Nonconvex Optimization: Opportunities and Challenges

Frank E. Curtis, Lehigh University

presented at

East Coast Optimization Meeting
George Mason University

Fairfax, Virginia

April 2, 2021

Outline

Introduction

Local vs. Global Search

Worst-case Complexity

Numerical Comparisons

Conclusion

Outline

Introduction

Local vs. Global Search

Worst-case Complexity

Numerical Comparisons

Conclusion

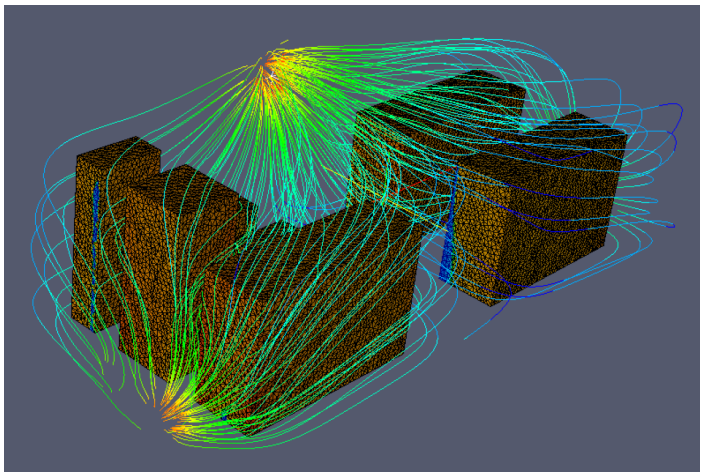
Optimization

Consider optimization problems of the form

$$\min_{x \in \mathcal{X}} \; f(x).$$

Problems arise throughout science and engineering:

- statistics, e.g., model fitting
- control, e.g., optimal trajectory
- operations research, e.g., minimizing cost
- economics, e.g., maximizing utility
- geophysics, e.g., seismic inversion



Optimization algorithms

What do we want from optimization algorithms?

Practitioners:

- reliable
- fast
- easy-to-use/write software

Algorithm designers and theorists:

- convergence guarantees
- convergence rate guarantees
- "simplicity"

Convex vs. nonconvex optimization

No one is an expert in all types of optimization problems/algorithms.

- continuous vs. discrete
- unconstrained vs. constrained
- nonlinear vs. linear
- deterministic vs. stochastic
- nonconvex vs. convex
- smooth vs. nonsmooth
- finite- vs. infinite-dimensional

In this talk, I focus on the classes above (in blue), where:

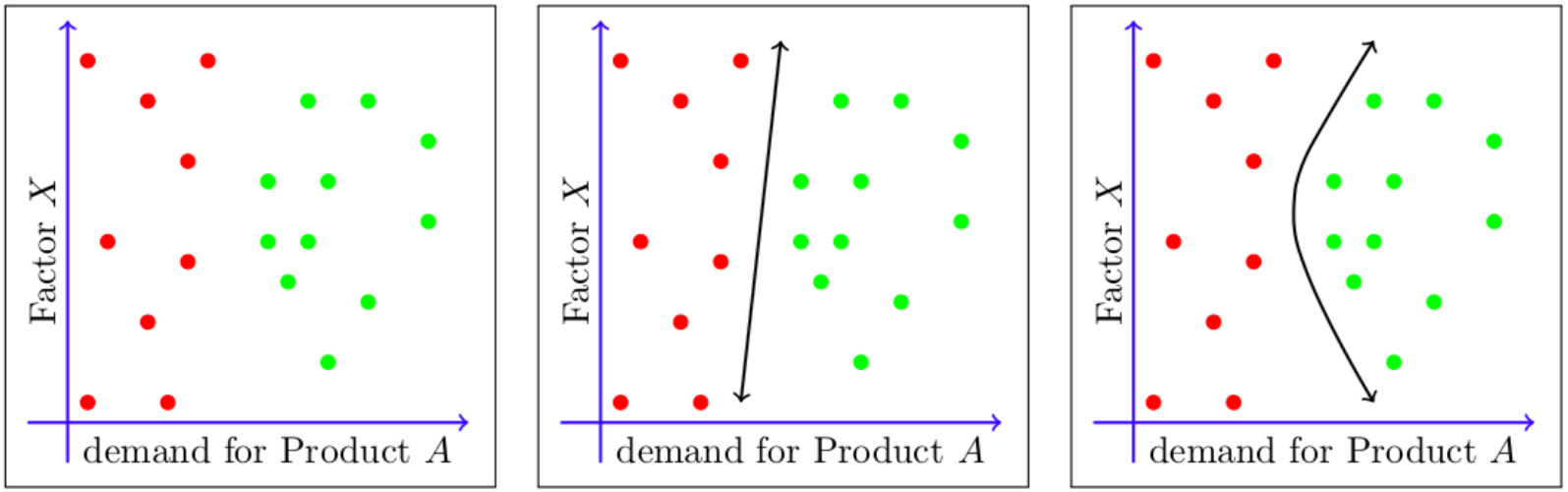
> *"Key distinction is not linear vs. nonlinear, but convex vs. nonconvex."*
> *- Rockafellar (paraphrased)*

I'll also touch on issues related to stochastic algorithms.

Convex optimization: support vector machines

A typical example of a convex optimization problem:

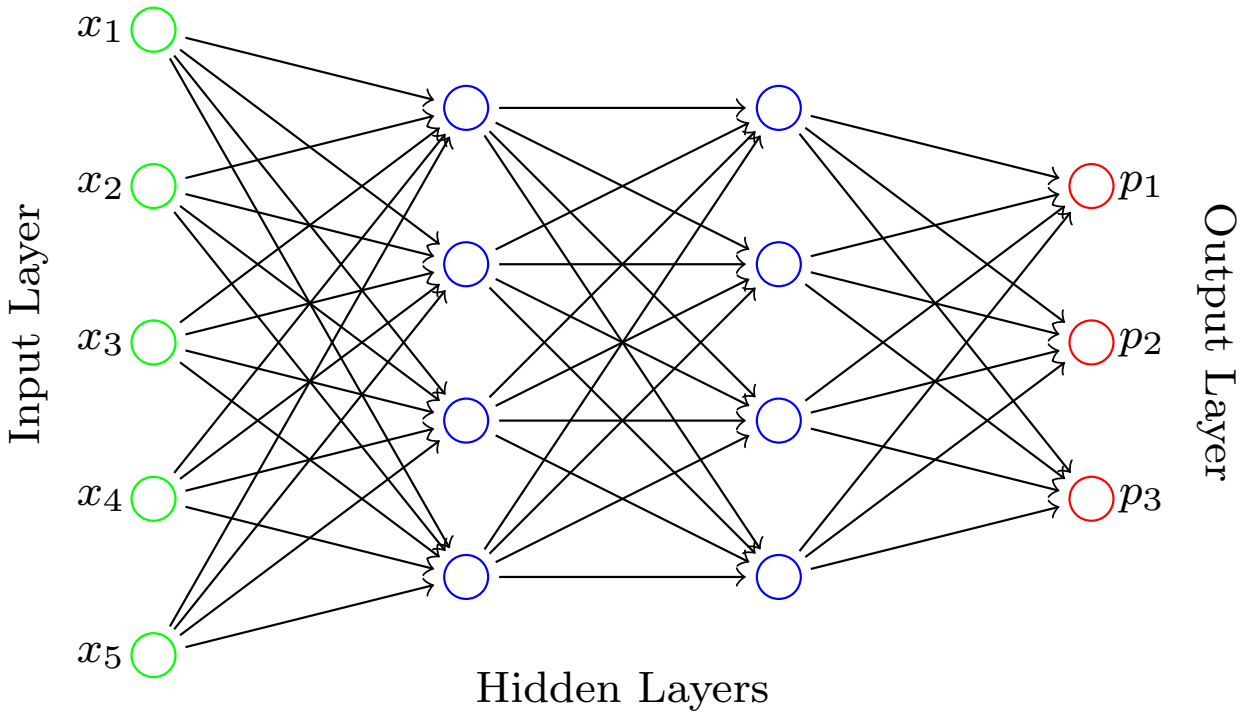
- finding a *support vector machine* classifier
- classically linear, but can find nonlinear classifiers with kernels



Nonconvex optimization: deep neural networks

Prevalent example of a nonconvex optimization problem:

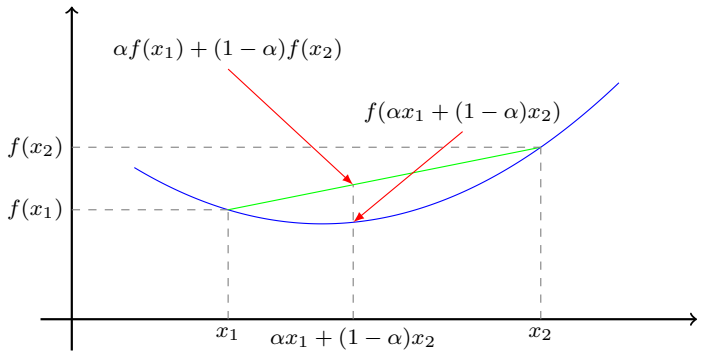
- training a *deep neural network*
- often nonsmooth as well (but this is swept under the rug)

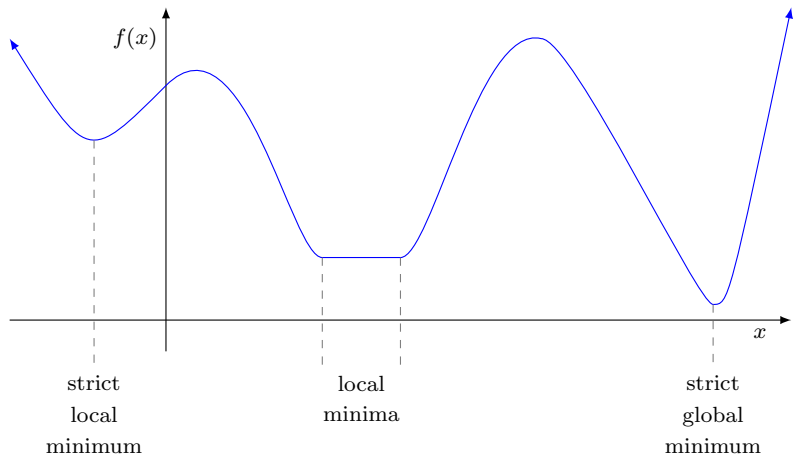


$$h(w;x) = a_l(W_l \dots (a_2(W_2(a_1(W_1 x + \omega_1)) + \omega_2)) \dots)$$

Convex vs. nonconvex optimization

What are the key differences?





Advantages in convex optimization:

- affine (global) underestimators
- strong duality
- local minimizers are global minimizers

Outline

Introduction

Local vs. Global Search

Worst-case Complexity

Numerical Comparisons

Conclusion

Local vs. global minimizers

Even 10 years ago, common sentiment among many continuous optimizers:

> *If your (continuous) problem is nonconvex, then it's a bad formulation.*

A related sentiment, also felt by some optimizers:

> *If your problem is nonconvex and your algorithm is not guaranteed to find a global minimizer, then it is not a worthwhile algorithm.*

On the contrary, there are worthwhile problems that are nonconvex, and algorithms that do not find global minimizers are worthwhile.

Example: (deep) learning

"Gradient Descent Converges to Minimizers"

- Lee, Simchowitz, Jordan, and Recht

"Gradient Descent Only Converges to Minimizers"

- Panageas and Piliouras

"Gradient Descent Finds Global Minima of Deep Neural Networks"

- Du, Lee, Li, Wang, and Zhai

"SGD Converges to Global Minimum in Deep Learning via Star-Convex Path"

- Zhou, Yang, Zhang, Liang, and Tarokh

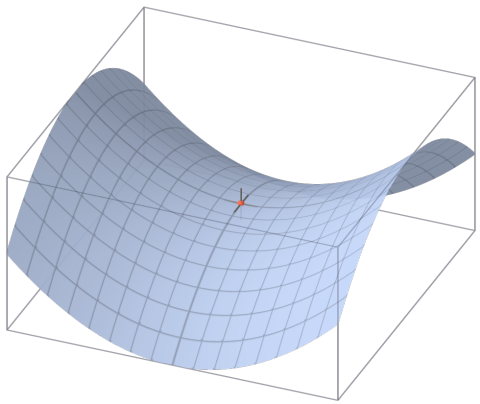
"On the Global Convergence of Gradient Descent for Over-parameterized Models using Optimal Transport"

- Chizat and Bach

. . . amongst many others

Key ideas:

- random initialization and/or
- random perturbations to directions



Example: ℓ_0-norm minimization / complementarity constraints

"Complementarity Formulations of ℓ_0-norm Optimization Problems"

- Feng, Mitchell, Pang, Shen, and Waechter

"ℓ_0-norm Minimization for Basis Selection"

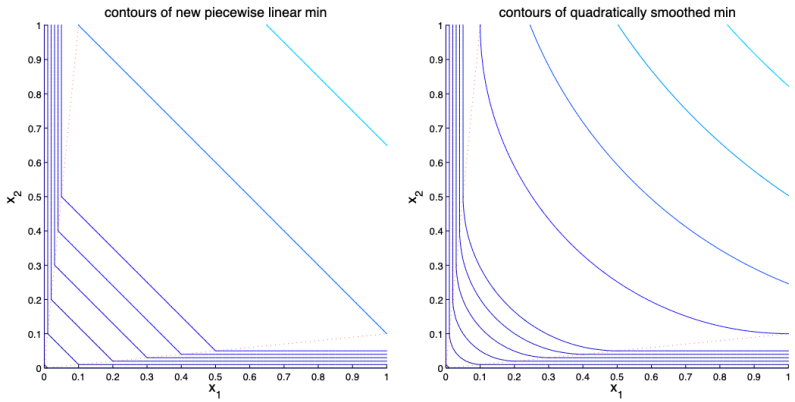
- Wipf and Rao

"Solving mathematical program with complementarity constraints as nonlinear programs"

- Fletcher and Leyffer

"An interior point method for mathematical programs with complementarity constraints (MPCCs)"

- Raghunathan and Biegler



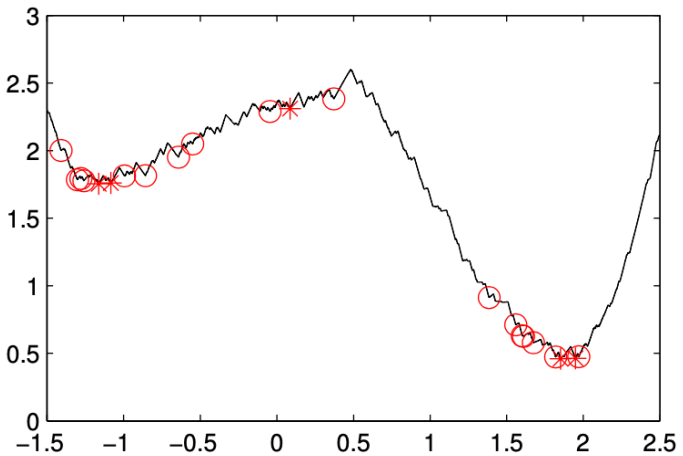
Example: chance-constrained optimization

Optimization problem with chance/probabilistic constraints:

$$\min_{x \in \mathbb{R}^n} \; f(x) \quad \text{s.t.} \quad \mathbb{P}[c(x, \xi) \leq 0] \geq 1 - \alpha$$

Deterministic approximation through cardinality constraints:

$$\min_{x \in \mathbb{R}^n} \; f(x) \quad \text{s.t.} \quad |\{i \in \mathcal{I} : c(x, \xi_i) \leq 0\}| \geq N - M$$



"A Sequential Algorithm for Solving Nonlinear Optimization Problems with Chance Constraints"

- Curtis, Waechter, and Zavala

Take-home messages

Two messages, in my opinion:

(1) Nonconvexity cannot always be avoided. And that's OK!

(2) Local search methods are useful, despite no global solution guarantee.

The questions then become:

(a) What kinds of algorithms are worth designing?

(b) How should we compare algorithm performance?

My overall message in the remainder of the talk:

Continuous optimization has become too theoretical in recent years!

- too much emphasis on theoretical performance guarantees;
- not enough emphasis on practical performance;
- typical analyses and numerical experiments are biasing optimizers toward certain algorithms, and we should snap out of it
- "gone down a bit of a rathole here" — S.J.W.

Outline

Introduction

Local vs. Global Search

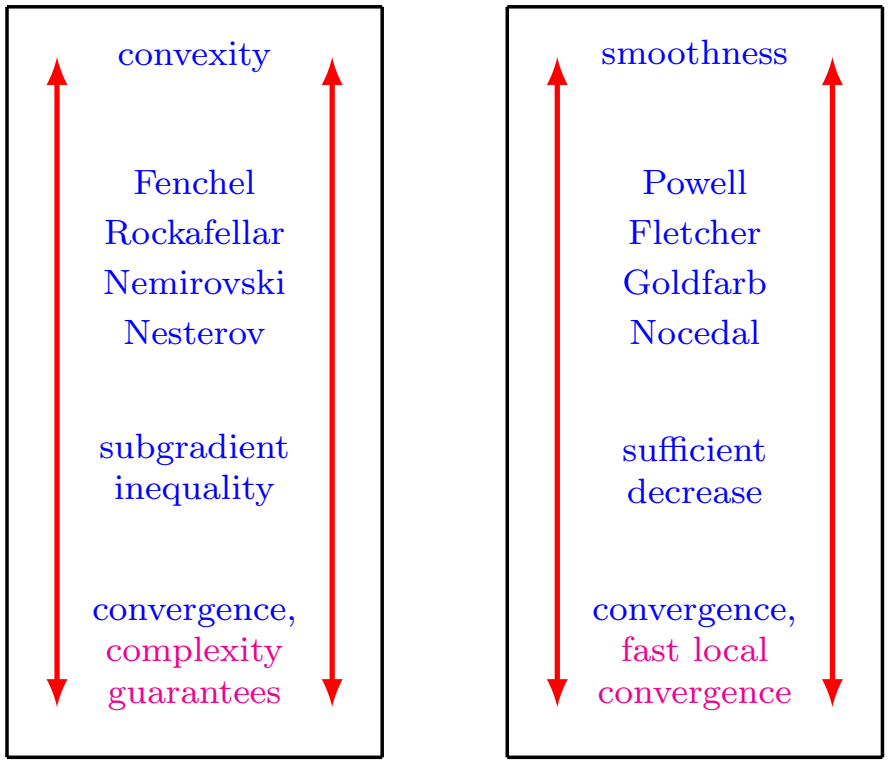
Worst-case Complexity

Numerical Comparisons

Conclusion

History

Nonlinear continuous optimization has seen parallel developments



These worlds have (finally) collided! Has it been beneficial?

Convex setting

Consider the problem

$$\min_{x \in \mathbb{R}^n} \; f(x),$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is

- convex and
- L-smooth, i.e., ∇f is Lipschitz with constant L.

Gradient descent employs the iteration

$$x_{k+1} \leftarrow x_k - \frac{1}{L} \nabla f(x_k).$$

It is well known that this algorithm yields

$$f(x_k) - f(x_*) \leq \frac{1}{2k}(L\|x_0 - x_*\|_2^2).$$

Nesterov acceleration

Famously, Nesterov proved that gradient descent is not optimal

- ... over iterative algorithms that impose

$$x_k \in x_0 + \text{span}\{\nabla f(x_0), \ldots, \nabla f(x_{k-1})\}.$$

- Key insight is that of generating estimate sequences.

Theoretical benefit can be seen in that

$$f(x_k) - f(x_*) \leq \frac{1}{2k}(L\|x_0 - x_*\|_2^2) \qquad \text{(gradient method)}$$

$$f(x_k) - f(x_*) \leq \frac{4}{(k+2)^2}(L\|x_0 - x_*\|_2^2) \qquad \text{(optimal method)}$$

This theoretical bound is tight.

Practical performance

In convex optimization...

practical performance is also often better for "better complexity" method.

$$\min_{x \in \mathbb{R}^n} \log \left(\sum_{i=1}^{m} \exp(a_i^T x + b_i) \right)$$

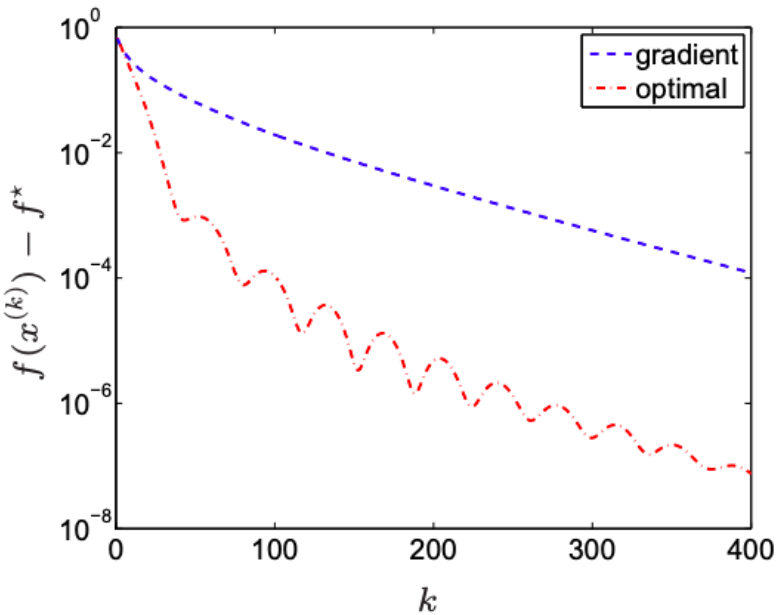


Image courtesy of Maryam Fazel (lecture notes)

Nonconvex setting

What about nonconvex settings?

Issue #1: Unreasonable to consider suboptimality or distance to a solution, i.e.,

$$f(x_k) - f(x_*) \quad \text{or} \quad \|x_k - x_*\|$$

Instead, consider approximate stationarity conditions, e.g.

$$\|\nabla f(x_k)\|_2 \leq \epsilon$$

or

$$\|\nabla f(x_k)\|_2 \leq \epsilon_g \quad \text{and} \quad \min(\text{eig}(\nabla^2 f(x_k))) \geq -\epsilon_H$$

Aside: Who's to say these are appropriate?

- Neither offers a guarantee for suboptimality or distance to a solution
- We are choosing the condition to benefit our algorithms!

Gradient descent

Issue #2: Typical analysis is *extremely* conservative.

Consider gradient descent with

$$x_{k+1} \leftarrow x_k - \frac{1}{L}\nabla f(x_k).$$

We are interested in the set

$$\mathcal{G}(\epsilon_g) := \{x \in \mathbb{R}^n : \|\nabla f(x)\|_2 \leq \epsilon_g\},$$

in particular, we aim to bound the cardinality of

$$\mathcal{K}_g(\epsilon_g) := \{k \in \mathbb{N} : x_k \notin \mathcal{G}(\epsilon_g)\}.$$

Upper bound on $|\mathcal{K}_g(\epsilon_g)|$

Using $s_k = -\frac{1}{L}\nabla f(x_k)$ and the upper bound

$$f_{k+1} \leq f_k + \nabla f(x_k)^T s_k + \frac{1}{2} L \|s_k\|_2^2,$$

one finds for $x_k \notin \mathcal{G}(\epsilon_g)$ that

$$\begin{aligned} & f_k - f_{k+1} \geq \frac{1}{2L}\|\nabla f(x_k)\|_2^2 \\ \Longrightarrow \quad & (f_0 - f_{\inf}) \geq \frac{1}{2L}|\mathcal{K}_g(\epsilon_g)|\epsilon_g^2 \\ \Longrightarrow \quad & |\mathcal{K}_g(\epsilon_g)| \leq \frac{2L(f_0 - f_{\inf})}{\epsilon_g^2}. \end{aligned}$$

Hence, gradient descent yields ϵ_g-approximate first-order stationarity in

$$\mathcal{O}(\epsilon_g^{-2}) \text{ iterations.}$$

"Nice" f

But what if f is "nice"?

...e.g., satisfying the Polyak-Lojasiewicz condition (for $c \in \mathbb{R}_{>0}$), i.e.,

$$f(x) - f_* \leq \frac{1}{2c}\|\nabla f(x)\|_2^2 \quad \text{for all} \quad x \in \mathbb{R}^n.$$

Now consider the set

$$\mathcal{F}(\epsilon_f) := \{x \in \mathbb{R}^n : f(x) - f_* \leq \epsilon_f\}$$

and consider an upper bound on the cardinality of

$$\mathcal{K}_f(\epsilon_f) := \{k \in \mathbb{N} : x_k \notin \mathcal{F}(\epsilon_f)\}.$$

Upper bound on $|\mathcal{K}_f(\epsilon_f)|$

Using $s_k = -\frac{1}{L}\nabla f(x_k)$ and the upper bound

$$f_{k+1} \leq f_k + \nabla f(x_k)^T s_k + \frac{1}{2} L \|s_k\|_2^2,$$

one finds for $x_k \notin \mathcal{F}(\epsilon_f)$ that

$$\begin{aligned}
& f_k - f_{k+1} \geq \frac{1}{2L}\|\nabla f(x_k)\|_2^2 \geq \frac{c}{L}(f_k - f_*) \\
\implies \quad & \left(1 - \frac{c}{L}\right)(f_k - f_*) \geq f_{k+1} - f_* \\
\implies \quad & \left(1 - \frac{c}{L}\right)^k (f_0 - f_*) \geq f_k - f_* \\
\implies \quad & |\mathcal{K}_f(\epsilon_f)| \leq \log\left(\frac{f_0 - f_*}{\epsilon_f}\right)\left(\log\left(\frac{L}{L-c}\right)\right)^{-1}.
\end{aligned}$$

Hence, gradient descent yields ϵ_f-approximate optimality in

$$\mathcal{O}\left(\log\left(\frac{1}{\epsilon_f}\right)\right) \text{ iterations.}$$

What's the difference?

In the "general nonconvex" analysis...

...the expected decrease for the first step is much more pessimistic:

$$\text{general nonconvex:} \qquad f_0 - f_1 \geq \frac{1}{2L}\epsilon_g^2$$

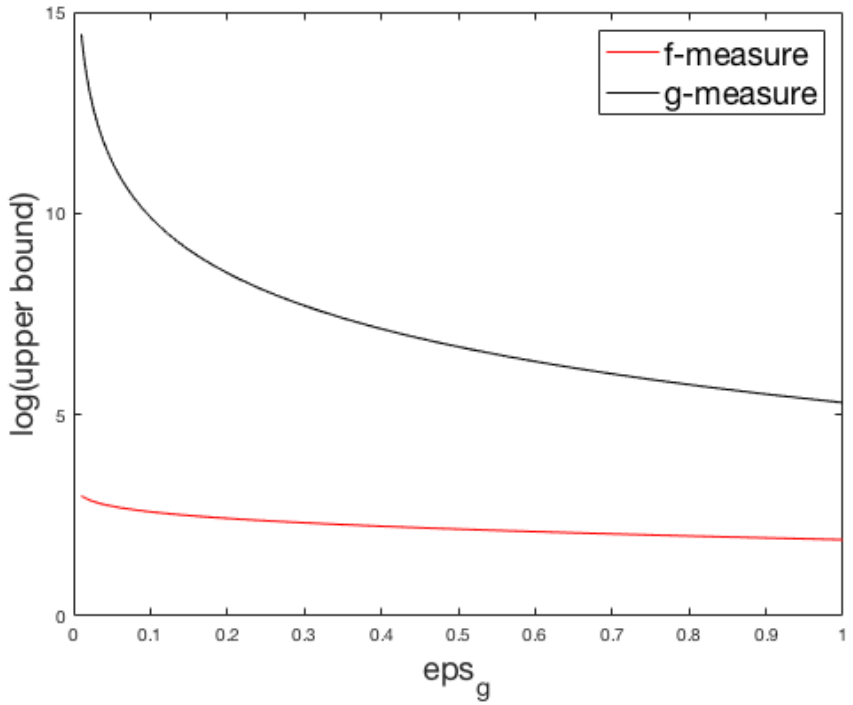
$$\text{PL condition:} \qquad \left(1 - \frac{c}{L}\right)(f_0 - f_*) \geq f_1 - f_*$$

...and it remains more pessimistic throughout!

Upper bounds on $|\mathcal{K}_f(\epsilon_f)|$ versus $|\mathcal{K}_g(\epsilon_g)|$

Let $f(x) = \frac{1}{2}x^2$, meaning that $\nabla f(x) = x$.

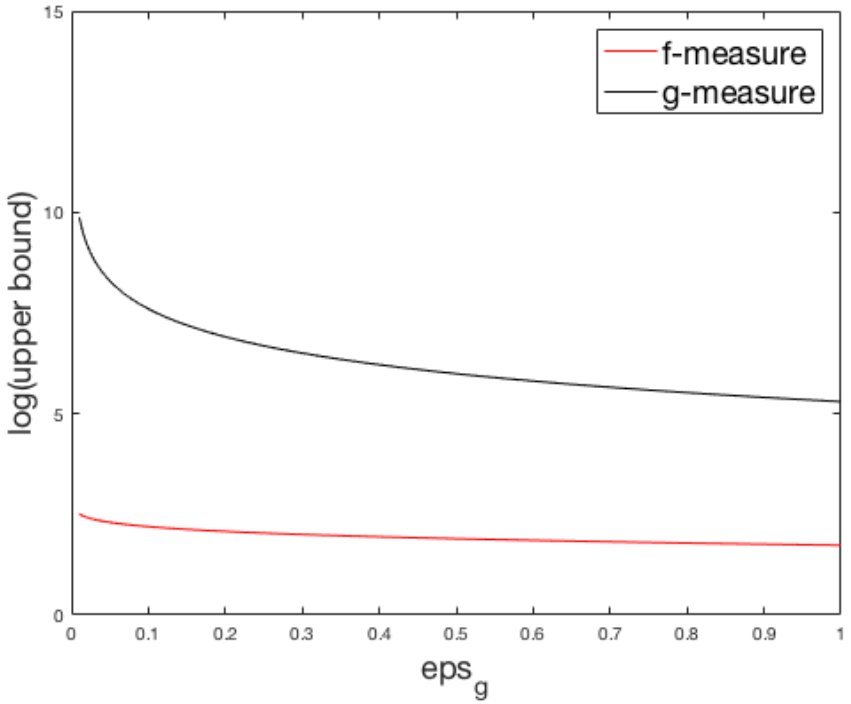
- Let $\epsilon_f = \frac{1}{2}\epsilon_g^2$, meaning that $\mathcal{F}(\epsilon_f) = \mathcal{G}(\epsilon_g)$.
- Let $x_0 = 10$, $c = 1$, and $L = 2$. (Similar pictures for any $L > 1$.)



Upper bounds on $|\mathcal{K}_f(\epsilon_f)|$ versus $|\{k \in \mathbb{N} : \frac{1}{2}\|\nabla f(x_k)\|_2^2 > \epsilon_g\}|$

Let $f(x) = \frac{1}{2}x^2$, meaning that $\frac{1}{2}\nabla f(x)^2 = \frac{1}{2}x^2$.

- Let $\epsilon_f = \epsilon_g$, meaning that $\mathcal{F}(\epsilon_f) = \mathcal{G}(\epsilon_g)$.
- Let $x_0 = 10$, $c = 1$, and $L = 2$. (Similar pictures for any $L > 1$.)



Tight bounds

Ok, but are the bounds tight? Yes!

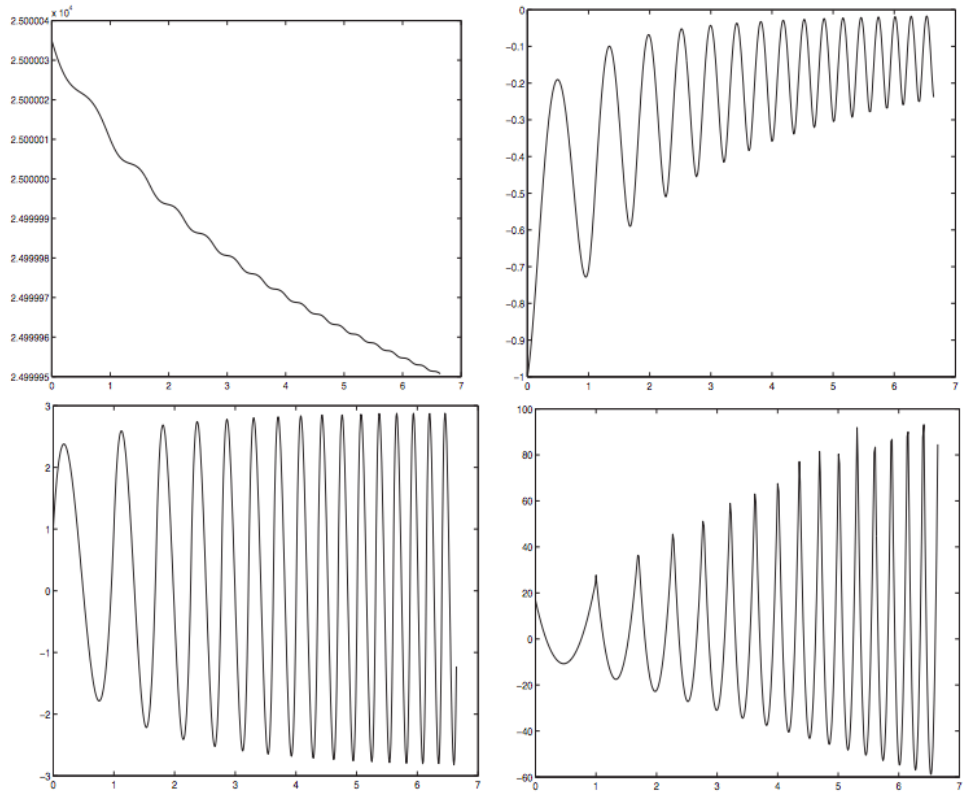


FIG. 2.1. *The function $f^{(1)}$ (top left) and its derivatives of order one (top right), two (bottom left), and three (bottom right) on the first* 16 *intervals.*

Cartis, Gould, Toint (2010)

Oracle complexity

It's not only first-order methods.

- Second-order methods finding approximate first-order stationarity:

$$\text{optimal complexity:} \quad \mathcal{O}\left(\epsilon^{-3/2}\right)$$

- pth-order methods finding approximate first-order stationarity:

$$\text{optimal complexity:} \quad \mathcal{O}\left(\epsilon^{-(p+1)/p}\right)$$

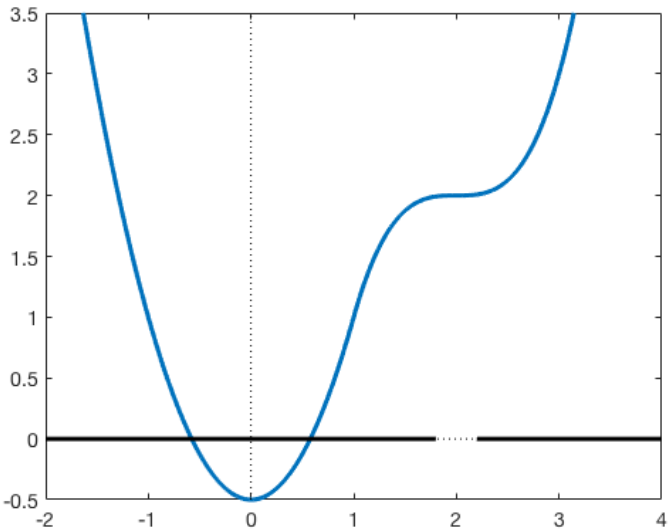
These bounds are tight.

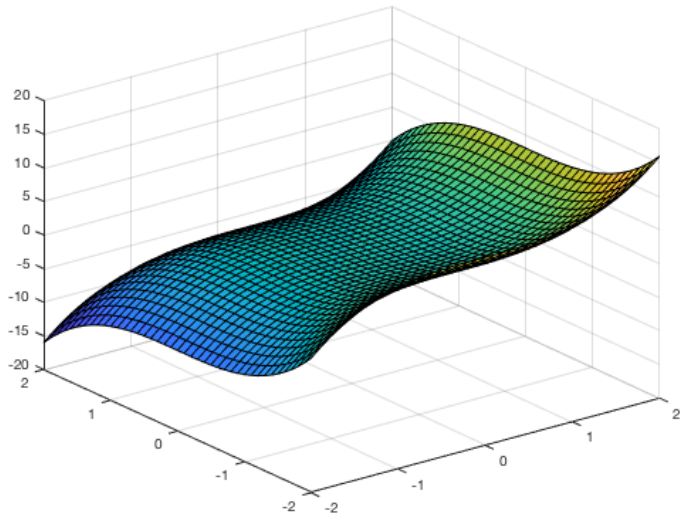
- Carmon, Duchi, Hinder, Sidford (2019)

Take-home message #1

Nonconvex functions, even pth order smooth ones, are too diverse.

Better to consider subclasses of problems, or analyze performance *regionally*:





"Regional complexity analysis of algorithms for nonconvex smooth optimization"

- Curtis and Robinson (2020)

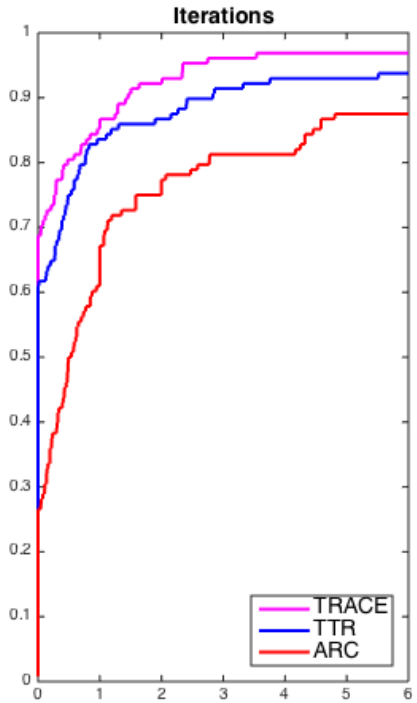
Take-home message #2

"Better complexity" has yet to mean "better performance" for nonconvex!

They say:

> *"Newton's method is as slow as gradient descent."*

This essentially ignores reality.



"A trust region algorithm with a worst-case iteration complexity of $\mathcal{O}(\epsilon^{-3/2})$ for nonconvex optimization"

- Curtis, Robinson, Samadi (2017)

Outline

Introduction

Local vs. Global Search

Worst-case Complexity

Numerical Comparisons

Conclusion

Performance profiles

A positive development in algorithm comparisons; Dolan and Moré (2002)

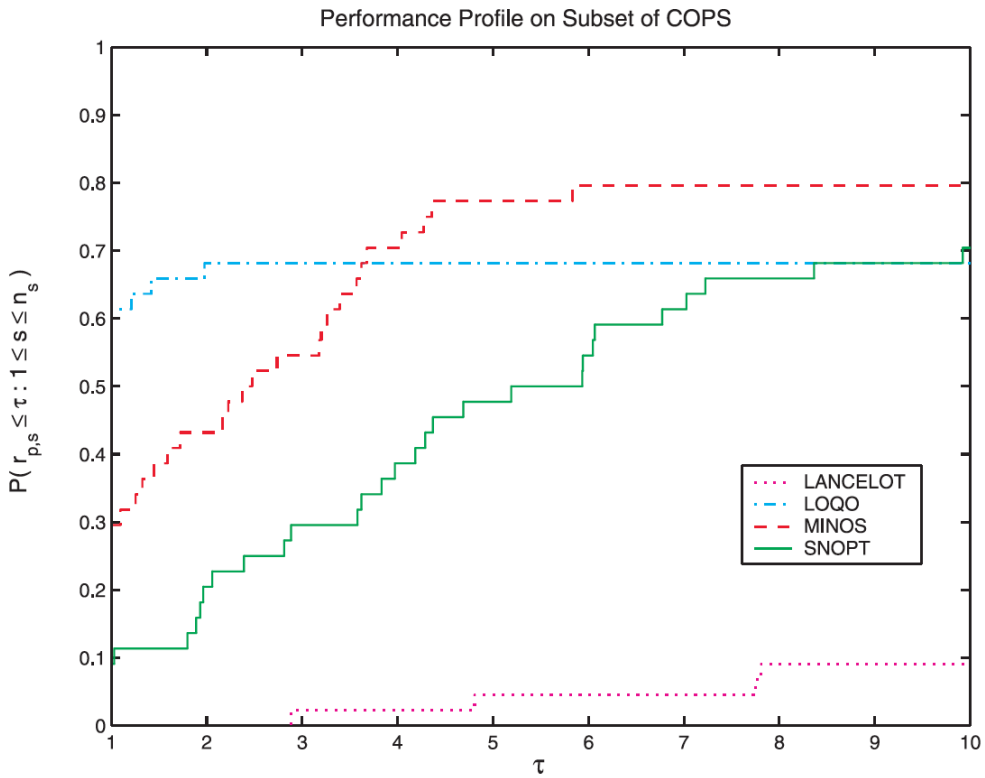


Fig. 1. Performance profile on [0, 10]

Not the first idea, but it became the "gold standard" approach.

Advantages and disadvantages

Advantages

- one visual
- encapsulates performance over large test set
- shows both speed and reliability

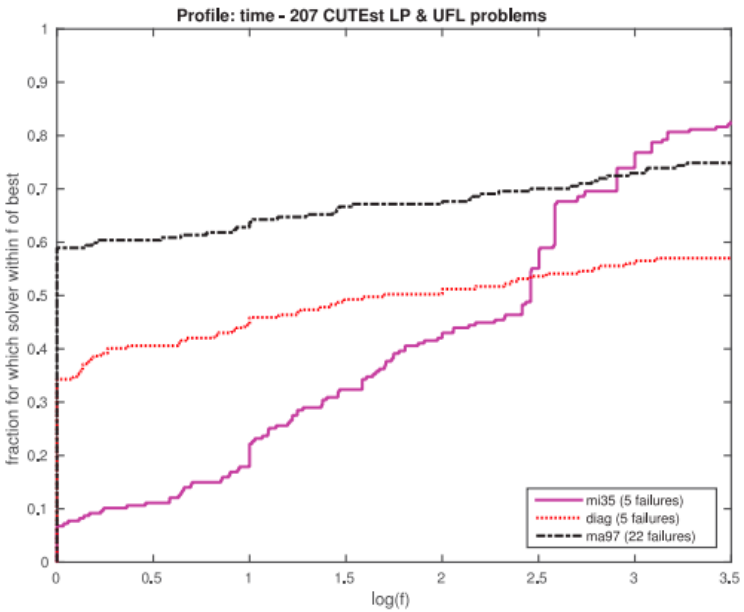
Disadvantages

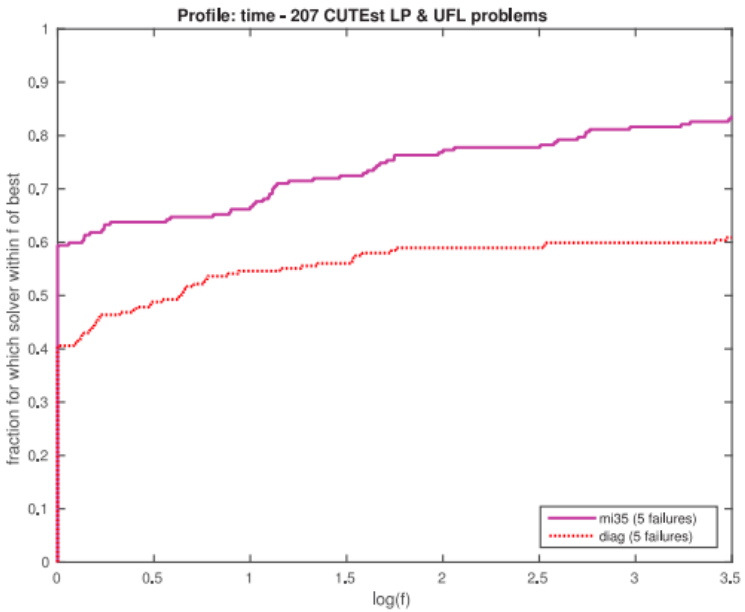
- selection of solvers can skew results
- even a large test set might not be enough
- do solvers (e.g., default parameters) become biased?

But there is a HUGE positive overall...

Proceed with caution

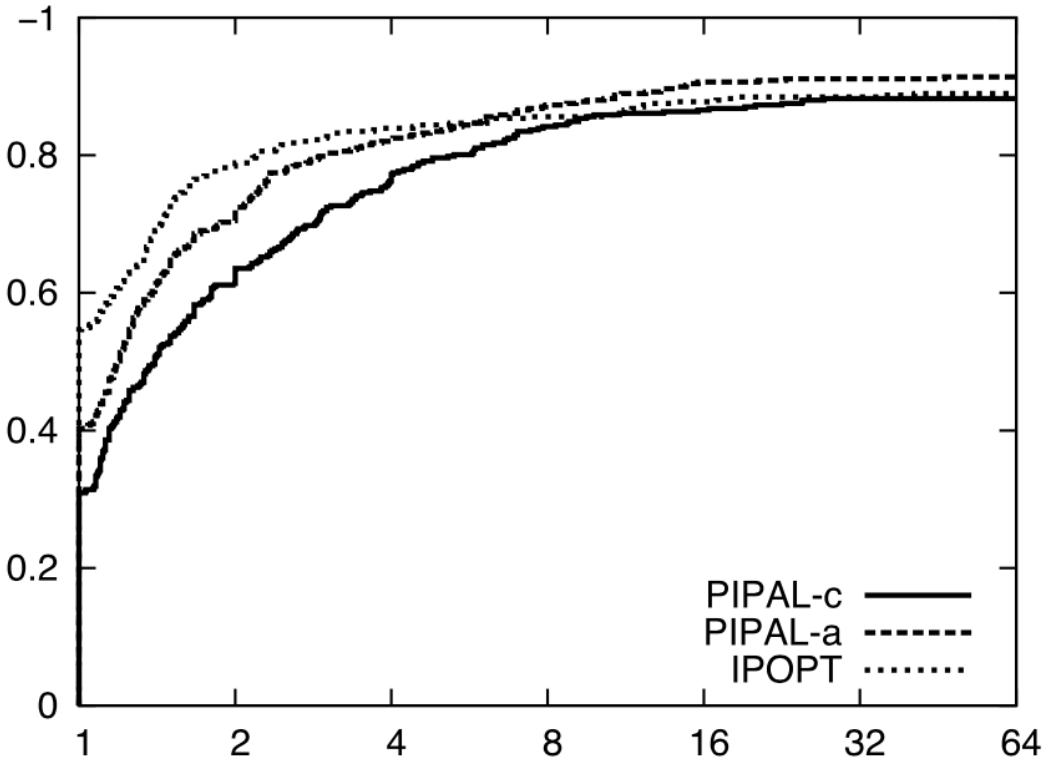
Selection of solvers can skew results; Gould and Scott (2016)





Comparisons with "state-of-the-art"

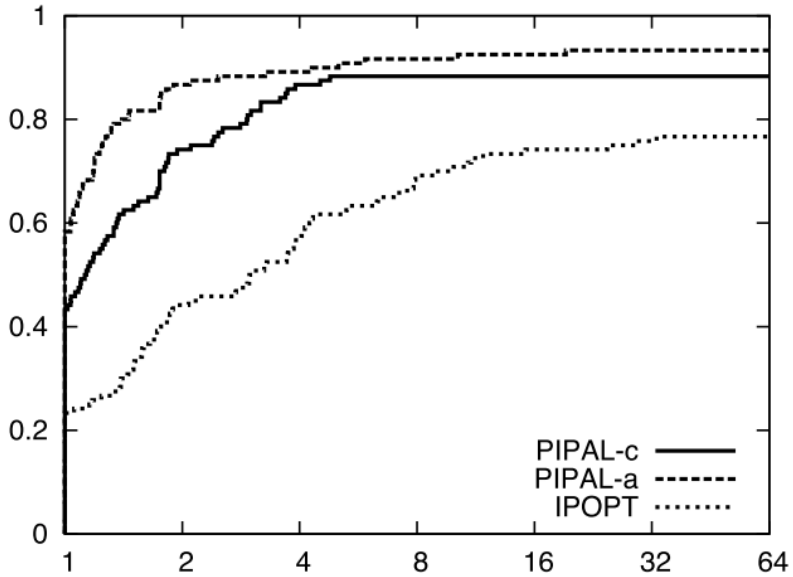
Hard to beat a highly tuned state-of-the-art solver! Curtis (2012)

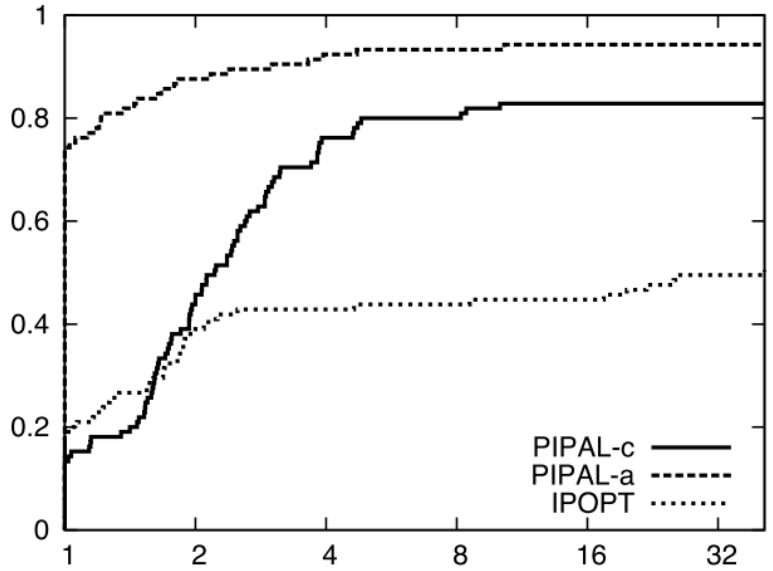


I've seen many papers rejected for this reason.

Comparisons with "state-of-the-art"

But if you change the test set, it's a different picture! Curtis (2012)





We should not let one test set (or a few) bias all research.

Overall positive

The overall positive is that...

- people have thought hard about algorithm comparisons.

Besides performance profiles:

- operational characteristics; Strongin and Sergeyev (2000)
- data profiles; Moré and Wild (2009)
- relative minimization profiles; Curtis, Mitchell, Overton (2017)

Stochastic optimization

There needs to be more work along these lines for stochastic optimization.

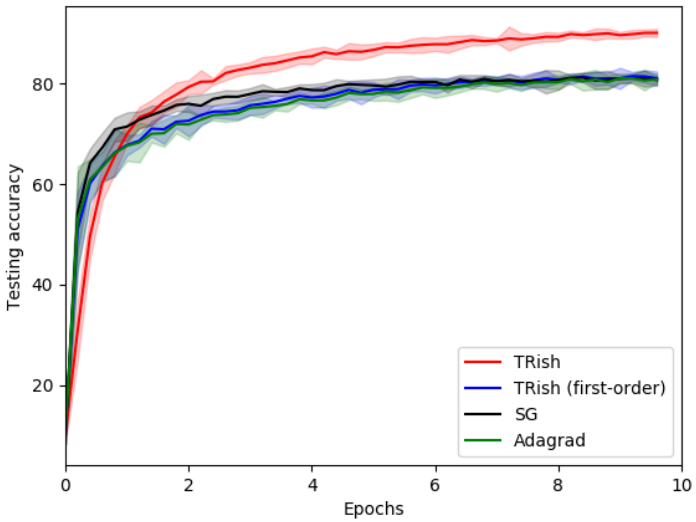
I know there has been some, but these have not yet been widely adopted.

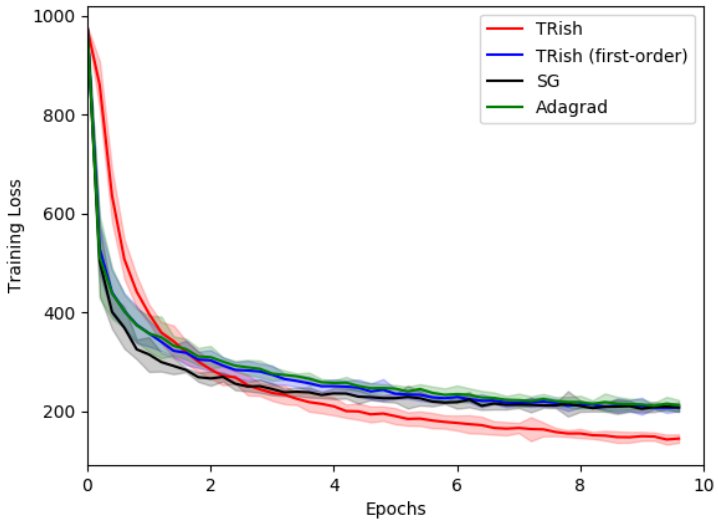
It is even more important than in the deterministic setting:

- running a solver on a problem once (or a few times) is not enough
- algorithm variations even more plentiful

Typical experiment

The results of a typical type of experiment:





Ok, but what about:

- other objectives? other networks? other datasets? other # of epochs?
- other hyperparameter settings? other algorithms?

Only using the same handful of problems, all research is biased toward them.

Fair comparisons

My goal is not to advocate for one approach for comparisons.

Take-home message:

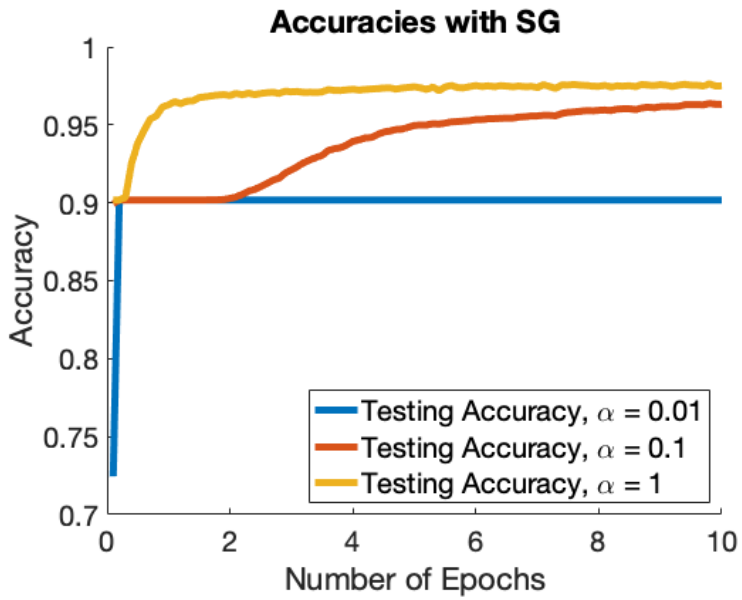
> *The only way for algorithm comparisons to be fair would be for them to* ***include all computational time spent tuning each algorithm.***

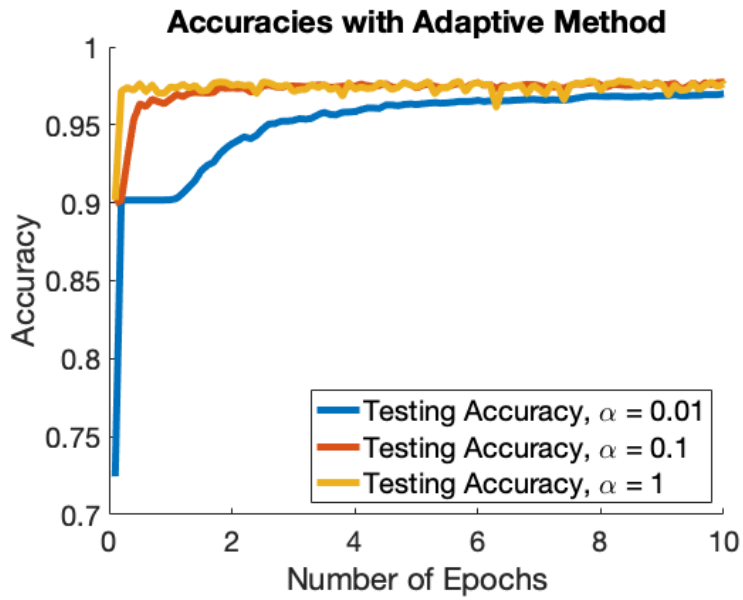
Asi and Duchi (2019):

- Training a neural network for a single task
- 750,000 CPU days of computation
- $\approx$ energy eqiuvalent to drive 4,000 Toyota Camrys 380 miles from SF to LA.

Adaptive stochastic optimization

I advocate for adaptive (second-order) algorithms for stochastic optimization.





Outline

Introduction

Local vs. Global Search

Worst-case Complexity

Numerical Comparisons

Conclusion

Summary

Take-home messages:

- Nonconvexity cannot always be avoided. And that's OK!
- Local search methods are useful.
- Continuous optimization has become too theoretical.
- For analysis, "nonconvex" problems are too diverse.
- "Better complexity" $\neq$ "better performance" for nonconvex optimization.
- Need to think hard about empirical comparisions
- . . . especially in the stochastic optimization setting.