

# Truncated Newton Method

- approximate Newton methods
- truncated Newton methods
- truncated Newton interior-point methods

# Newton's method

- minimize convex  $f : \mathbf{R}^n \rightarrow \mathbf{R}$
- Newton step  $\Delta x_{\text{nt}}$  found from (SPD) Newton system

$$\nabla^2 f(x) \Delta x_{\text{nt}} = -\nabla f(x)$$

using Cholesky factorization

- backtracking line search on function value  $f(x)$  or norm of gradient  $\|\nabla f(x)\|$
- stopping criterion based on Newton decrement  $\lambda^2/2 = -\nabla f(x)^T \Delta x_{\text{nt}}$  or norm of gradient  $\|\nabla f(x)\|$

## Approximate or inexact Newton methods

- use as search direction an *approximate solution*  $\Delta x$  of Newton system
- idea: no need to compute  $\Delta x_{\text{nt}}$  exactly; only need a good enough search direction
- number of iterations may increase, but if effort per iteration is smaller than for Newton, we win
- examples:
  - solve  $\hat{H}\Delta x = -\nabla f(x)$ , where  $\hat{H}$  is diagonal or band of  $\nabla^2 f(x)$
  - factor  $\nabla^2 f(x)$  every  $k$  iterations and use most recent factorization

## Truncated Newton methods

- approximately solve Newton system using CG or PCG, terminating (sometimes way) early
- also called *Newton-iterative methods*; related to limited memory Newton (or BFGS)
- total effort is measured by cumulative sum of CG steps done
- for good performance, need to tune CG stopping criterion, to use just enough steps to get a good enough search direction
- less reliable than Newton's method, but (with good tuning, good preconditioner, fast  $z \rightarrow \nabla^2 f(x)z$  method, and some luck) can handle very large problems

## Truncated Newton method

- backtracking line search on  $\|\nabla f(x)\|$
- typical CG termination rule: stop after  $N_{\max}$  steps or

$$\eta = \frac{\|\nabla^2 f(x)\Delta x + \nabla f(x)\|}{\|\nabla f(x)\|} \leq \epsilon_{\text{pcg}}$$

- with simple rules,  $N_{\max}$ ,  $\epsilon_{\text{pcg}}$  are constant
- more sophisticated rules adapt  $N_{\max}$  or  $\epsilon_{\text{pcg}}$  as algorithm proceeds (based on, *e.g.*, value of  $\|\nabla f(x)\|$ , or progress in reducing  $\|\nabla f(x)\|$ )  
 $\eta = \min(0.1, \|\nabla f(x)\|^{1/2})$  guarantees (with large  $N_{\max}$ ) superlinear convergence

## CG initialization

- we use CG to approximately solve  $\nabla^2 f(x) \Delta x + \nabla f(x) = 0$
- if we initialize CG with  $\Delta x = 0$ 
  - after one CG step,  $\Delta x$  points in direction of negative gradient (so,  $N_{\max} = 1$  results in gradient method)
  - all CG iterates are descent directions for  $f$
- another choice: initialize with  $\Delta x = \Delta x_{\text{prev}}$ , the previous search step
  - initial CG iterates need not be descent directions
  - but can give advantage when  $N_{\max}$  is small

- simple scheme: if  $\Delta x_{\text{prev}}$  is a descent direction ( $\Delta x_{\text{prev}}^T \nabla f(x) < 0$ )  
start CG from

$$\Delta x = \frac{-\Delta x_{\text{prev}}^T \nabla f(x)}{\Delta x_{\text{prev}}^T \nabla^2 f(x) \Delta x_{\text{prev}}} \Delta x_{\text{prev}}$$

otherwise start CG from  $\Delta x = 0$

## Example

$\ell_2$ -regularized logistic regression

$$\text{minimize } f(w) = (1/m) \sum_{i=1}^m \log(1 + \exp(-b_i x_i^T w)) + \sum_{i=1}^n \lambda_i w_i^2$$

- variable is  $w \in \mathbf{R}^n$
- problem data are  $x_i \in \mathbf{R}^n$ ,  $b_i \in \{-1, 1\}$ ,  $i = 1, \dots, m$ , and regularization parameter  $\lambda \in \mathbf{R}_+^n$
- $n$  is number of features;  $m$  is number of samples/observations



## Hessian and gradient

$$\nabla^2 f(w) = A^T D A + 2\Lambda, \quad \nabla f(w) = A^T g + 2\Lambda w$$

where

$$A = [b_1 x_1 \cdots b_m x_m]^T, \quad D = \mathbf{diag}(h), \quad \Lambda = \mathbf{diag}(\lambda)$$

$$g_i = -(1/m)/(1 + \exp(Aw)_i)$$

$$h_i = (1/m) \exp(Aw)_i / (1 + \exp(Aw)_i)^2$$

we never form  $\nabla^2 f(w)$ ; we carry out multiplication  $z \rightarrow \nabla^2 f(w)z$  as

$$\nabla^2 f(w)z = (A^T D A + 2\Lambda) z = A^T (D(Az)) + 2\Lambda z$$

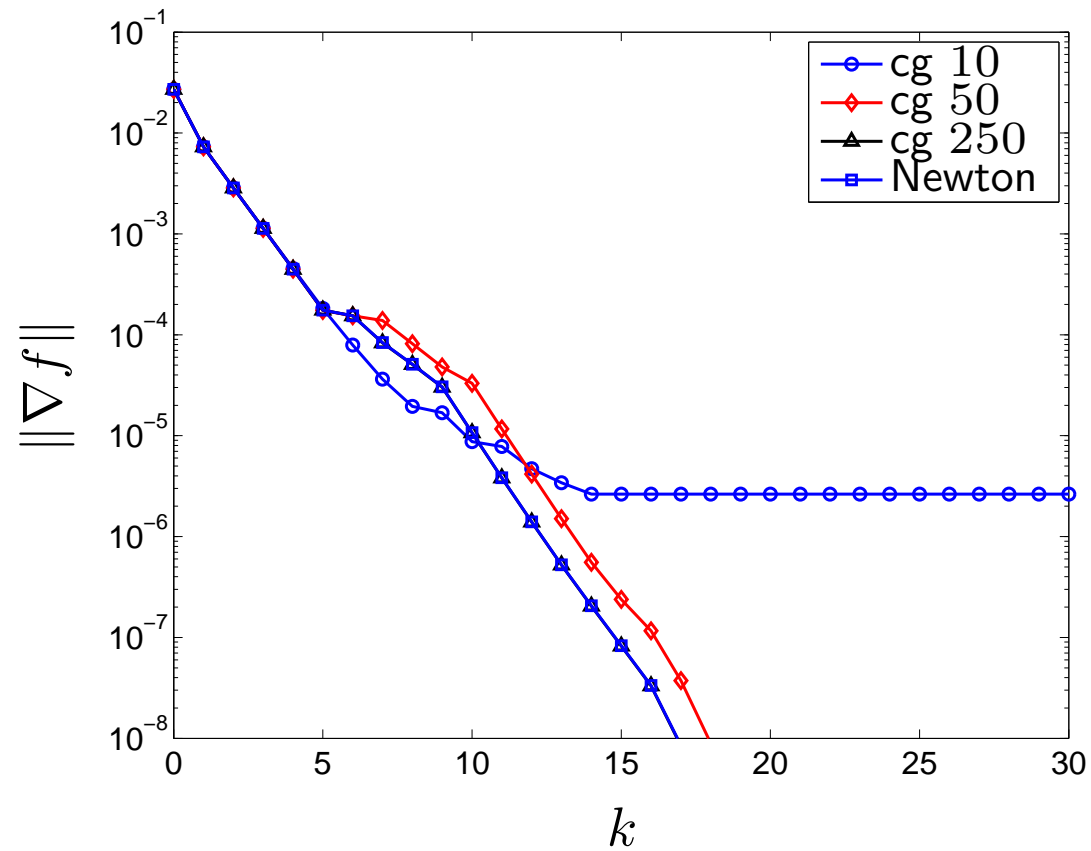
## Problem instance

- $n = 10000$  features,  $m = 20000$  samples (10000 each with  $b_i = \pm 1$ )
- $x_i$  have random sparsity pattern, with around 10 nonzero entries
- nonzero entries in  $x_i$  drawn from  $\mathcal{N}(b_i, 1)$
- $\lambda_i = 10^{-8}$
- around 500000 nonzeros in  $\nabla^2 f$ , and 30M nonzeros in Cholesky factor

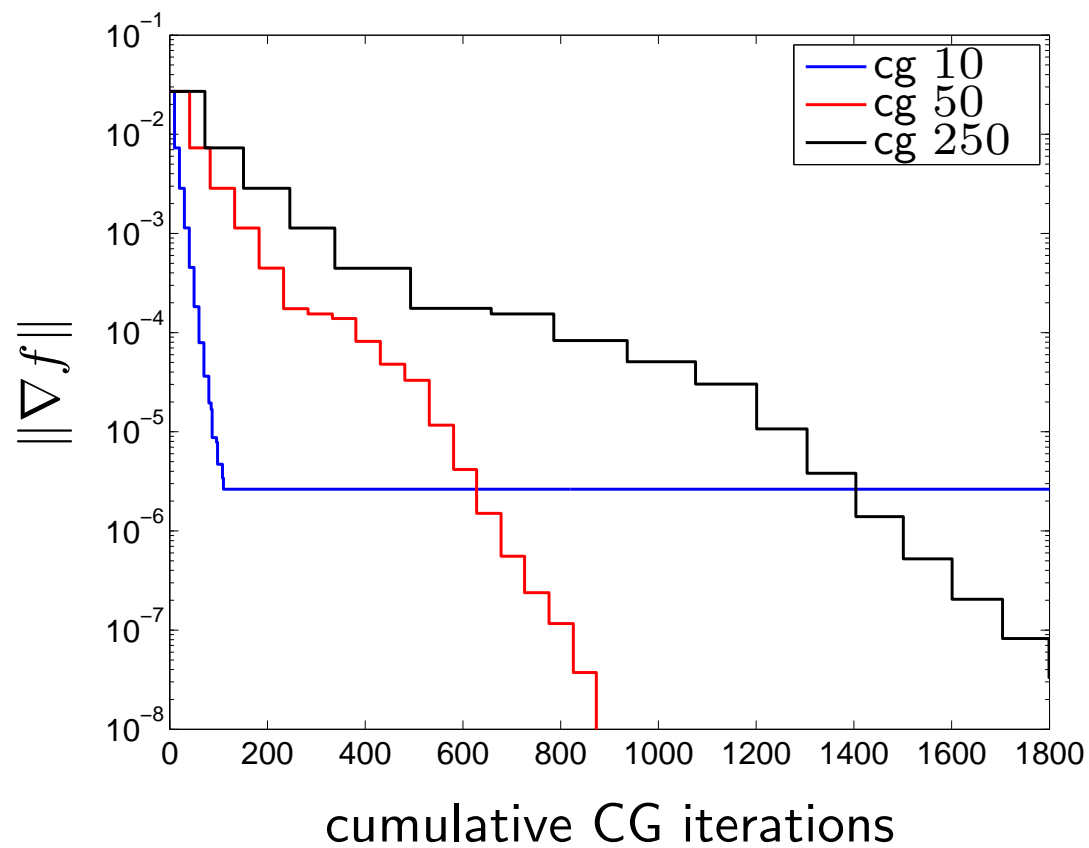
## Methods

- Newton (using Cholesky factorization of  $\nabla^2 f(w)$ )
- truncated Newton with  $\epsilon_{\text{cg}} = 10^{-4}$ ,  $N_{\text{max}} = 10$
- truncated Newton with  $\epsilon_{\text{cg}} = 10^{-4}$ ,  $N_{\text{max}} = 50$
- truncated Newton with  $\epsilon_{\text{cg}} = 10^{-4}$ ,  $N_{\text{max}} = 250$

# Convergence versus iterations



## Convergence versus cumulative CG steps

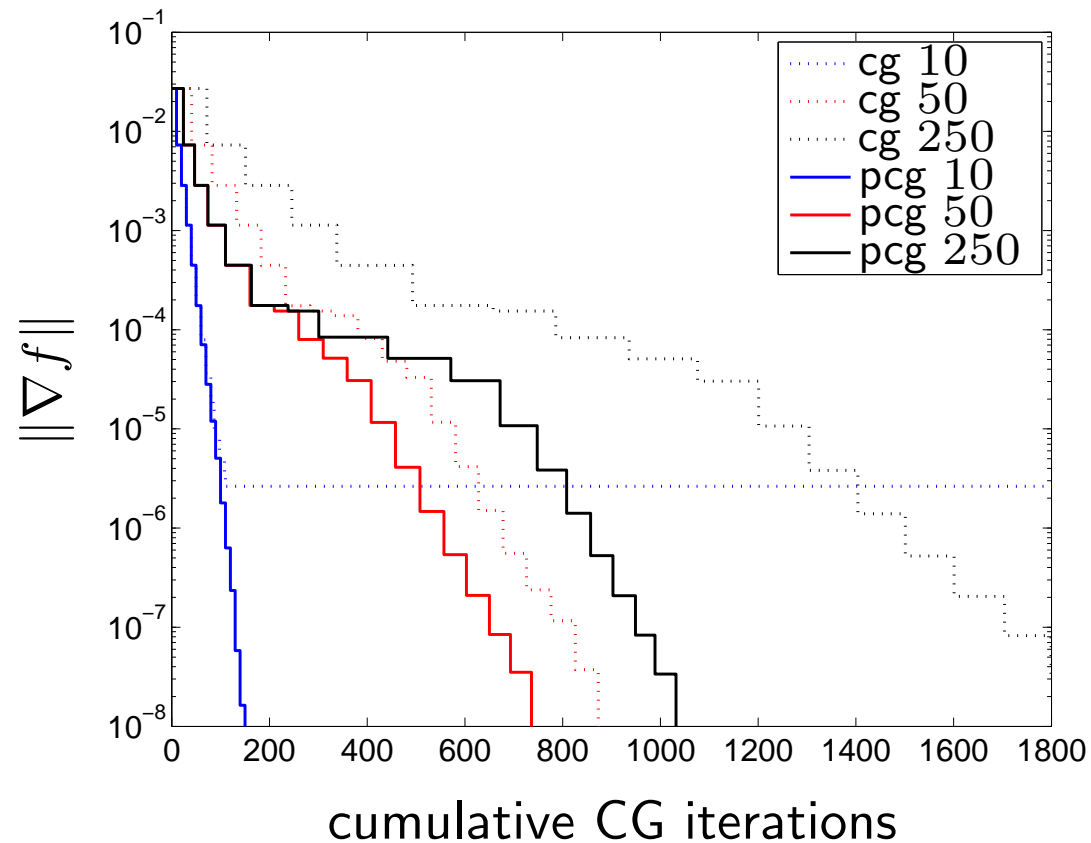


- convergence of exact Newton, and truncated Newton methods with  $N_{\max} = 50$  and 250 essentially the same, in terms of iterations
- in terms elapsed time (and memory!), truncated Newton methods far better than Newton
- truncated Newton with  $N_{\max} = 10$  seems to jam near  $\|\nabla f(w)\| \approx 10^{-6}$
- times (on AMD270 2GHz, 12GB, Linux) in sec:

method	$\ \nabla f(w)\  \leq 10^{-5}$	$\ \nabla f(w)\  \leq 10^{-8}$
Newton	1600	2600
cg 10	4	—
cg 50	17	26
cg 250	35	54

# Truncated PCG Newton method

approximate search direction found via diagonally preconditioned PCG



- diagonal preconditioning allows  $N_{\max} = 10$  to achieve high accuracy; speeds up other truncated Newton methods
- times:

method	$\ \nabla f(w)\  \leq 10^{-5}$	$\ \nabla f(w)\  \leq 10^{-8}$
Newton	1600	2600
cg 10	4	—
cg 50	17	26
cg 250	35	54
pcg 10	3	5
pcg 50	13	24
pcg 250	23	34

- speedups of 1600:3, 2600:5 are not bad  
(and we really didn't do much tuning . . . )



## Extensions

- can extend to (infeasible start) Newton's method with equality constraints
- since we don't use exact Newton step, equality constraints not guaranteed to hold after finite number of steps (but  $\|r_p\| \rightarrow 0$ )
- can use for barrier, primal-dual methods

## Truncated Newton interior-point methods

- use truncated Newton method to compute search direction in interior-point method
- tuning PCG parameters for optimal performance on a given problem class is tricky, since linear systems in interior-point methods often become ill-conditioned as algorithm proceeds
- but can work well (with luck, good preconditioner)

# Network rate control

rate control problem

$$\begin{array}{ll} \text{minimize} & -U(f) = -\sum_{j=1}^n \log f_j \\ \text{subject to} & Rf \preceq c \end{array}$$

with variable  $f$

- $f \in \mathbf{R}_{++}^n$  is vector of flow rates
- $U(f) = \sum_{j=1}^n \log f_j$  is flow utility
- $R \in \mathbf{R}^{m \times n}$  is route matrix ( $R_{ij} \in \{0, 1\}$ )
- $c \in \mathbf{R}^m$  is vector of link capacities

## Dual rate control problem

dual problem

$$\begin{array}{ll}\text{maximize} & g(\lambda) = n - c^T \lambda + \sum_{i=1}^m \log(r_i^T \lambda) \\ \text{subject to} & \lambda \succeq 0\end{array}$$

with variable  $\lambda \in \mathbf{R}^m$

duality gap

$$\begin{aligned}\eta &= -U(f) - g(\lambda) \\ &= -\sum_{j=1}^n \log f_j - n + c^T \lambda - \sum_{i=1}^m \log(r_i^T \lambda)\end{aligned}$$

## Primal-dual search direction (BV §11.7)

primal-dual search direction  $\Delta f$ ,  $\Delta \lambda$  given by

$$(D_1 + R^T D_2 R) \Delta f = g_1 - (1/t) R^T g_2, \quad \Delta \lambda = D_2 R \Delta f - \lambda + (1/t) g_2$$

where  $s = c - Rf$ ,

$$D_1 = \mathbf{diag}(1/f_1^2, \dots, 1/f_n^2), \quad D_2 = \mathbf{diag}(\lambda_1/s_1, \dots, \lambda_m/s_m)$$

$$g_1 = (1/f_1, \dots, 1/f_n), \quad g_2 = (1/s_1, \dots, 1/s_m)$$

## Truncated Newton primal-dual algorithm

primal-dual residual:

$$r = (r_{\text{dual}}, r_{\text{cent}}) = (-g_2 + R^T \lambda, \text{diag}(\lambda)s - (1/t)\mathbf{1})$$

**given**  $f$  with  $Rf \prec c$ ;  $\lambda \succ 0$

**while**  $\eta/g(\lambda) > \epsilon$

$t := \mu m / \eta$

compute  $\Delta f$  using PCG as approximate solution of

$$(D_1 + R^T D_2 R) \Delta f = g_1 - (1/t) R^T g_2$$

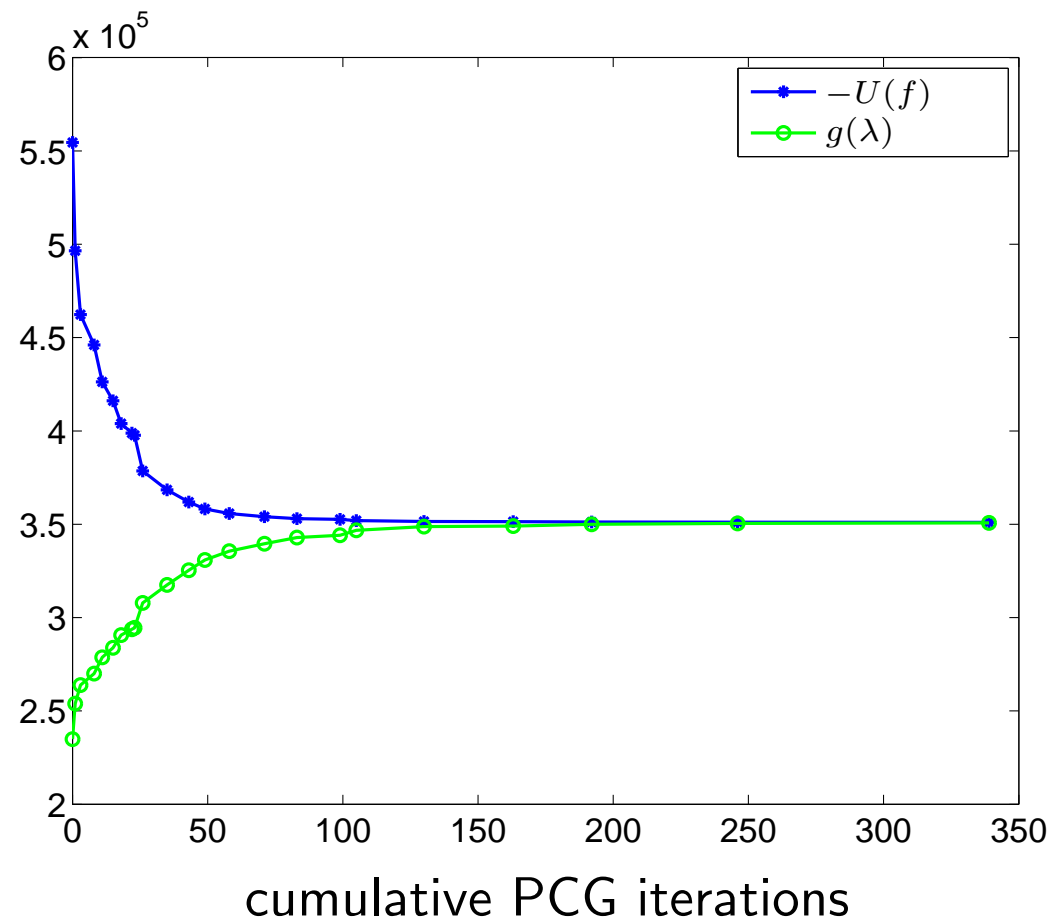
$$\Delta \lambda := D_2 R \Delta f - \lambda + (1/t) g_2$$

carry out line search on  $\|r\|_2$ , and update:

$$f := f + \gamma \Delta f, \lambda := \lambda + \gamma \Delta \lambda$$

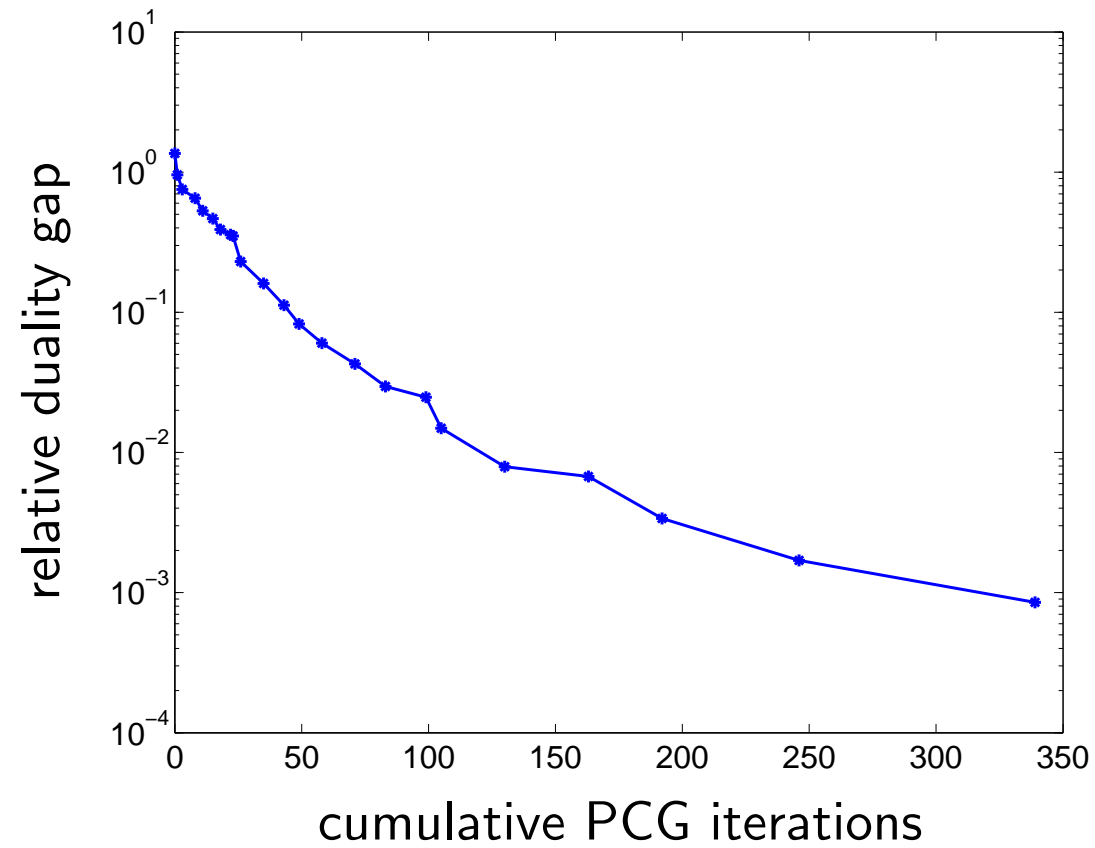
- problem instance
  - $m = 200000$  links,  $n = 100000$  flows
  - average of 12 links per flow, 6 flows per link
  - capacities random, uniform on  $[0.1, 1]$
- algorithm parameters
  - truncated Newton with  $\epsilon_{\text{cg}} = \min(0.1, \eta/g(\lambda))$ ,  $N_{\text{max}} = 200$   
( $N_{\text{max}}$  never reached)
  - diagonal preconditioner
  - warm start
  - $\mu = 2$
  - $\epsilon = 0.001$  (*i.e.*, solve to guaranteed 0.1% suboptimality)

# Primal and dual objective evolution

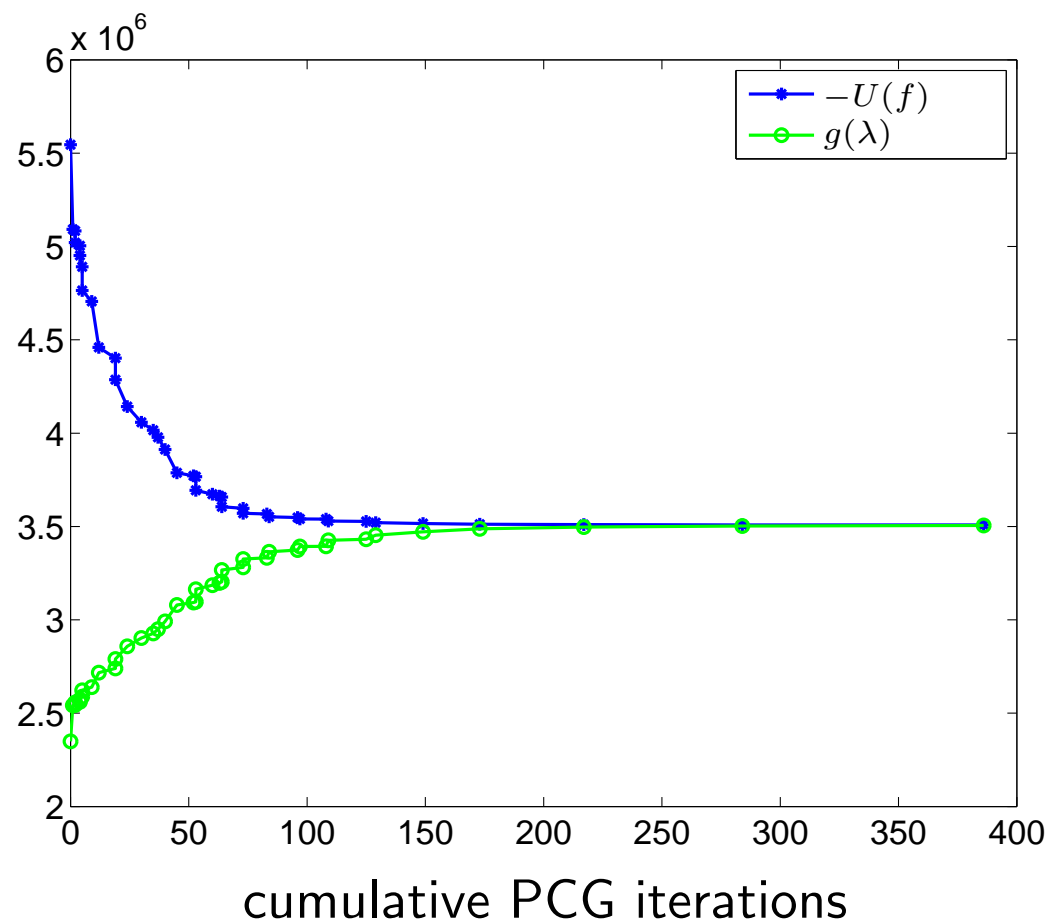




## Relative duality gap evolution



## Primal and dual objective evolution ( $n = 10^6$ )



## Relative duality gap evolution ( $n = 10^6$ )

