Complexity analysis of second-order algorithms based on line search for smooth nonconvex optimization

Clément Royer - University of Wisconsin-Madison
Joint work with S. Wright

INFORMS Optimization Conference

Denver, CO - March 24, 2018

Smooth nonconvex optimization

We consider an unconstrained smooth problem:

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

Assumptions on f

- f bounded from below.
- f twice continuously differentiable.
- f **is not convex** (generic).

Optimality conditions

Second-order necessary point

x^* satisfies the second-order necessary conditions if

$$\nabla f(x^*) = 0, \quad \nabla^2 f(x^*) \succeq 0.$$

Basic paradigm

If x is not a second-order necessary point, $\exists\ d$ such that

1. $d^\top \nabla f(x) < 0$: gradient-related direction.
   and/or
2. $d^\top \nabla^2 f(x) d < 0$: negative curvature direction
   $\Rightarrow$ **specific to nonconvex problems**,

and f **decreases** along d.

Motivation

Example: Nonconvex formulations of low-rank matrix problems

$$\min_{U \in \mathbb{R}^{n \times r}, V \in \mathbb{R}^{m \times r}} f(U V^\top).$$

For common classes of problems:

- Second-order necessary points are global minimizers (or close).
- Saddle points have negative curvature (strict saddle property).

Other example: Fourth-order tensor orthogonal factorization.

$$\min_{\{u_i\} \subset \mathbb{S}^{n-1}} \sum_{i \neq j} T(u_i, u_i, u_j, u_j).$$

- Solutions are second-order necessary points (Ge et al, '15).

Motivation

Example: Nonconvex formulations of low-rank matrix problems

$$\min_{U \in \mathbb{R}^{n \times r}, V \in \mathbb{R}^{m \times r}} f(U V^\top).$$

For common classes of problems:

- Second-order necessary points are global minimizers (or close).
- Saddle points have negative curvature (strict saddle property).

Other example: Fourth-order tensor orthogonal factorization.

$$\min_{\{u_i\} \subset \mathbb{S}^{n-1}} \sum_{i \neq j} T(u_i, u_i, u_j, u_j).$$

- Solutions are second-order necessary points (Ge et al, '15).

- Interest: Second-order necessary points of nonconvex problems.
- Need: **Efficient algorithms**.

Second-order complexity

Target: $\nabla f(x^*) = 0, \quad \nabla^2 f(x^*) \succeq 0.$

Principle

For a given method generating $\{x_k\}$, two tolerances $\epsilon_g, \epsilon_H \in (0, 1)$:

- **Obj:** bound the **worst-case cost** of reaching x_k such that

$$\|\nabla f(x_k)\| \leq \epsilon_g, \qquad \lambda_k = \lambda_{\min}(\nabla^2 f(x_k)) \geq -\epsilon_H.$$

- Focus: Bound dependencies on ϵ_g, ϵ_H.

Second-order complexity

Target: $\nabla f(x^*) = 0, \quad \nabla^2 f(x^*) \succeq 0.$

Principle

For a given method generating $\{x_k\}$, two tolerances $\epsilon_g, \epsilon_H \in (0, 1)$:

- **Obj:** bound the **worst-case cost** of reaching x_k such that

$$\|\nabla f(x_k)\| \leq \epsilon_g, \qquad \lambda_k = \lambda_{\min}(\nabla^2 f(x_k)) \geq -\epsilon_H.$$

- Focus: Bound dependencies on ϵ_g, ϵ_H.

- Definition of **cost** ?
- Best rates ?

Existing complexity results

Nonlinear optimization literature

- Classical cost: Number of (expensive) iterations.
- Best methods: Newton-type frameworks.
  1 iteration $\Leftrightarrow$ 1 subproblem/Newton system solve.

Existing complexity results

Nonlinear optimization literature

- Classical cost: Number of (expensive) iterations.
- Best methods: Newton-type frameworks.
  1 iteration $\Leftrightarrow$ 1 subproblem/Newton system solve.

Some algorithms	Bounds
Classical trust region *Cartis, Gould, Toint '10*	$\mathcal{O}\left(\max\{\epsilon_g^{-2}\epsilon_H^{-1}, \epsilon_H^{-3}\}\right)$
Cubic regularization *Nesterov, Polyak '06* *Cartis, Gould, Toint '10*	$\mathcal{O}\left(\max\{\epsilon_g^{-\frac{3}{2}}, \epsilon_H^{-3}\}\right)$

Existing complexity results (2)

Learning/Statistics community

- Specific setting $\epsilon_g = \epsilon, \epsilon_H = \mathcal{O}(\sqrt{\epsilon})$.
- Cheaper iterations.
- Main cost: *Hessian-vector products/gradient evaluations*.

Existing complexity results (2)

Learning/Statistics community

- Specific setting $\epsilon_g = \epsilon, \epsilon_H = \mathcal{O}(\sqrt{\epsilon})$.
- Cheaper iterations.
- Main cost: *Hessian-vector products/gradient evaluations*.

Some algorithms	Bounds
Gradient descent methods with random noise *Jin et al '17*	$\tilde{\mathcal{O}}\left(\epsilon^{-2}\right)$
Accelerated gradient methods for nonconvex problems *Carmon et al '16, Allen-Zhu et al '16*	$\tilde{\mathcal{O}}(\epsilon^{-\frac{7}{4}})$

- $\tilde{\mathcal{O}}(\cdot)$: logarithmic factors.
- Results hold with **high probability**.

Our objective

Illustrate all the possible complexities...

- In terms of iterations, evaluations, etc.
- For arbitrary ϵ_g, ϵ_H.
- Deterministic and high probability results.

Our objective

Illustrate all the possible complexities...

- In terms of iterations, evaluations, etc.
- For arbitrary ϵ_g, ϵ_H.
- Deterministic and high probability results.

...in a single framework

- Based on line search.
- Only require Hessian-vector products/gradient calls.
- Best known complexity guarantees.

Outline

Outline

Framework

Parameters: $x_0 \in \mathbb{R}^n,\ \theta \in (0,1),\ \eta > 0, \epsilon_g \in (0,1),\ \epsilon_H \in (0,1)$.

For k=$0, 1, 2, \ldots$

1. Compute a search direction d_k.
2. Perform a backtracking line search to compute $\alpha_k = \theta^{j_k}$ such that

$$f(x_k + \alpha_k d_k) \ < \ f(x_k) - \eta \alpha_k^3 \|d_k\|^3.$$

3. Set $x_{k+1} = x_k + \alpha_k\, d_k$.

Framework

Parameters: $x_0 \in \mathbb{R}^n,\ \theta \in (0,1),\ \eta > 0, \epsilon_g \in (0,1),\ \epsilon_H \in (0,1)$.

For k=$0, 1, 2, \ldots$

1. Compute a search direction $d_k = d_k(\epsilon_g, \epsilon_H)$.
2. Perform a backtracking line search to compute $\alpha_k = \theta^{j_k}$ such that

$$f(x_k + \alpha_k d_k) \ < \ f(x_k) - \eta \alpha_k^3 \|d_k\|^3.$$

3. Set $x_{k+1} = x_k + \alpha_k\, d_k$.

Selecting the search direction $d_k = d_k(\epsilon_g, \epsilon_H)$

Step 1: Use gradient related information

- Compute

$$g_k = \nabla f(x_k), \;\; R_k = \frac{g_k^\top \nabla^2 f(x_k) g_k}{\|g_k\|^2}.$$

- If $R_k < -\epsilon_H$, set

$$d_k = \frac{R_k}{\|g_k\|} g_k.$$

- Elseif $R_k \in [-\epsilon_H, \epsilon_H]$ **and** $\|g_k\| > \epsilon_g$, set

$$d_k = -\frac{g_k}{\|g_k\|^{1/2}}.$$

- Otherwise perform Step 2.

Selecting the search direction d_k (2)

Step 2: Use eigenvalue information

- Compute an eigenpair (v_k, λ_k) such that $\lambda_k = \lambda_{\min}(\nabla^2 f(x_k))$ and

$$\nabla^2 f(x_k) v_k = \lambda_k v_k, \quad v_k^\top g_k \leq 0, \quad \|v_k\| = 1.$$

- **Case $\lambda_k < -\epsilon_H$:** $d_k = -\lambda_k v_k$;
- **Case $\lambda_k > \epsilon_H$ - Newton step:**

$$\nabla^2 f(x_k) d_k = -g_k;$$

- **Case $\lambda_k \in [-\epsilon_H, \epsilon_H]$ - regularized Newton step:**

$$\left(\nabla f^2(x_k) + 2\epsilon_H\right) d_k = -g_k.$$

All possible directions

Direction	$\|g_k\|$	λ_k
$\left(g_k^\top \nabla^2 f(x_k) g_k / \|g_k\|^3\right) g_k$	$\|g_k\| \neq 0$	$\lambda_k \leq R_k \leq -\epsilon_H$
$\left(-1/\|g_k\|^{1/2}\right) g_k$	$\|g_k\| \geq \epsilon_g$	$\lambda_k \leq R_k \leq \epsilon_H$
$\nabla^2 f(x_k) v = \lambda_k v$	-	$\lambda_k < -\epsilon_H < R_k$
$\nabla^2 f(x_k) d = -\nabla f(x_k)$	-	$\lambda_k > \epsilon_H$
$[\nabla^2 f(x_k) + 2\epsilon_H] d = -\nabla f(x_k)$	-	$\|\lambda_k\| \leq \epsilon_H$

All possible directions

Direction	$\|g_k\|$	λ_k
$\nabla^2 f(x_k)v = \lambda_k v$	-	$\lambda_k < -\epsilon_H$
$\nabla^2 f(x_k)d = -\nabla f(x_k)$	-	$\lambda_k > \epsilon_H$
$[\nabla^2 f(x_k) + 2\epsilon_H]d = -\nabla f(x_k)$	-	$\|\lambda_k\| \leq \epsilon_H$

- Gradient steps are **optional** for complexity...
- ...but can save calculation!

Towards a complexity result

Approximate solution

x_k is an (ϵ_g, ϵ_H)-**point** if

$$\min\{\|\nabla f(x_k)\|, \|\nabla f(x_{k+1})\|\} \leq \epsilon_g, \quad \lambda_{\min}(\nabla^2 f(x_k)) \geq -\epsilon_H.$$

Towards a complexity result

Approximate solution

x_k is an (ϵ_g, ϵ_H)-**point** if

$$\min\{\|\nabla f(x_k)\|, \|\nabla f(x_{k+1})\|\} \leq \epsilon_g, \quad \lambda_{\min}(\nabla^2 f(x_k)) \geq -\epsilon_H.$$

On $\min\{\|\nabla f(x_k)\|, \|\nabla f(x_{k+1})\|\}$

- $\|\nabla f(x_{k+1})\|$ appears because of Newton-type steps.
- With a stopping criterion, **same** results with

$$\|\nabla f(x_k)\| \leq \epsilon_g, \quad \lambda_{\min}(\nabla^2 f(x_k)) \geq -\epsilon_H.$$

Assumptions and notations

Assumptions

- $\mathcal{L}_f(x_0) = \{x | f(x) \leq f(x_0)\}$ compact.
- f twice continuously differentiable on a open set containing $\mathcal{L}_f(x_0)$, with Lipschitz continuous Hessian.

- L_H: Lipschitz constant for $\nabla^2 f$.
- f_{low}: lower bound on $\{f(x_k)\}$.
- U_H: upper bound on $\|\nabla^2 f(x_k)\| + 2$.

Decrease bound

Idea: Lower bound the decrease at every step.

Decrease bound

Idea: Lower bound the decrease at every step.

- Five directions;

Decrease bound

Idea: Lower bound the decrease at every step.

- Five directions;
- One line-search proof technique:
  - Unit step case;
  - Backtracking ensures lower bound on α_k.

Decrease bound

Idea: Lower bound the decrease at every step.

- Five directions;
- One line-search proof technique:
  - Unit step case;
  - Backtracking ensures lower bound on α_k.

General decrease lemma

If at the k-th iteration, an (ϵ_g, ϵ_H)-point has not been reached, then

$$f(x_k) - f(x_{k+1}) \geq c \min \left\{ \epsilon_g^{\frac{3}{2}}, \epsilon_H^3, \epsilon_g^3 \epsilon_H^{-3} \right\},$$

where c depends on L_H, η, θ.

Iteration complexity

Iteration complexity bound

The method reaches an (ϵ_g, ϵ_H)-point in at most

$$C \max\left\{\epsilon_g^{-\frac{3}{2}}, \epsilon_H^{-3}, \epsilon_g^{-3}\epsilon_H^3\right\}$$

iterations, with $C = \mathcal{O}\left((f_0 - f_{\text{low}})L_H^3\right)$.

Iteration complexity

Iteration complexity bound

The method reaches an (ϵ_g, ϵ_H)-point in at most

$$C \max\left\{\epsilon_g^{-\frac{3}{2}}, \epsilon_H^{-3}, \epsilon_g^{-3}\epsilon_H^3\right\}$$

iterations, with $C = \mathcal{O}\left((f_0 - f_{\text{low}})L_H^3\right)$.

Specific rates:

- $\epsilon_g = \epsilon,\ \epsilon_H = \sqrt{\epsilon}$: $\mathcal{O}(\epsilon^{-\frac{3}{2}})$.
- $\epsilon_g = \epsilon_H = \epsilon$: $\mathcal{O}(\epsilon^{-3})$.
- Optimal bounds for second-order Newton-type methods (Cartis, Gould and Toint '17).

Outline

Introducing inexactness

The method so far

- **Optimal** iteration complexity;
- Needs **Hessian information**:
  - Linear system solve;
  - Eigenvalue/Eigenvector computation.

Introducing inexactness

The method so far

- **Optimal** iteration complexity;
- Needs **Hessian information**:
  - Linear system solve;
  - Eigenvalue/Eigenvector computation.

Inexactness

- Perform the matrix operations inexactly.
- Main cost unit: **Hessian-vector product**/gradient evaluation.

Example of inexact subroutines

- Linear systems: Accelerated gradient, CG,...
- Eigenvalues: Approximate PCA, Lanczos, CG,...

Conjugate gradient for linear systems

- For Newton/Regularized Newton steps.
- We solve systems of the form $Hd = -g$, with $\epsilon_H I \preceq H \preceq U_H I$.

Conjugate gradient for linear systems

- For Newton/Regularized Newton steps.
- We solve systems of the form $Hd = -g$, with $\epsilon_H I \preceq H \preceq U_H I$.

Conjugate Gradient (CG)

- We apply the conjugate gradient algorithm with stopping criterion:

$$\|Hd + g\| \leq \frac{\xi}{2} \min\{\|g\|, \epsilon_H \|d\|\}, \qquad \xi \in (0, 1).$$

- CG finds such a vector in at most

$$\min\left\{n, \tfrac{1}{2} U_H^{\frac{1}{2}} \epsilon_H^{-\frac{1}{2}} \ln\left(\frac{4}{\xi} U_H^{\frac{3}{2}} \epsilon_H^{-\frac{3}{2}}\right)\right\}$$

iterations/**matrix-vector products.**

Lanczos for eigenvalue computation

- For computing a minimum eigenvalue or eigenvector.
- Can fail if deterministic $\Rightarrow$ **Random start uniform on the sphere**.

Lanczos for eigenvalue computation

- For computing a minimum eigenvalue or eigenvector.
- Can fail if deterministic $\Rightarrow$ **Random start uniform on the sphere**.

Lanczos iterations

Let $H \in \mathbb{R}^{n \times n}$ symmetric with $\|H\| \leq U_H,\ \epsilon_H > 0,\ \delta \in (0,1)$.
If Lanczos is run for (at most)

$$\min\left\{n, \frac{\ln(n/\delta^2)}{2}\sqrt{\frac{U_H}{\epsilon_H}}\right\}$$

iterations/matrix-vector products, then it finds a unit vector v such that

$$v^\top H v \ \leq\ \lambda_{\min}(H) + \frac{\epsilon_H}{2},$$

with probability at least $1 - \delta$.

Framework (inexact ?)

Parameters: $x_0 \in \mathbb{R}^n,\ \theta \in (0,1),\ \eta > 0, \epsilon_g \in (0,1),\ \epsilon_H \in (0,1)$.

For k=$0, 1, 2, \ldots$

1. Compute a search direction $d_k = d_k(\epsilon_g, \epsilon_H)$.
2. Perform a backtracking line search to compute $\alpha_k = \theta^{j_k}$ such that

$$f(x_k + \alpha_k d_k) \ < \ f(x_k) - \frac{\eta}{6}\alpha_k^3 \|d_k\|^3.$$

3. Set $x_{k+1} = x_k + \alpha_k\, d_k$.

Selecting the search direction d_k - Inexact version

Step 1: Use gradient related information

- Compute

$$g_k = \nabla f(x_k), \; R_k = \frac{g_k^\top \nabla^2 f(x_k) g_k}{\|g_k\|^2}.$$

- If $R_k < -\epsilon_H$, set

$$d_k = \frac{R_k}{\|g_k\|} g_k.$$

- Elseif $R_k \in [-\epsilon_H, \epsilon_H]$ **and** $\|g_k\| \geq \epsilon_g$, set

$$d_k = -\frac{g_k}{\|g_k\|^{\frac{1}{2}}}.$$

- Otherwise perform the Inexact Step 2.

Selecting the direction d_k - Inexact version (2)

Inexact Step 2: Use (inexact) eigenvalue information

- Using Lanczos, compute a pair (v_k^i, λ_k^i) such that **w. p.** $1-\delta$,

$$\lambda_k^i = [v_k^i]^\top \nabla^2 f(x_k) v_k^i \leq \lambda_k + \frac{\epsilon_H}{2}, \quad [v_k^i]^\top g_k \leq 0, \quad \|v_k^i\| = 1.$$

- **Case** $\lambda_k^i < -\frac{1}{2}\epsilon_H$: $d_k = -\lambda_k^i v_k^i$;
- **Case** $\lambda_k^i > \frac{3}{2}\epsilon_H$:
  - Inexact Newton: Use CG to obtain

$$\left\|\nabla^2 f(x_k) d_k + g_k\right\| \leq \frac{\xi}{2} \min\left\{\|g_k\|, \epsilon_H \|d_k\|\right\};$$

- **Case** $\lambda_k^i \in [-\frac{1}{2}\epsilon_H, \frac{3}{2}\epsilon_H]$:
  - Inexact regularized Newton: Use CG to obtain

$$\left\|\left[\nabla^2 f(x_k) + 2\epsilon_H\right] d_k + g_k\right\| \leq \frac{\xi}{2} \min\left\{\|g_k\|, \epsilon_H \|d_k\|\right\}.$$

Complexity analysis of the inexact method

- Identical reasoning: 5 steps, 1 line search, 1 proof technique.
- With CG, slightly different formulas.
- Lanczos with random start: decrease only holds in probability.

Decrease lemma

For any iteration k, if x_k is not an (ϵ_g, ϵ_H)-point,

$$f(x_k) - f(x_{k+1}) \geq \hat{c} \min \left\{ \epsilon_g^{\frac{3}{2}}, \epsilon_H^3, \epsilon_g^3 \epsilon_H^{-3} \right\},$$

with probability at least $1 - \delta$, and $\hat{c}$ only depends on L_H, η, θ.

Complexity results

Iteration complexity

An (ϵ_g, ϵ_H)-point is reached in at most

$$\hat{K} := \hat{C} \max \left\{ \epsilon_g^{-\frac{3}{2}}, \epsilon_H^{-3}, \epsilon_g^{-3}\epsilon_H^3, \right\}, \quad \hat{C} = \mathcal{O}\left((f_0 - f_{\text{low}})L_H^3\right)$$

iterations, **with probability at least $1 - \hat{K}\delta$.**

Computational complexity

The number of **Hessian-vector products or gradient evaluations** needed to reach an (ϵ_g, ϵ_H)-point is at most

$$\min \left\{ n, \mathcal{O}\left(U_H^{1/2} \epsilon_H^{-\frac{1}{2}} \log(\epsilon_H^{-1}/\xi) \right), \mathcal{O}\left(U_H^{1/2} \epsilon_H^{-\frac{1}{2}} \log(n/\delta^2) \right) \right\} \hat{K},$$

w. p. at least $1 - \hat{K}\delta$.

Complexity results

Simplified setting: $\epsilon_g = \epsilon, \epsilon_H = \sqrt{\epsilon}$.

An $(\epsilon, \sqrt{\epsilon})$-point is reached in at most

- $\mathcal{O}(\epsilon^{-3/2})$ iterations,
- $\tilde{\mathcal{O}}\left(\epsilon^{-7/4}\right)$ Hessian-vector products/gradient evaluations,

w. p. at least $1 - \mathcal{O}(\epsilon^{-3/2}\delta)$.

$\Rightarrow$ Matches best known results for accelerated gradient-based methods.

Complexity results

Simplified setting: $\epsilon_g = \epsilon, \epsilon_H = \sqrt{\epsilon}$.

An $(\epsilon, \sqrt{\epsilon})$-point is reached in at most

- $\mathcal{O}(\epsilon^{-3/2})$ iterations,
- $\tilde{\mathcal{O}}\left(\epsilon^{-7/4}\right)$ Hessian-vector products/gradient evaluations,

w. p. at least $1 - \mathcal{O}(\epsilon^{-3/2}\delta)$.

$\Rightarrow$ Matches best known results for accelerated gradient-based methods.

Setting $\delta = 0$ gives *in probability 1*:

- Iterations: $\mathcal{O}(\epsilon^{-3/2})$.
- Hessian-vector/gradients: $\mathcal{O}\left(\epsilon^{-7/4}\right)$.

A MATLAB illustration

Setting

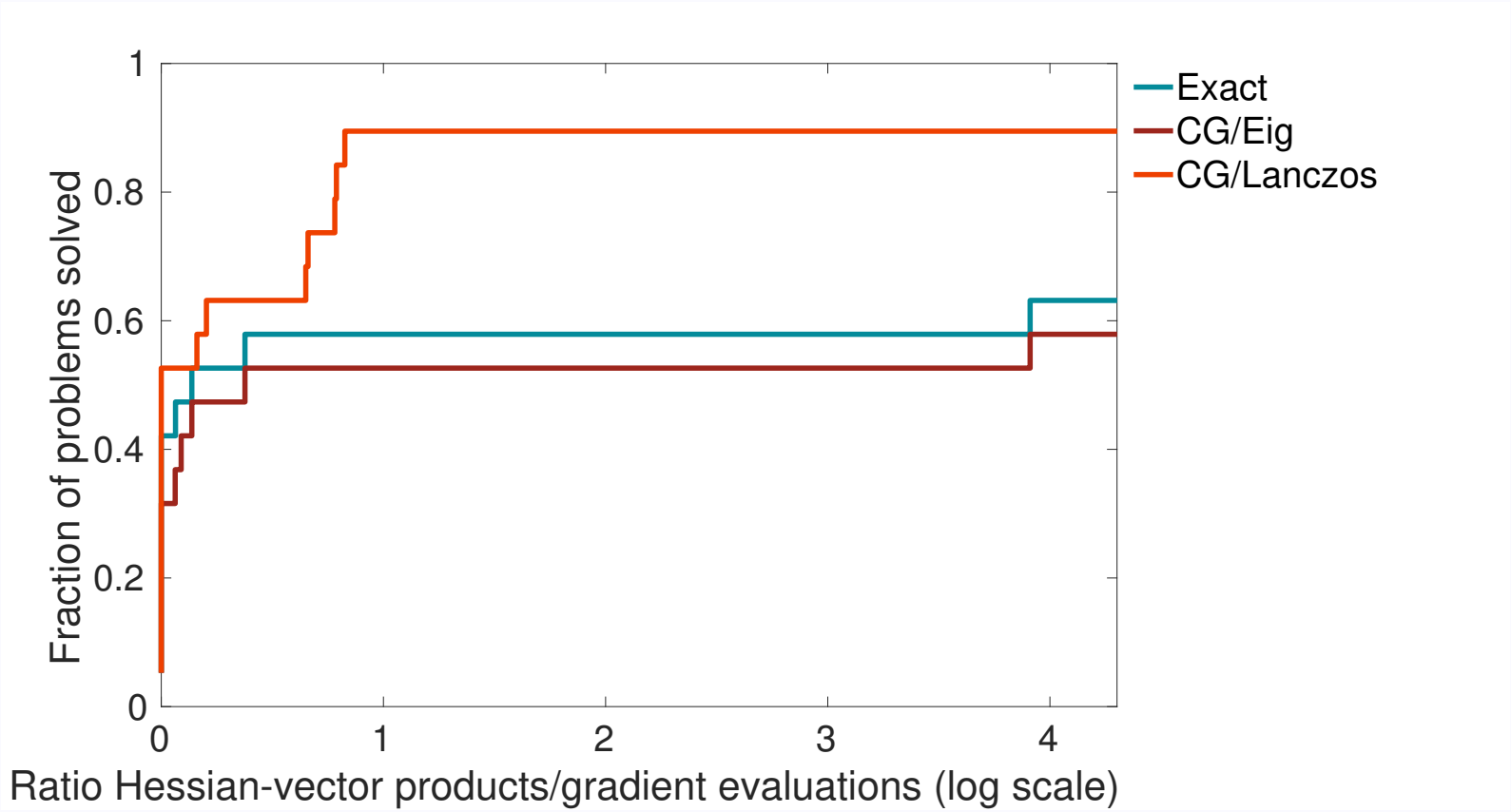
- 56 nonconvex problems from the CUTEst collection.
- Small dimensions (up to 50 variables).
- Convergence test: Find (ϵ_g, ϵ_H)-point with $\epsilon_g = 10^{-3}, \epsilon_H = \sqrt{\epsilon_g}$.

Algorithmic variants

Variant	Eigenvalue	Lin. systems	Gradient Steps
Exact	`eig`	Exact solve	Yes
CG/Eig	`eig`	CG solve	No
CG/Lanczos	Lanczos($\delta = 0.1$)	CG solve	No.

Cost comparison of exact and inexact variants

- Cost unit: One Hessian-vector product or gradient calculation.



Overview of our method

Our algorithm

- Exploits second-order information.
- Match best known complexity guarantees.
- Can be implemented inexactly.
- No need for Lipschitz constants (line search).
- **Complexity analysis of second-order line-search algorithms for smooth nonconvex optimization**, C. W. Royer and S. J. Wright, accepted in *SIAM J. Optim*.

Overview of our method

Our algorithm

- Exploits second-order information.
- Match best known complexity guarantees.
- Can be implemented inexactly.
- No need for Lipschitz constants (line search).
- **Complexity analysis of second-order line-search algorithms for smooth nonconvex optimization**, C. W. Royer and S. J. Wright, accepted in *SIAM J. Optim*.

Since then...

- A new version solely based on CG;
- **A Newton-CG algorithm with complexity guarantees for smooth unconstrained optimization**, C. W. Royer, M. O'Neill and S. J. Wright, arXiv:1803.02924.

One perspective

Follow-up: Tune our method to particular settings:

- Finite-sum problems;
- Hessian properties, e.g. non-degeneracy.

One perspective

Follow-up: Tune our method to particular settings:

- Finite-sum problems;
- Hessian properties, e.g. non-degeneracy.

Thank you for your attention!

croyer2@wisc.edu