Arc Search Methods for Linearly Constrained Optimization

Nick Henderson

Institute for Computational and Mathematical Engineering
Stanford University

INFORMS 2015 Annual Meeting

Outline

Mathematical optimization in $\mathbf{R}^n$

Unconstrained optimization:

$$\underset{x}{\text{minimize}} \quad F(x)$$

Linearly constrained optimization:

$$\begin{array}{rl} \underset{x}{\text{minimize}} & F(x) \\ \text{subject to} & Ax \geq b \end{array}$$

Specifics:

- A is a real matrix with m rows and n columns
- b is a real vector of length m
- x_k is always feasible
- have access to first and second derivatives

$$g_k = \nabla F(x_k), \quad H_k = \nabla^2 F(x_k)$$

This talk

Discuss optimization algorithms that:

- are based on Newton's method for fast convergence
- converge to points satisfying the second order necessary optimality conditions
- use second derivatives while avoiding difficult scaling choices
- work on problems with many variables

Present a research implementation:

- showing that the features listed above can be obtained in practice
- compare to `IPOPT` and `SNOPT` on problems in the CUTEr/st dataset
- show results of experiments with Quasi-Newton methods

Unconstrained optimization

Generate a sequence of iterates $\{x_0, x_1, x_2, \dots\}$ that converges to an optimal point x^*.

Major steps at iterate x_k:

1. compute local information
   - $g_k = \nabla F(x_k)$
   - $H_k = \nabla^2 F(x_k)$
2. check termination conditions
   - First order necessary: $g^* = 0$
   - Second order necessary: $H^* \succeq 0$
   - Second order sufficient: $H^* \succ 0$
3. compute new iterate x_{k+1}
   - the secret sauce

Newton's method

The gold standard in optimization is Newton's method:

$$x_{k+1} = x_k - H_k^{-1} g_k$$

The good:

- if $H_k \succ 0$, then x_{k+1} is the minimizer of a quadratic model of F around x_k
- has a quadratic rate of convergence if $H^* \succ 0$ and x_k is close enough to x^*

The bad:

- x_{k+1} is not defined if H_k is singular

The ugly:

- given arbitrary starting point $\{x_k\}$ may diverge, cycle, or converge to maximizer

Basic strategy: descent methods

Descent methods are designed to emulate Newton's method when it works, but enforce convergence to satisfactory points. These methods satisfy the descent property:

$$F(x_{k+1}) < F(x_k) \text{ for } k = 0, 1, 2, \ldots$$

Two common algorithm classes:

- line search methods
- trust region methods

This talk:

- arc search methods

Line search

The process:

1. compute a search direction p_k
2. invoke a line search procedure to compute step size α_k, which ensures $F(x_k + \alpha_k p_k) < F(x_k)$
3. Update: $x_{k+1} = x_k + \alpha_k p_k$

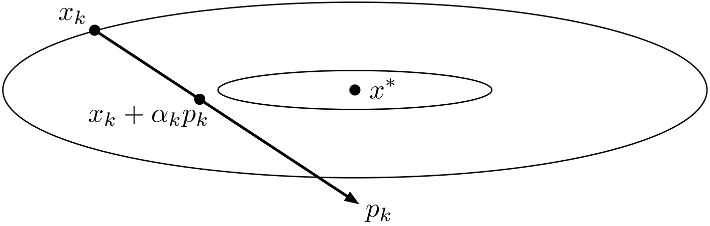


Figure: line search diagram

Trust region

The process:

1. decide trust region radius, Δ_k
2. compute step s_k such that $s_k^T s_k \leq \Delta_k$
3. if $F(x_k + s_k) < F(x_k)$ perform update $x_{k+1} = x_k + s_k$, otherwise decrease Δ_k and recompute s_k

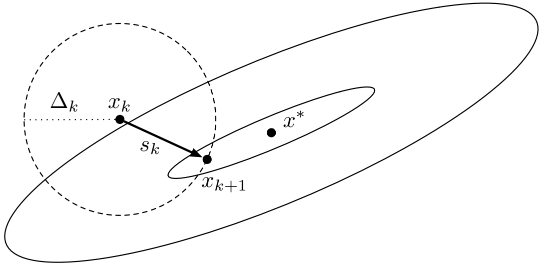


Figure: trust region diagram

Ok, what's the problem?

Line search:

- must come up with special methods to deal with indefinite H_k, not clear what's best
- not clear how to use directions of negative curvature

Trust region:

- may need to solve linear system multiple times
- scaling of the trust region is an issue
- added subproblem difficulty in presence of constraints

Arc search

Arc search methods construct an arc, Γ_k, at each iteration. A univariate search procedure selects an appropriate step parameter α_k, which gives the update:

$$x_{k+1} = x_k + \Gamma_k(\alpha_k)$$

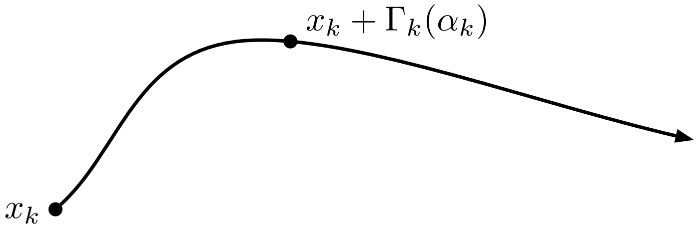


Figure: arc search diagram

Arc search

Arc search methods construct an arc, Γ_k, at each iteration. A univariate search procedure selects an appropriate step parameter α_k, which gives the update:

$$x_{k+1} = x_k + \Gamma_k(\alpha_k)$$

Basic properties:

- $\Gamma_k(\alpha)$ is twice continuously differentiable for $\alpha \in [0, \infty)$
- $\Gamma_k(0) = 0$
- $g_k^T \Gamma'_k(0) < 0$

Issues:

- convergence theory
- definition of arcs

Arc search

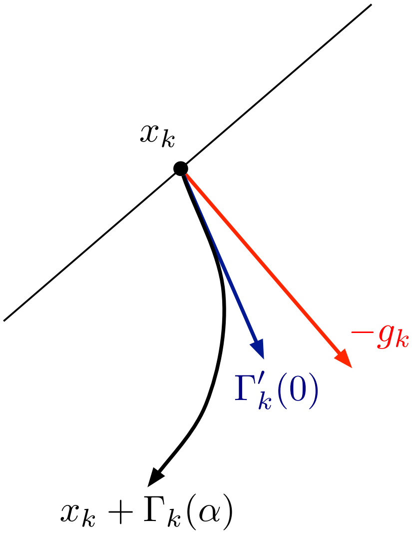


Figure: diagram of basic arc properties

Arc search: $H_k \succ 0$

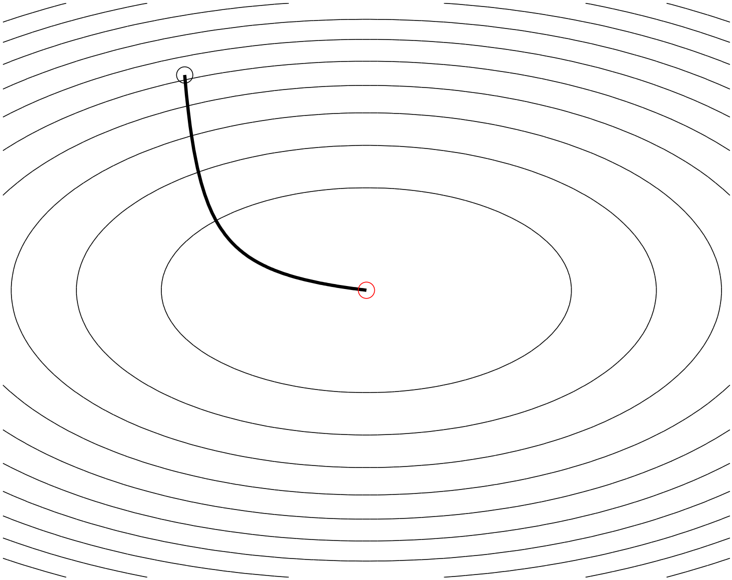


Figure: example arc on positive definite quadratic

Arc search: $H_k \nsucc 0$

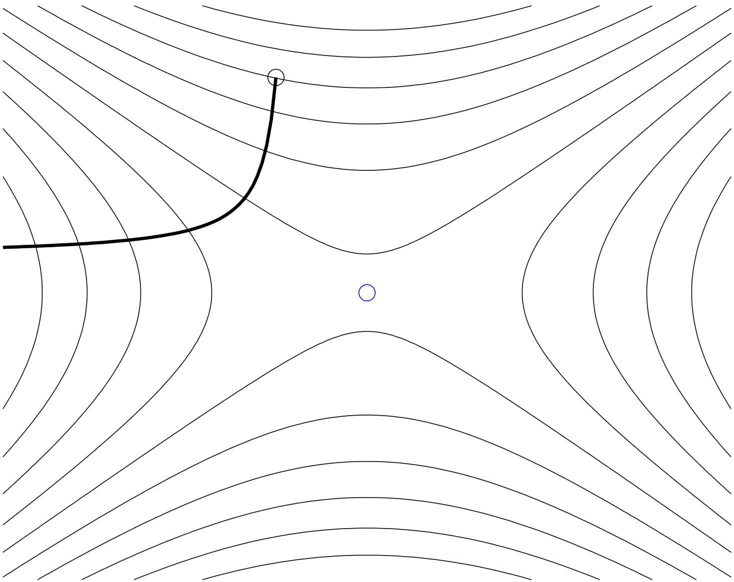


Figure: example arc on indefinite quadratic

Example arcs

Several arcs have been studied. In the following table, s_k is a descent direction and d_k is a direction of negative curvature.

algorithm	$\Gamma_k(\alpha)$
Moré & Sorensen	$\alpha^2 s_k + \alpha d_k$
Goldfarb	$\alpha s_k + \alpha^2 d_k$
Forsgren & Murray	$\alpha(s_k + d_k)$
Behrman	$w(\alpha) : w'(\alpha) = -H_k w(\alpha) - g_k$
Del Gatto	$Qw(\alpha) : w'(\alpha) = -Q^T H_k Q w(\alpha) - Q^T g_k$
`arcopt`	$Qw(\alpha) : (Q^T H_k Q + \pi(\alpha) I) w(\alpha) = -Q^T g_k$

We can prove convergence to second order points with these arcs. For the first three, it is not clear how to scale d_k and we still have to modify H_k to get s_k.

Trust region arc

The trust region subproblem is:

$$\begin{array}{ll} \underset{w}{\text{minimize}} & \frac{1}{2}w^T H_k w + g_k^T w \\ \text{subject to} & \frac{1}{2}w^T w \leq \Delta^2 \end{array}$$

If we have the Lagrange multiplier, π, we can get the solution by solving

$$(H_k + \pi I)w = -g_k$$

We can also think of the solution as an arc parameterized by Δ or π.

Trust region arc details

Given the spectral decomposition $H_k = U\Lambda U^T$, a reparameterized solution to the trust region subproblem is:

$$w(\alpha) = -Uq(\alpha, \Lambda)U^T g_k$$
$$q(\alpha, \lambda) = \frac{\alpha}{\alpha(\lambda - \lambda_{\min}) + 1}$$

- $w(0) = 0$
- $w'(0) = -g_k$, initially steepest descent
- If $H_k \succ 0$, then $\alpha = 1/\lambda_{\min}$ gives the Newton step.
- If $H_k \not\succeq 0$, then $w(s)$ diverges away from saddle point.
- $d/d\alpha \|w(\alpha)\| > 0$ for all $\alpha \geq 0$
- Can apply a perturbation to get second order convergence
- Maintains important properties in 2D subspaces

Linear equality constraints

$$\begin{array}{ll} \underset{x}{\text{minimize}} & F(x) \\ \text{subject to} & Ax = b \end{array}$$

Can be solved with line search in the nullspace of A:

- start with a feasible point x_0, such that $Ax_0 = b$
- $x_{k+1} = x_k + \alpha_k p$ is feasible if $p \in \mathbf{null}(A)$
- say $p = Zp_z$, with $AZ = 0$. The columns of Z span $\mathbf{null}(A)$
- compute p_z by solving $Z^T H_k Z p_z = -Z^T g_k$
- terminate when $Zg = 0$ and $Z^T H Z$ positive semi-definite

Linear inequality constraints

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & F(x) \\ \text{subject to} \quad & Ax \geq b \end{aligned}$$

- active set methods solve this problem by identifying a set of constraints (rows of A) that give equality at the solution, $\overset{*}{A}x^* = b^*$
- the remaining constraints are inactive (strictly feasible), $\bar{A}x^* > \bar{b}$
- during the procedure, the algorithm must be able to:
  - add a constraint when one is encountered by the search procedure
  - delete a constraint when doing so allows a decrease in the objective function

Restricted and unrestricted steps

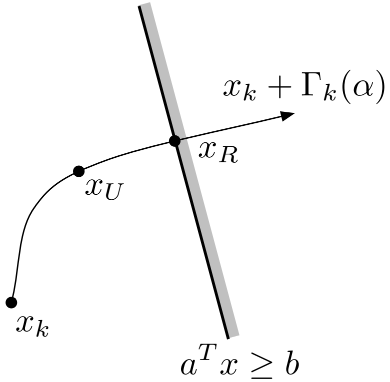


Figure: x_R is a point with a restricted step. x_U is a point with an unrestricted step.

`ARCOPT`: the implementation

Details:

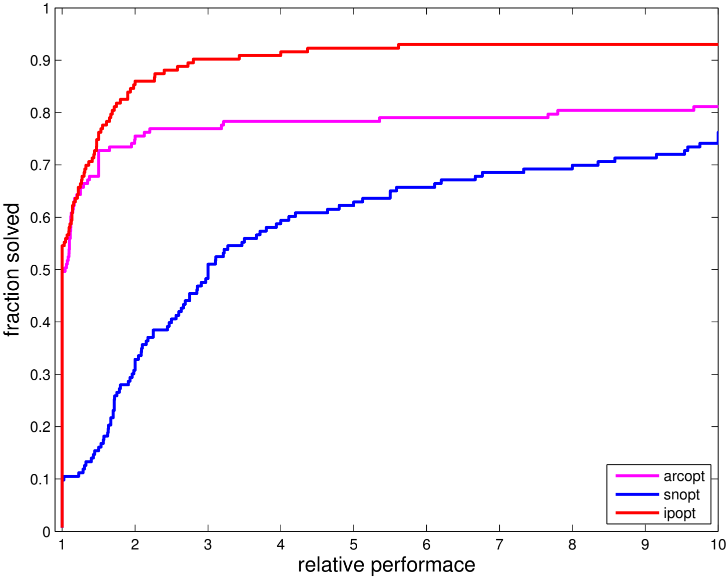
- a Matlab implementation of a trust region arc search method
- an active set, reduced-gradient method
- sparse and stable factorization and updates with LUSOL
  - allows efficient products with Z
- direction of negative curvature computed with `eigs` on $Z^T HZ$
- modified Newton direction computed with `pcg` or `minres` on $(Z^T HZ + \delta I)p = -Z^T g$
- uses EXPAND procedure to
  - improve conditioning of basis matrix
  - mitigate cycling
  - remove roots at $\Gamma_k(0)$ when deleting constraints

CUTEr experiments

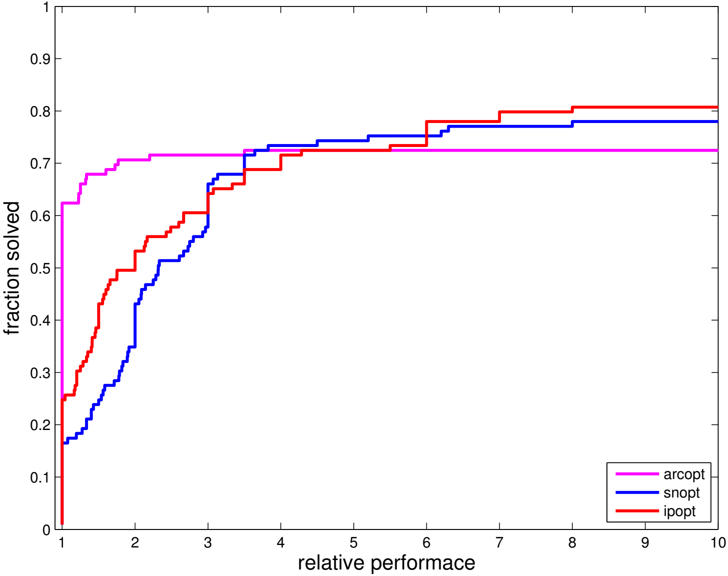
The Constrained and Unconstrained Test Environment provides a collection of about 1000 optimization test problems. The conducted experiments compare `ARCOPT` with SNOPT and IPOPT on:

- 145 unconstrained problems
- 109 problems with bounds on the variables
- 35 problems with a nonlinear objective and linear constraints

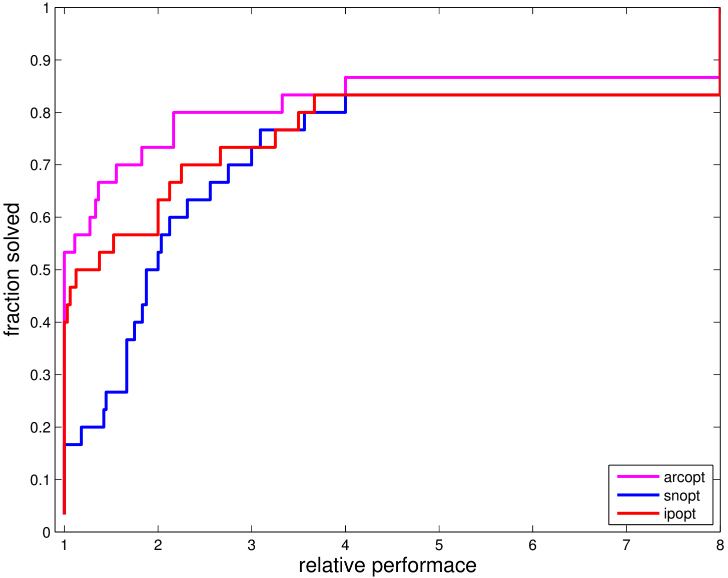
Performance on 145 unconstrained problems



Performance on 109 bounded problems



Performance on 35 linearly constrained problems



Quasi-Newton experiments

Quasi-Newton methods emulate Newton's method by maintaining an approximation to the second derivative:

$$B_k \approx H_k$$

The approximation is updated each iteration with a formula. Two possible updates are:

- BFGS

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k s_k}$$

- SR1

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k)(y_k - B_k s_k)^T}{(y_k - B_k s_k)^T s_k}$$

with $s_k = x_{k+1} - x_k$ and $y_k = g_{k+1} - g_k$

Comparison of BFGS and SR1

BFGS

- update maintains positive definiteness
- rank-2 update
- line search must satisfy curvature condition
  - may not be satisfied if a constraint is hit
  - may have to skip many updates or do something complicated

SR1

- does not maintain positive definiteness
  - cannot be directly applied to line search method
  - may be singular
- rank-1 update
- update condition is weaker and almost always satisfied
  - do not need to skip updates

Comparison of BFGS and SR1

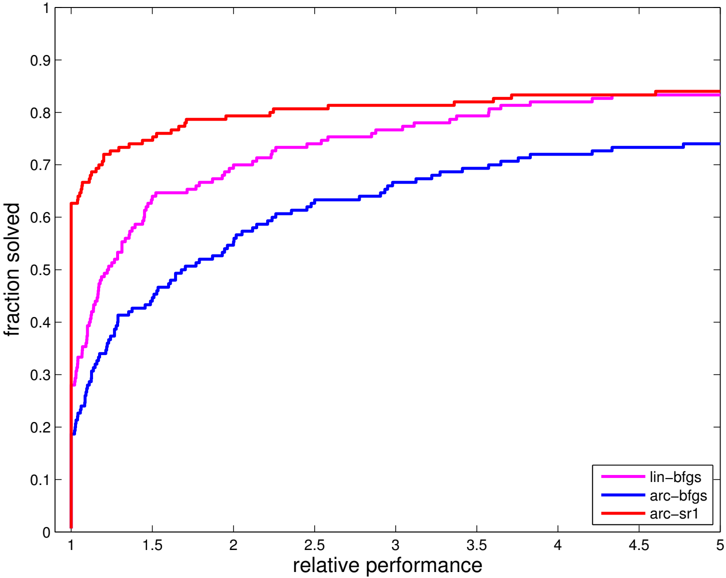
The experiment looks at the relative performance between 3 methods:

- line search with BFGS update
- arc search with BFGS update
- arc search with SR1 update

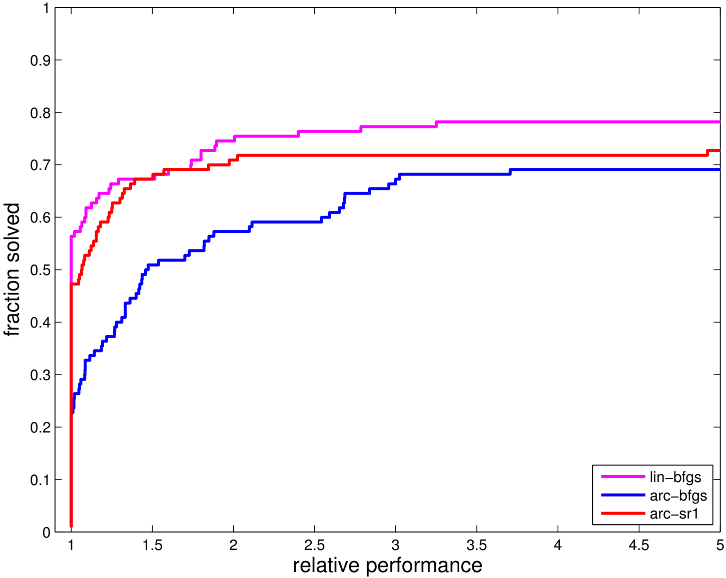
These are all implemented in the same code:

- univariate search procedure is the same
- termination conditions are the same

BFGS/SR1 on 145 unconstrained problems



BFGS/SR1 on 109 bound constrained problems



Thanks!

Anders Forsgren and Walter Murray.
Newton methods for large-scale linear inequality-constrained minimization.
SIAM Journal on Optimization, 7(1):162–176, 1997.

Jorge J. Moré and Danny C. Sorensen.
On the use of directions of negative curvature in a modified newton method.
Mathematical Programming, 16:1–20, 1979.

B. A. Murtagh and M. A. Saunders.
Large-scale linearly constrained optimization.
Mathematical Programming, 14:41–72, 1978.