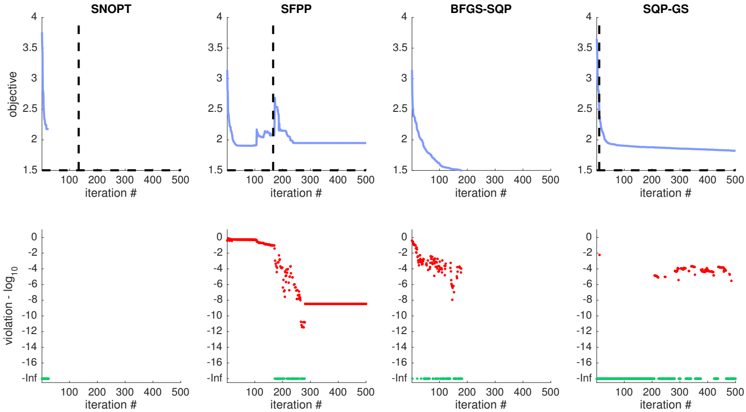
$$\min_{X \in \mathbb{R}^{M \times P}} \ \max\{\rho(A_i + B_i X C_i) : i \in \{m+1, \ldots, m+q\}\}$$
$$\text{s.t. } \rho(A_i + B_i X C_i) \leq 1, \ i \in \{1, \ldots, m\}$$

A nonsmooth, nonconvex, non-Lipschitz optimization problem arising in control design via "static output feedback".

Data matrices A_i, B_i, C_i generated randomly and scaled such that $\rho(A_i) < 1, i = 1, \ldots, m$, and hence $X = 0$ is a feasible initial point.

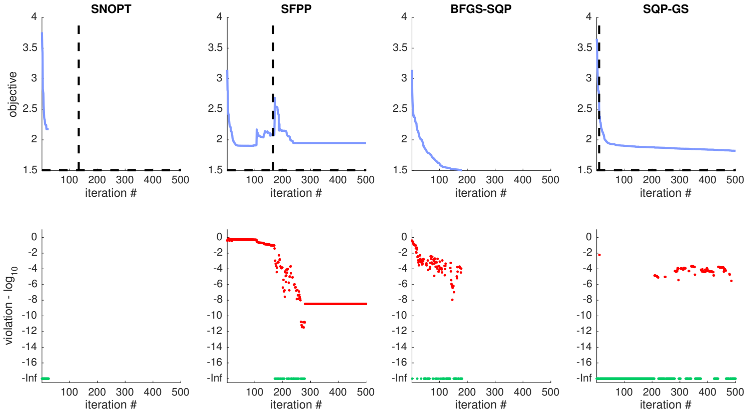
The gradient of the spectral radius can be computed from the right and left eigenvectors for the eigenvalue with largest modulus, assuming this is unique and simple — which it will be with probability one, failing only in the limit.

A Spectral Radius Sample Problem



Tracking Objective and Violation Values

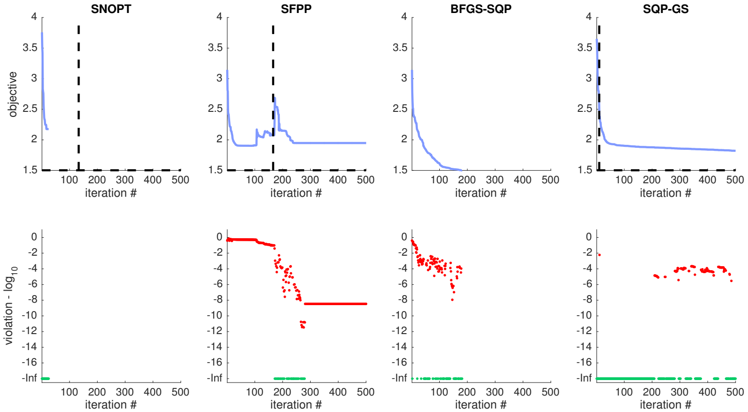
A SPECTRAL RADIUS SAMPLE PROBLEM



Tracking Objective and Violation Values

BFGS-SQP finds the best result in this case

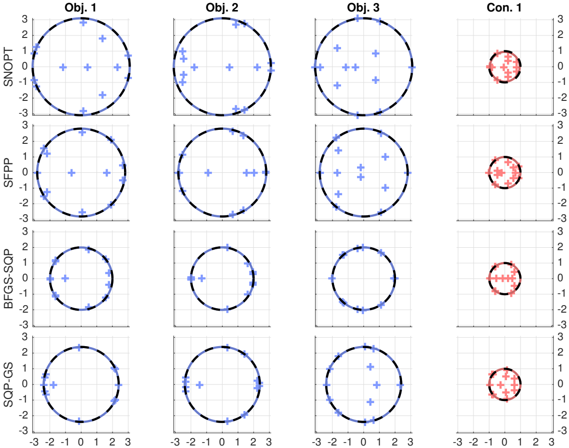
A Spectral Radius Sample Problem



Tracking Objective and Violation Values

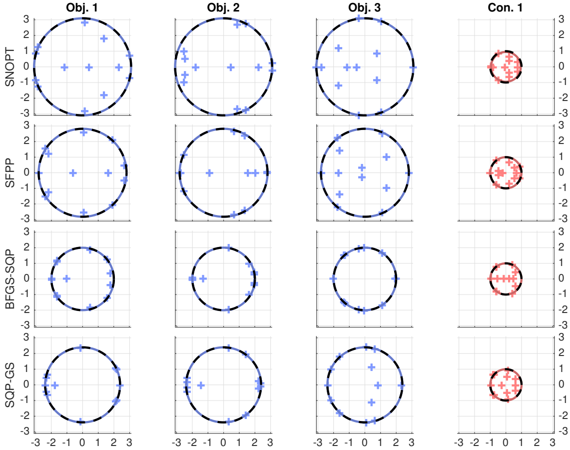
BFGS-SQP finds the best result in this case
— but this is just one problem

FINAL SPECTRAL CONFIGURATIONS



These plots are in the complex plane: +'s are eigenvalues

Final Spectral Configurations



These plots are in the complex plane: +'s are eigenvalues

Note the "ties" for the objective max values and constraint activity: indicates nonsmoothness in objective and constraint in the limit.

Pseudospectral Radius Optimization

The pseudospectral radius of a matrix $M \in \mathbb{C}^{N \times N}$ is

$$\rho_\varepsilon(M) := \max\{|\lambda| : \lambda \in \sigma(M + \Delta), \Delta \in \mathbb{C}^{N \times N}, \|\Delta\|_2 \leq \varepsilon\},$$

where $\rho_0(M) = \rho(M)$. We say that M is (marginally) stable with respect to the perturbation level $\varepsilon > 0$ if $\rho_\varepsilon(M) \leq 1$.

PSEUDOSPECTRAL RADIUS OPTIMIZATION

The pseudospectral radius of a matrix $M \in \mathbb{C}^{N \times N}$ is

$$\rho_\varepsilon(M) := \max\{|\lambda| : \lambda \in \sigma(M + \Delta), \Delta \in \mathbb{C}^{N \times N}, \|\Delta\|_2 \leq \varepsilon\},$$

where $\rho_0(M) = \rho(M)$. We say that M is (marginally) stable with respect to the perturbation level $\varepsilon > 0$ if $\rho_\varepsilon(M) \leq 1$.
For fixed ε, consider

$$\begin{aligned} \min_{X \in \mathbb{R}^{M \times P}} \quad & \max\{\rho_\varepsilon(A_i + B_i X C_i) : i \in \{m+1, \ldots, m+q\}\} \\ \text{s.t.} \quad & \rho_\varepsilon(A_i + B_i X C_i) \leq 1, \; i \in \{1, \ldots, m\} \end{aligned}$$

A nonsmooth, nonconvex, Lipschitz optimization problem arising in control design via "static output feedback".

PSEUDOSPECTRAL RADIUS OPTIMIZATION

The pseudospectral radius of a matrix $M \in \mathbb{C}^{N \times N}$ is

$$\rho_\varepsilon(M) := \max\{|\lambda| : \lambda \in \sigma(M + \Delta), \Delta \in \mathbb{C}^{N \times N}, \|\Delta\|_2 \leq \varepsilon\},$$

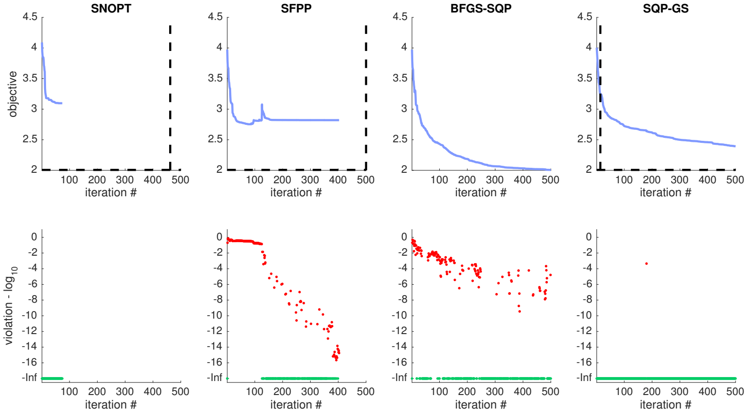
where $\rho_0(M) = \rho(M)$. We say that M is (marginally) stable with respect to the perturbation level $\varepsilon > 0$ if $\rho_\varepsilon(M) \leq 1$.
For fixed ε, consider

$$\min_{X \in \mathbb{R}^{M \times P}} \quad \max\{\rho_\varepsilon(A_i + B_i X C_i) : i \in \{m+1, \ldots, m+q\}\}$$
$$\text{s.t.} \quad \rho_\varepsilon(A_i + B_i X C_i) \leq 1, \; i \in \{1, \ldots, m\}$$

A nonsmooth, nonconvex, Lipschitz optimization problem arising in control design via "static output feedback".

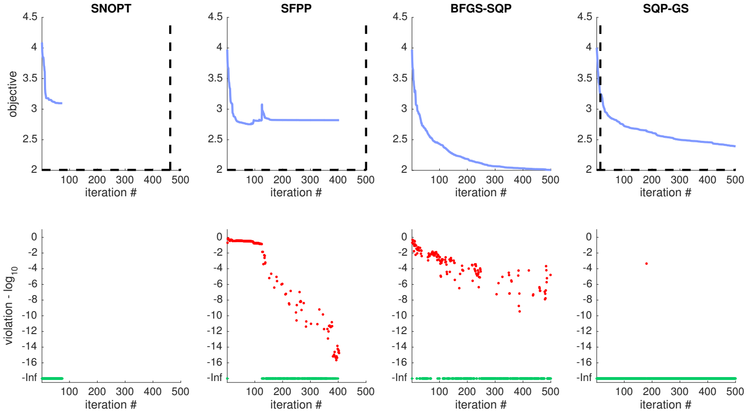
Computation of ρ_ε: [Burke, Lewis, O. 2003]. Its gradient exists with probability one, failing only in the limit.

A PSEUDOSPECTRAL RADIUS SAMPLE PROBLEM



Objective and Violation Values

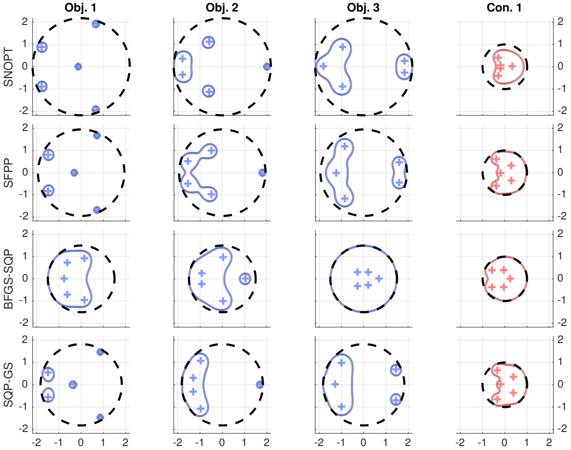
A PSEUDOSPECTRAL RADIUS SAMPLE PROBLEM



Objective and Violation Values

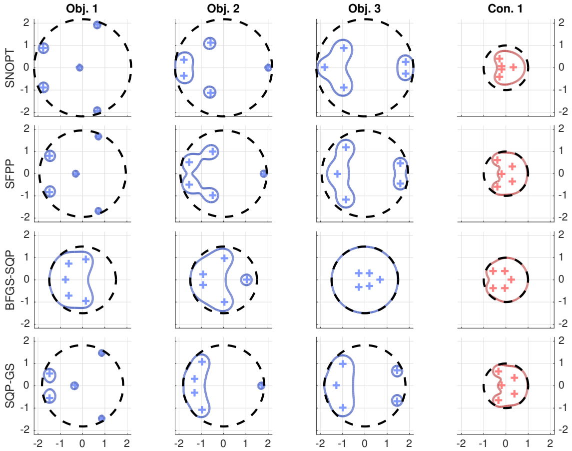
Again, BFGS-SQP is the best.

Final Pseudospectral Configurations



Solid curves are ε-pseudospectral boundaries

Final Pseudospectral Configurations



Solid curves are ε-pseudospectral boundaries

Note the “ties" for the objective max values and constraint activity: indicates nonsmoothness in objective and constraint in the limit.

Benchmarking Optimization Software

For a comparative benchmark of solvers $s \in \mathcal{S}$ on problems $p \in \mathcal{P}$, we would like a data representation that:

BENCHMARKING OPTIMIZATION SOFTWARE

For a comparative benchmark of solvers $s \in \mathcal{S}$ on problems $p \in \mathcal{P}$, we would like a data representation that:

- depicts the overall comparative performances for all $s \in \mathcal{S}$

Benchmarking Optimization Software

For a comparative benchmark of solvers $s \in \mathcal{S}$ on problems $p \in \mathcal{P}$, we would like a data representation that:

- depicts the overall comparative performances for all $s \in \mathcal{S}$
- is informative and easy to read

Benchmarking Optimization Software

For a comparative benchmark of solvers $s \in \mathcal{S}$ on problems $p \in \mathcal{P}$, we would like a data representation that:

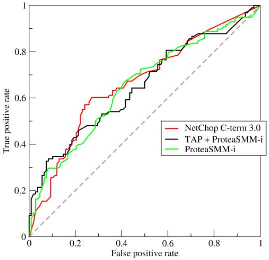
- depicts the overall comparative performances for all $s \in \mathcal{S}$
- is informative and easy to read
- is not sensitive to just a subset of problems (e.g. difficult or large ones)

Benchmarking Optimization Software

For a comparative benchmark of solvers $s \in \mathcal{S}$ on problems $p \in \mathcal{P}$, we would like a data representation that:

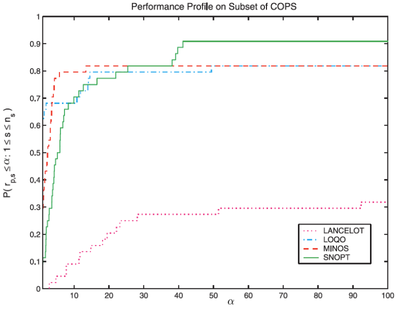
- depicts the overall comparative performances for all $s \in \mathcal{S}$
- is informative and easy to read
- is not sensitive to just a subset of problems (e.g. difficult or large ones)
- does not require or is not sensitive to parameters to generate:
  - parameters of the benchmark itself
  - parameters of the solvers

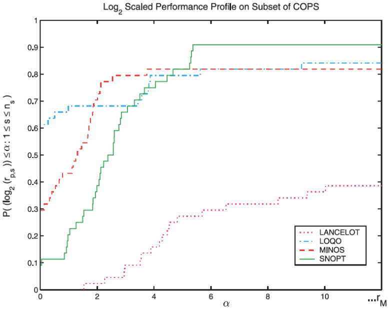
Receiver Operating Characteristic



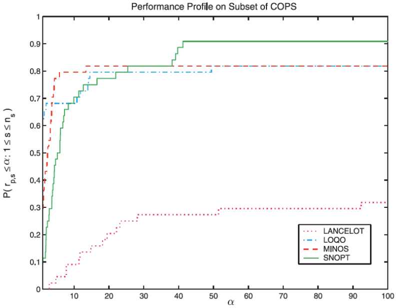
- ROC or ROC Curve
  [Developed by electrical/radar engineers during WWII]
- Popular in psychology, medicine, radiology, biometrics, and now machine learning, data mining too
- Plots the performance of a binary classifier dependent upon its discrimination parameter (sensitivity)
  - Relates the true positive rate with the false positive rate *as a classifier is tuned*
- More area under the curve indicates better performance

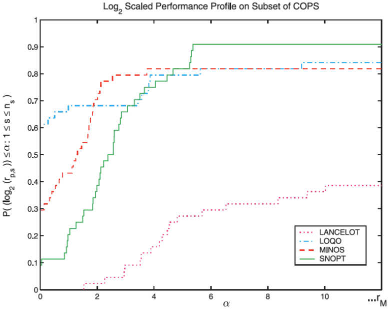
Performance Profiles [Dolan and Moré 2002]





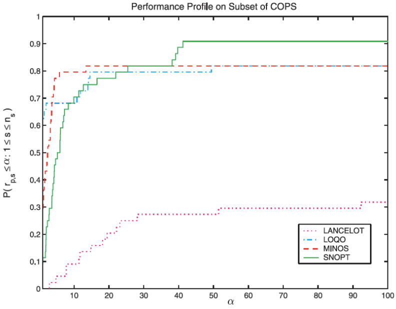
Performance Profiles [Dolan and Moré 2002]

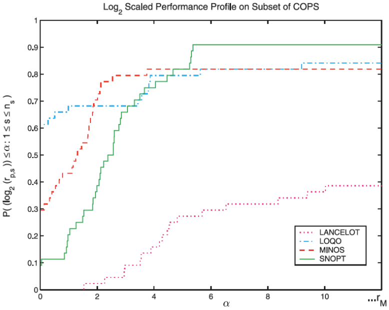




- Now widely used to benchmark numerical software (1692 Google Scholar citations)
- More area under a curve is better
- Plots how a solver's rate of per-problem success on $\mathcal{P}$ changes, using a binary classification of success/failure, as some allowable performance limit is varied

PERFORMANCE PROFILES [DOLAN AND MORÉ 2002]





- Usually the performance metric is running time
- A plot passing thru (α, y) indicates a solver $s \in \mathcal{S}$:
  - successfully solved $100 \times y$ percent of test set $\mathcal{P}$
  - provided that s was only allowed at most α-times as much time as the *fastest successful solver per problem*
    (i.e. taking longer is considered a failure for that value of α)
  - $\alpha = 1$ gives overall "first to answer" performance

Performance Profiles: Pros and Cons

Benefits:

- easily understood
- comprehensive, measures failures
- not sensitive to:
  - heterogenous test sets (w.r.t. difficulty or dimension)
  - perturbations to the performance ratios
- natural fit for convex programs (with or without constraints) — because then we expect all solvers to find the same answer eventually

Performance Profiles: Pros and Cons

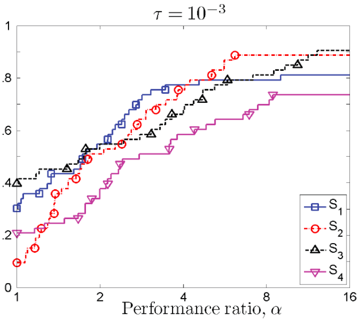
Benefits:

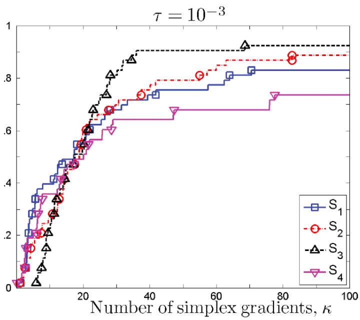
- easily understood
- comprehensive, measures failures
- not sensitive to:
  - heterogenous test sets (w.r.t. difficulty or dimension)
  - perturbations to the performance ratios
- natural fit for convex programs (with or without constraints) — because then we expect all solvers to find the same answer eventually

Limitations:

- requires a binary success/failure test (e.g. on target values, $\|\nabla f\|$)
  - success tolerance is fixed, how should we choose it?
  - performance profile curve is potentially sensitive to this choice
  - no credit is given for progress made
  - what is a good success/failure metric for nonconvex problems?
- doesn't allow computational budgets

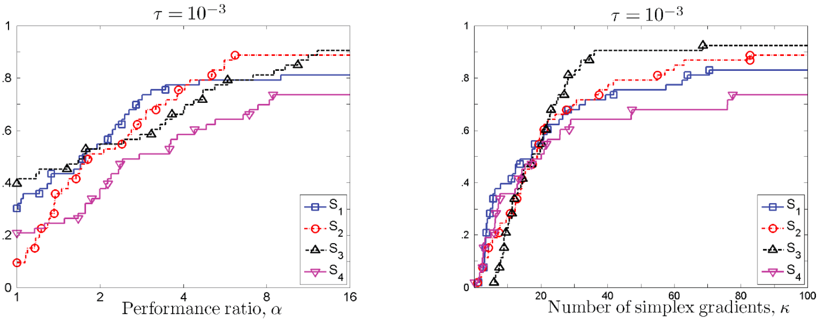
Data Profiles [Moré and Wild 2009]





- Motivated for benchmarking solvers for derivative-free optimization:
  - solvers may find low or high accuracy solutions
  - there may be constraints on the computational budget
  - want to know the relationship between accuracy and cost
  - performance profiles don't depict progress towards solutions
- Similar looking to performance profiles but not the same
- Again, more area under a curve is better

Data Profiles [Moré and Wild 2009]



- Proposed data profiles (right) to be used with performance profiles (left), using a convergence test, to depict complementary information
- Performance profiles compare solvers relative to each other
- Data profiles are designed to assess short-term behavior: plots the percentage of problems solved (to a tolerance) dependent upon on the number of function evaluations.

What Do We Care to Assess For a Benchmark?

For nonsmooth, nonconvex constrained optimization, we wish to evaluate four algorithms over two test sets, each of 100 problems with randomly generated data: one set of Lipschitz pseudospectral radius optimization problems, the other a set of non-Lipschitz spectral radius optimization problems, simultaneously in terms of:

What Do We Care to Assess For a Benchmark?

For nonsmooth, nonconvex constrained optimization, we wish to evaluate four algorithms over two test sets, each of 100 problems with randomly generated data: one set of Lipschitz pseudospectral radius optimization problems, the other a set of non-Lipschitz spectral radius optimization problems, simultaneously in terms of:

- reliability:
  - percentage of feasible solutions found over the test set
  - constraint violation should be less than some tolerance (can be zero for inequalities - we use this here)

What Do We Care to Assess For a Benchmark?

For nonsmooth, nonconvex constrained optimization, we wish to evaluate four algorithms over two test sets, each of 100 problems with randomly generated data: one set of Lipschitz pseudospectral radius optimization problems, the other a set of non-Lipschitz spectral radius optimization problems, simultaneously in terms of:

- reliability:
  - percentage of feasible solutions found over the test set
  - constraint violation should be less than some tolerance (can be zero for inequalities - we use this here)
- performance:
  - quality of minimization achieved
  - e.g. the lowest objective value encountered on the feasible set

What Do We Care to Assess For a Benchmark?

For nonsmooth, nonconvex constrained optimization, we wish to evaluate four algorithms over two test sets, each of 100 problems with randomly generated data: one set of Lipschitz pseudospectral radius optimization problems, the other a set of non-Lipschitz spectral radius optimization problems, simultaneously in terms of:

- reliability:
  - percentage of feasible solutions found over the test set
  - constraint violation should be less than some tolerance (can be zero for inequalities - we use this here)
- performance:
  - quality of minimization achieved
  - e.g. the lowest objective value encountered on the feasible set
- progress versus computational cost:
  - how do the algorithms' progress compare relative to each other or some computational budget

Relative Minimization Profiles (RMPs)

Let us first focus only on reliability and performance.

For problem $p_i \in \mathcal{P}$, consider:

$$\begin{aligned} \omega_i &:= \text{target objective value} \\ \Omega &:= \{\omega_i\} \text{ (for all } p_i \in \mathcal{P}) \\ f_i(x) &:= \text{objective function} \\ v_i(x) &:= \text{violation function} \\ \{x_k\}_i^s &:= \text{iterates produced by solver } s \in \mathcal{S}. \end{aligned}$$

The best computed objective value for solver $s \in \mathcal{S}$ on problem $p_i \in \mathcal{P}$, in terms of violation tolerance $\tau_v \geq 0$ is:

$$f_i^s(\tau_v) := \min \{ f_i(x) \text{ s.t. } x \in \{x_k\}_i^s, v_i(x) \leq \tau_v \} .$$

In the absence of any *a priori* information, set the target value

$$\omega_i := \min \{ f_i^s(\tau_v) : s \in \mathcal{S} \} .$$

(as suggested for data profiles).

Relative Minimization Profiles (RMPs)

Consider the relative residual function and its associated indicator function:

$$r(\varphi, \tilde{\varphi}) := \begin{cases} \infty & \text{if } \varphi = \infty \text{ or } \tilde{\varphi} = \infty \\ \left| \frac{\varphi - \tilde{\varphi}}{\varphi} \right| & \text{otherwise,} \end{cases}$$

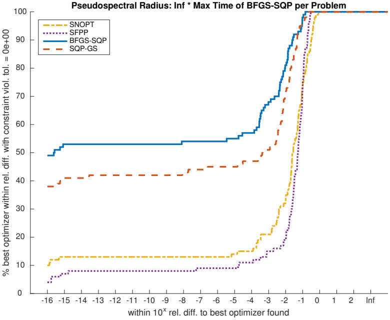
$$\mathbb{1}_r(\varphi, \tilde{\varphi}, \gamma) := \begin{cases} 1 & \text{if } r(\varphi, \tilde{\varphi}) \leq \gamma \\ 0 & \text{otherwise.} \end{cases}$$

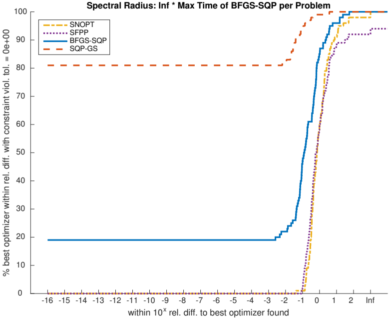
For violation tolerance $\tau_v \geq 0$, per-problem target values $\Omega := \{\omega_i\}$, and solver $s \in \mathcal{S}$, its relative minimization profile curve is defined as:

$$r^{s,\infty}_{\Omega,\tau_v}(\gamma) := \frac{1}{|\mathcal{P}|} \sum_{i=1}^{|\mathcal{P}|} \mathbb{1}_r\left(\omega_i, f_i^s(\tau_v), \gamma\right),$$

where γ specifies the max relative difference allowed w.r.t. $\omega_i \in \Omega$.

RELATIVE MINIMIZATION PROFILES (RMPS)



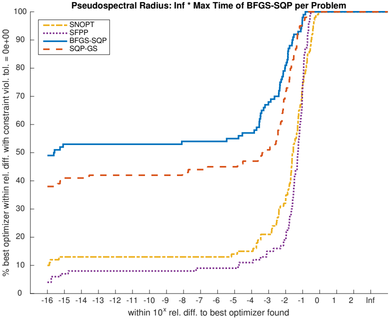


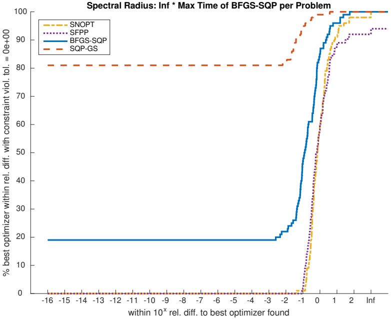
Pseudospectral radius test set
Lipschitz
BFGS-SQP wins
(despite no theory)

Spectral radius test set
Not Lipschitz
SQP-GS wins (but takes a long time)
(although its theory not relevant)

Note the Inf: no restrictions on running time

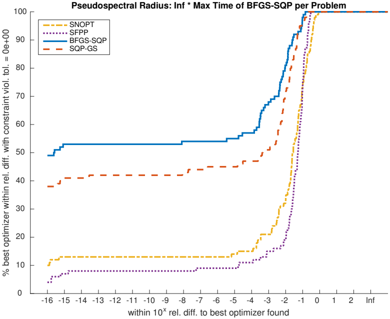
Relative Minimization Profiles (RMPs)

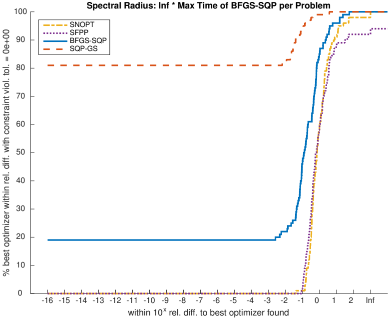




- (γ, y) plots the percentage of problems that a solver encountered:
    - feasible iterates which were also
    - within a relative difference γ of the best known objective values
- no convergence test: success/failure classification is no longer fixed

Relative Minimization Profiles (RMPs)

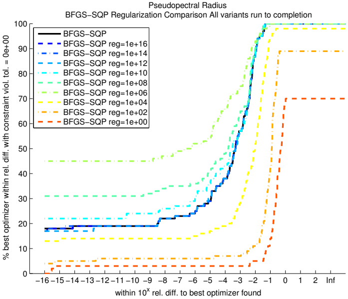


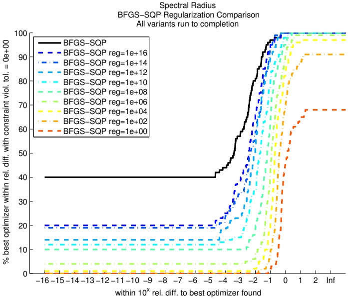


Like an ROC Curve, an RMP shows the effect of tuning the convergence/success classifier over its entire range:

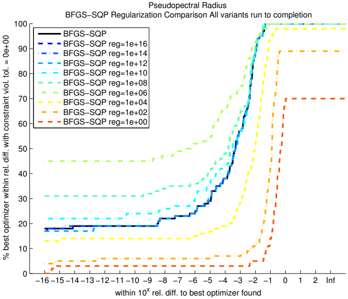
- $\gamma = \varepsilon_{mach}$ (left) - objective value agrees to machine precision, feasible
- $\gamma = \infty$ (right) - only requiring feasibility with no agreement at all
- tolerance is required only for constraint violation (zero here)
- compact and nicely scaled ($\log_{10}$ representation of entire range)

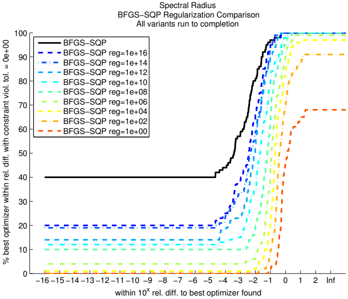
RMP of Regularizing Hessian in BFGS-SQP





RMP of Regularizing Hessian in BFGS-SQP





Pseudospectral radius test set
Lipschitz
$\sqrt{\varepsilon_{\text{mch}}}$ wins

Spectral radius test set
Not Lipschitz
unmodified BFGS-SQP wins

reg = 1: replace BFGS update by scaled identity (steepest descent)

BENCHMARKING EFFICIENCY VIA MULTIPLE β-RMPS

For solver $s \in \mathcal{S}$ on problem $p_i \in \mathcal{P}$, define:

$$t_i^s(j) := \text{cumulative cost to compute } \{x_0, \ldots, x_j\} \subseteq \{x_k\}_i^s$$

and for some given cost limit $t > 0$, the set of iterates encountered within that limit:

$$\mathcal{X}_i^s(t) := \begin{cases} \{x_k\}_i^s & \text{if } t = \infty \\ \{x_j : x_j \in \{x_k\}_i^s \text{ and } t_i^s(j) \leq t\} & \text{otherwise.} \end{cases}$$

and the redefinition of the best computed objective value now also subject to cost limit t:

$$f_i^s(\tau_v, t) := \min \{f_i(x) : \text{ s.t. } x \in \mathcal{X}_i^s(t), v_i(x) \leq \tau_v\}$$

BENCHMARKING EFFICIENCY VIA MULTIPLE β-RMPS

To assess progress with respect to cost, we will need to define a computational budget per problem:

$\mathcal{B} := \{b_i : b_i$ is max computational cost allowed for problem $p_i \in \mathcal{P}\}$.

We set b_i to cost of running BFGS-SQP on $p_i \in \mathcal{P}$. Alternatives:

- user-supplied budget values
- set b_i to average or median cost of solvers $s \in \mathcal{S}$ on $p_i \in \mathcal{P}$

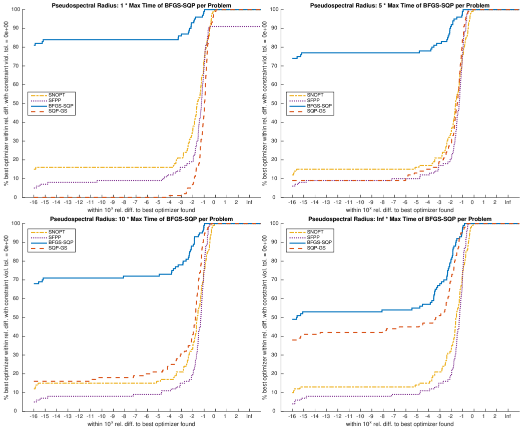
Then set target values

$$\omega_i := \min \{f_i^s(\tau_v, \beta b_i), s \in \mathcal{S}\}$$

Then the β-relative minimization profile curve $r^{s,\beta}_{\Omega,\tau_v} : \mathbb{R}^+ \to [0,1]$ for solver s is defined by:

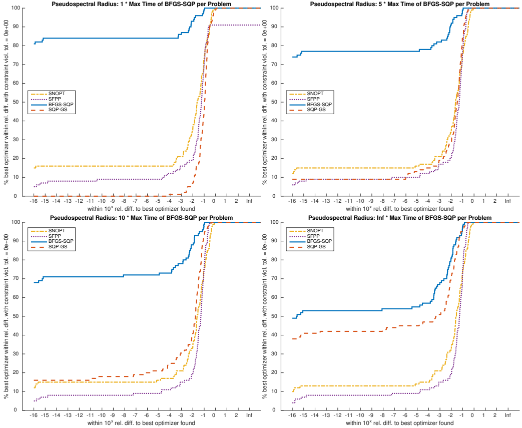
$$r^{s,\beta}_{\Omega,\tau_v}(\gamma) := \frac{1}{|\mathcal{P}|} \sum_{i=1}^{|\mathcal{P}|} \mathbb{1}_r \left(\omega_i, f_i^s(\tau_v, \beta b_i), \gamma\right).$$

β-RMPs: $\beta = \{1, 5, 10, \infty\}$, b_i = total CPU-time of BFGS-SQP on p_i



Pseudospectral Radius Test Set (Lipschitz)

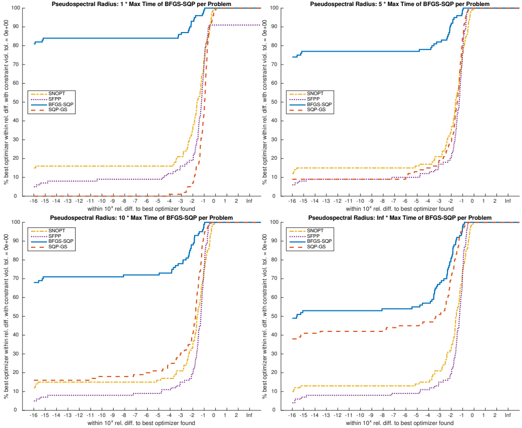
β-RMPs: $\beta = \{1, 5, 10, \infty\}$, b_i = TOTAL CPU-TIME OF BFGS-SQP ON p_i



Pseudospectral Radius Test Set (Lipschitz)

- $\beta = 1$ (top left): all solvers quit when BFGS-SQP finishes

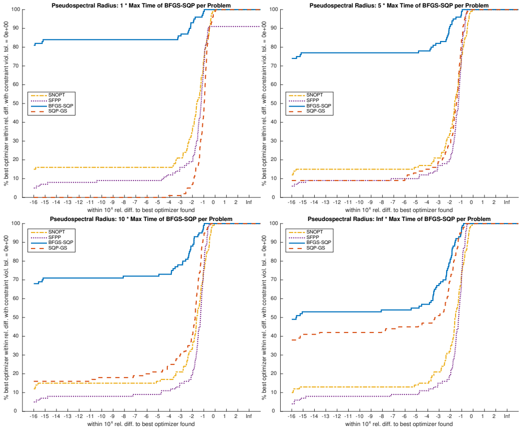
β-RMPs: $\beta = \{1, 5, 10, \infty\}$, b_i = total CPU-time of BFGS-SQP on p_i



Pseudospectral Radius Test Set (Lipschitz)

- $\beta = 1$ (top left): all solvers quit when BFGS-SQP finishes
- $\beta = 5$ (top right): all solvers get 5 times as much time as BFGS-SQP

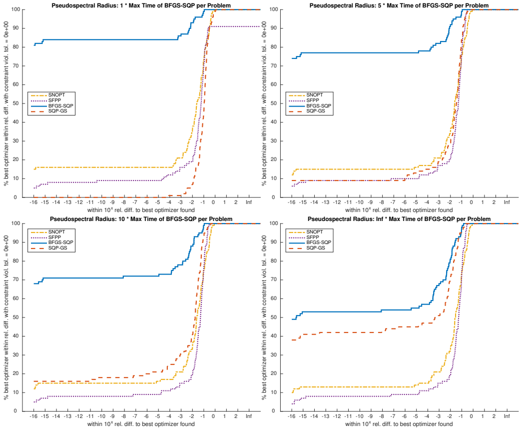
β-RMPS: $\beta = \{1, 5, 10, \infty\}$, b_i = TOTAL CPU-TIME OF BFGS-SQP ON p_i



Pseudospectral Radius Test Set (Lipschitz)

- $\beta = 1$ (top left): all solvers quit when BFGS-SQP finishes
- $\beta = 5$ (top right): all solvers get 5 times as much time as BFGS-SQP
- $\beta = 10$ (bottom left): all solvers get 10 times as much time as BFGS-SQP

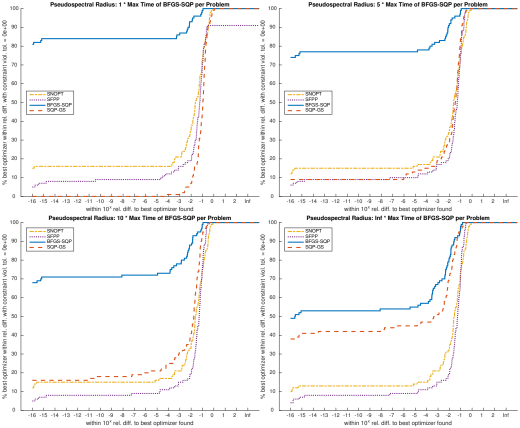
β-RMPs: $\beta = \{1, 5, 10, \infty\}$, b_i = total CPU-time of BFGS-SQP on p_i



Pseudospectral Radius Test Set (Lipschitz)

- $\beta = 1$ (top left): all solvers quit when BFGS-SQP finishes
- $\beta = 5$ (top right): all solvers get 5 times as much time as BFGS-SQP
- $\beta = 10$ (bottom left): all solvers get 10 times as much time as BFGS-SQP
- $\beta = \infty$ (bottom right): all solvers get unlimited time (max iters = 500)

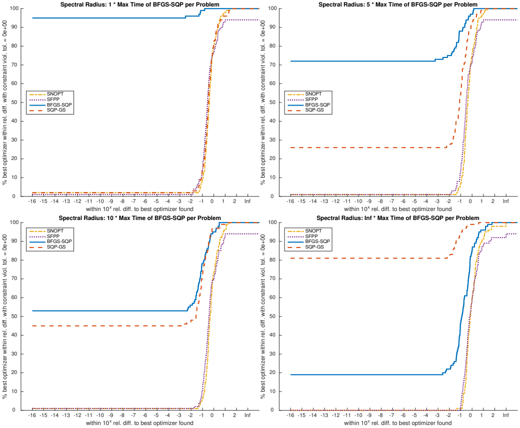
β-RMPs: $\beta = \{1, 5, 10, \infty\}$, b_i = TOTAL CPU-TIME OF BFGS-SQP ON p_i



Pseudospectral Radius Test Set (Lipschitz)

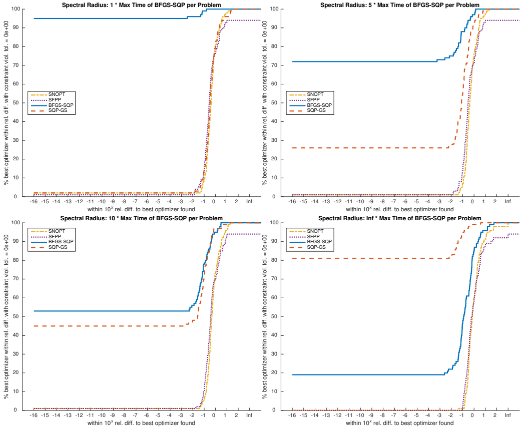
- $\beta = 1$ (top left): all solvers quit when BFGS-SQP finishes
- $\beta = 5$ (top right): all solvers get 5 times as much time as BFGS-SQP
- $\beta = 10$ (bottom left): all solvers get 10 times as much time as BFGS-SQP
- $\beta = \infty$ (bottom right): all solvers get unlimited time (max iters = 500)
- Even with no time limit, SQP-GS is behind BFGS-SQP

β-RMPs: $\beta = \{1, 5, 10, \infty\}$, b_i = total CPU-time of BFGS-SQP on p_i



Spectral Radius Test Set (Not Lipschitz)

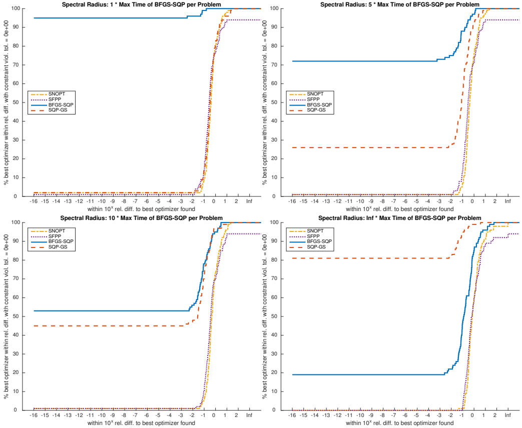
β-RMPS: $\beta = \{1, 5, 10, \infty\}$, b_i = TOTAL CPU-TIME OF BFGS-SQP ON p_i



Spectral Radius Test Set (Not Lipschitz)

- $\beta = 1$ (top left): all solvers quit when BFGS-SQP finishes

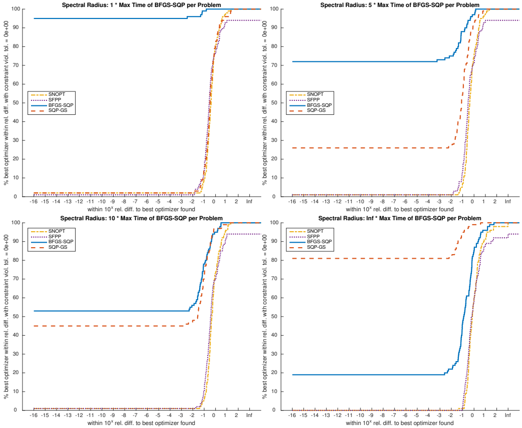
β-RMPs: $\beta = \{1, 5, 10, \infty\}$, b_i = total CPU-time of BFGS-SQP on p_i



Spectral Radius Test Set (Not Lipschitz)

- $\beta = 1$ (top left): all solvers quit when BFGS-SQP finishes
- $\beta = 5$ (top right): all solvers get 5 times as much time as BFGS-SQP

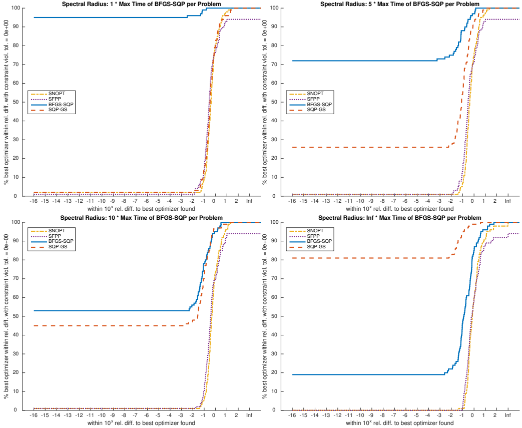
β-RMPS: $\beta = \{1, 5, 10, \infty\}$, b_i = TOTAL CPU-TIME OF BFGS-SQP ON p_i



Spectral Radius Test Set (Not Lipschitz)

- $\beta = 1$ (top left): all solvers quit when BFGS-SQP finishes
- $\beta = 5$ (top right): all solvers get 5 times as much time as BFGS-SQP
- $\beta = 10$ (bottom left): all solvers get 10 times as much time as BFGS-SQP

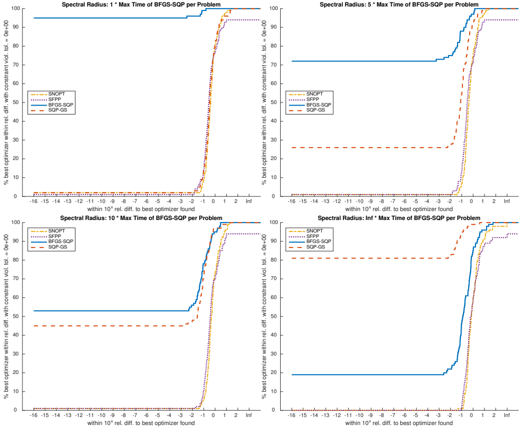
β-RMPs: $\beta = \{1, 5, 10, \infty\}$, b_i = TOTAL CPU-TIME OF BFGS-SQP ON p_i



Spectral Radius Test Set (Not Lipschitz)

- $\beta = 1$ (top left): all solvers quit when BFGS-SQP finishes
- $\beta = 5$ (top right): all solvers get 5 times as much time as BFGS-SQP
- $\beta = 10$ (bottom left): all solvers get 10 times as much time as BFGS-SQP
- $\beta = \infty$ (bottom right): all solvers get unlimited time (max iters = 500)

β-RMPs: $\beta = \{1, 5, 10, \infty\}$, b_i = total CPU-time of BFGS-SQP on p_i



Spectral Radius Test Set (Not Lipschitz)

- $\beta = 1$ (top left): all solvers quit when BFGS-SQP finishes
- $\beta = 5$ (top right): all solvers get 5 times as much time as BFGS-SQP
- $\beta = 10$ (bottom left): all solvers get 10 times as much time as BFGS-SQP
- $\beta = \infty$ (bottom right): all solvers get unlimited time (max iters = 500)
- SQP-GS pulls ahead when time limit is removed (ironic, as not Lipschitz)

β-RMPs in Practice

Requires:

- history of iterates:
  - $\{f_i\}$, $\{v_i\}$
  - cumulative cost to compute each i^{th} iterate
    - possibly obtained with OOP if solver doesn't provide
    - can be estimated via average if not attainable (this talk)
- user-selected violation tolerance and a handful of pertinent β values
- test solvers should be run with tight tolerances to maximize amount of data collected - β-RMPs simulate different stopping criteria

Optional (can be automatically generated from experimental data):

- target values $\omega_i \in \Omega$
- budget values $b_i \in \mathcal{B}$

Does not require:

- success/failure criterion

We applied RMPs to nonsmooth, nonconvex, constrained optimization but they can be used in a much broader context.

ACKNOWLEDGMENT

Thanks to Philip Gill and Elizabeth Wong for making (at short notice!) an improved version of SNOPT's Matlab interface available so that the iteration history needed to create β-RMPs could be collected.

y

Muchas Gracias a Todos Ustedes!

Last slide: another birthday meeting. . .

CELEBRATING ANDREW CONN'S 70TH BIRTHDAY

Workshop on Nonlinear Optimization Algorithms and Industrial Applications

June 2 – 4, 2016

The Fields Institute for Research in Mathematical Sciences
Toronto

Organizers:
Michael Overton, NYU
Oleksandr Romanko, IBM Canada
Tamás Terlaky, Lehigh University
Henry Wolkowicz, University of Waterloo

No speaking slots left, but plenty of room in a poster session. Everyone is welcome to attend. Some funding is available for students and early career researchers presenting posters.

REFERENCES

F. CURTIS, T. MITCHELL, AND M. OVERTON, *A BFGS-SQP method for nonsmooth, nonconvex, constrained optimization and its evaluation using relative minimization profiles*, Revised version submitted to Optimization Methods and Software, Dec 31, 2015, (2016).

F. CURTIS AND M. OVERTON, *A sequential quadratic programming algorithm for nonconvex, nonsmooth constrained optimization*, SIAM J. Optim., 22 (2012), pp. 474–500.

E. DOLAN AND J. MORÉ, *Benchmarking optimization software with performance profiles*, Math. Program., 91 (2002), pp. 201–213.

A. LEWIS AND M. OVERTON, *Nonsmooth optimization via quasi-Newton methods*, Math. Program., 141 (2013), pp. 135–163.

J. MORÉ AND S. WILD, *Benchmarking derivative-free optimization algorithms*, SIAM J. Optim., 20 (2009), pp. 172–191.