Parameter Selection and Pre-Conditioning for a Graph Form Solver

Chris Fougner and **Stephen Boyd**

Springer Lectures, UC Berkeley, 4/6/15

Outline

Graph form problem

$$\begin{array}{ll} \text{minimize} & f(y) + g(x) \\ \text{subject to} & y = Ax \end{array}$$

- $x \in \mathbf{R}^n$ and $y \in \mathbf{R}^m$ are variables
- $f : \mathbf{R}^m \to \mathbf{R} \cup \{\infty\}$, $g : \mathbf{R}^n \to \mathbf{R} \cup \{\infty\}$ are convex closed proper
- infinite values of f, g encode constraints
- constraint is $(x, y) \in \mathcal{G} = \{(x, y) \mid y = Ax\}$, the graph of $x \mapsto Ax$
- graph form includes many common convex problems

Example: Cone programming

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax \preceq_K b \end{array}$$

- $g(x) = c^T x$
- $f(y) = I_K(b - y)$ (I is indicator function)
- includes LP, SOCP, SDP, ...
  (so via CVX*, most convex problems in practice)

Example: Generalized linear model fitting

$$\text{minimize} \quad L(Ax, z) + r(x)$$

- variable x is parameter in statistical model
- z is observed data; A contains associated regressors
- L is loss function, convex in first argument
- r is convex regularizer
- includes LASSO, SVM, logistic regression, ...
- in graph form, $f(y) = L(y, z)$, $g(x) = r(x)$

Radiation treatment planning

$$\begin{array}{ll} \text{minimize} & f(y) \\ \text{subject to} & y = Ax, \quad x \geq 0 \end{array}$$

- x gives n beam intensities; y is radiation dose to m voxels
- A depends on beam/voxel geometry/physics; $A_{ij} \geq 0$
- objective is $f(y) = \sum_{i=1}^m f_i(y_i)$, with

$$f_i(y_i) = \begin{cases} w_i^-(d_i - y_i)_+ + w_i^+(y_i - d_i)_+ & \text{voxel } i \text{ in tumor} \\ w_i^+ y_i & \text{voxel } i \text{ not in tumor} \end{cases}$$

- d_i is prescribed dosage; w_i^+, w_i^- are positive weights
- in graph form, $g(x) = I_+(x)$ encodes $x \geq 0$

Portfolio optimization

$$\begin{array}{ll} \text{maximize} & \mu^T x - \gamma x^T (FF^T + D) x \\ \text{subject to} & x \geq 0, \quad \mathbf{1}^T x = 1 \end{array}$$

- $x \in \mathbf{R}^n$ gives portfolio weights (allocation)
- μ is expected asset return vector
- $\Sigma = FF^T + D$ is asset return covariance ('factor model')
- $F \in \mathbf{R}^{n \times k}$ is factor loading, D is diagonal ('idiosyncratic risk')
- objective is risk-adjusted return; $\gamma > 0$ is risk aversion parameter
- in graph form: $y = \begin{bmatrix} F^T \\ \mathbf{1}^T \end{bmatrix} x \in \mathbf{R}^{k+1}$,

$$g(x) = -\mu^T x + \gamma x^T D x + I_+(x), \quad f(y) = \gamma y^T y + I_{y_{k+1}=1}(y)$$

Outline

Dual problem

- Lagrange function: $L(x, y, \nu) = f(y) + g(x) + \nu^T(Ax - y)$
- dual function:

$$\inf_{x,y} L(x, y, \nu) = -f^*(\nu) - g^*(-A^T\nu)$$

- dual problem, with new variable $\mu = -A^T\nu$

$$\begin{array}{ll} \text{maximize} & -f^*(\nu) - g^*(\mu) \\ \text{subject to} & \mu = -A^T\nu \end{array}$$

  ...also a graph form problem

- duality gap $\eta = f(y) + f^*(\nu) + g(x) + g^*(\mu)$
- for (x, y, μ, ν) feasible, $\eta \geq 0$ (and gives bound on suboptimality)

Optimality conditions

1. *primal feasibility*: $y = Ax$
2. *dual feasibility*: $\mu = -A^T\nu$
3. *zero gap*: $f(y) + f^*(\nu) + g(x) + g^*(\mu) = 0$

- for any x, y, μ, ν (by definition),

$$f(y) + f^*(\nu) \geq \nu^T y, \qquad g(x) + g^*(\mu) \geq \mu^T x$$

  so can replace zero gap with *Fenchel feasibility*:

$$f(y) + f^*(\nu) = \nu^T y, \quad g(x) + g^*(\mu) = \mu^T x$$

- same as: y minimizes $f(y) - \nu^T y$, x minimizes $g(x) - \mu^T x$

Outline

Graph form problem

Dual and optimality conditions

Algorithm

Pre-conditioning

POGS

ADMM for constrained minimization

- convex constrained problem

$$\begin{array}{ll} \text{minimize} & \phi(x) \\ \text{subject to} & x \in \mathcal{C} \end{array}$$

- ADMM (alternating directions method of multipliers):

  for $k = 1, 2, \ldots$

  $$x^{k+1/2} := \mathbf{prox}_\phi(x^k - \tilde{x}^k)$$
  $$x^{k+1} := \Pi(x^{k+1/2} + \tilde{x}^k)$$
  $$\tilde{x}^{k+1} := \tilde{x}^k + x^{k+1/2} - x^{k+1}$$

  until converged

- $\mathbf{prox}_\phi$ is proximal operator of ϕ,

$$\mathbf{prox}_\phi(v) = \operatorname*{argmin}_x \left(\phi(x) + (\rho/2)\|x - v\|_2^2\right)$$

- convergence theory: $x^k - x^{k+1/2} \to 0$, $\phi(x^{k+1/2}) \to \inf_{x \in \mathcal{C}} \phi(x)$

Graph projection ADMM [Parikh 2014]

- apply ADMM for constrained minimization to graph form problem
- yields graph projection ADMM:

  for $k = 1, 2, \ldots$

  $$(x^{k+1/2},\ y^{k+1/2}) := \left(\mathbf{prox}_g(x^k - \tilde{x}^k),\ \mathbf{prox}_f(y^k - \tilde{y}^k)\right)$$
  $$(x^{k+1}, y^{k+1}) := \Pi(x^{k+1/2} + \tilde{x}^k,\ y^{k+1/2} + \tilde{y}^k)$$
  $$(\tilde{x}^{k+1}, \tilde{y}^{k+1}) := (\tilde{x}^k + x^{k+1/2} - x^{k+1},\ \tilde{y}^k + y^{k+1/2} - y^{k+1})$$

  until converged
- projection onto $\mathcal{G}$ is

$$\Pi(c, d) = K^{-1} \begin{bmatrix} c + A^T d \\ 0 \end{bmatrix}, \qquad K = \begin{bmatrix} I & A^T \\ A & -I \end{bmatrix}$$

Efficient graph projection

- direct method:
  - factorize K (which is quasidefinite)
  - cache factorization so each subsequent iteration is a back-solve
- indirect/iterative method:
  - use CG/LSQR to approximately compute projection
  - warm start subsequent projections from last iterate

Iterate properties

- iterates (x^k, y^k, μ^k, ν^k) are primal and dual feasible,

$$Ax^{k+1/2} = y^{k+1/2}, \qquad -A^T\nu^{k+1/2} = \mu^{k+1/2}$$

  and Fenchel feasible in limit (when f and g are smooth)

- with $\mu^{k+1/2} = -\rho(x^{k+1/2} - x^k + \tilde{x}^k)$, $\nu^{k+1/2} = -\rho(y^{k+1/2} - y^k + \tilde{y}^k)$,

$$(x^{k+1/2}, y^{k+1/2}, \mu^{k+1/2}, \nu^{k+1/2})$$

  is Fenchel feasible, and primal and dual feasible in limit:

$$Ax^{k+1/2} - y^{k+1/2} \to 0, \qquad A^T\nu^{k+1/2} + \mu^{k+1/2} \to 0$$

  (with *no assumptions* on f and g)

Outline

Graph form problem

Dual and optimality conditions

Algorithm

Pre-conditioning

POGS

Pre-conditioning

- with D, E invertible, define $\hat{y} = Dy, \quad \hat{x} = E^{-1}x$
- solve (graph form) problem with variables $\hat{x}$, $\hat{y}$

$$\begin{array}{ll} \text{minimize} & f(D^{-1}\hat{y}) + g(E\hat{x}) \\ \text{subject to} & \hat{y} = (DAE)\hat{x} \end{array}$$

- called *pre-conditioned graph form problem*
- scaling D and E has same effect as changing ρ
- goal: choose D, E so
  - graph projection ADMM is not (much) harder to carry out
  - practical convergence is faster
- first condition holds when f, g are separable and D, E are diagonal

Diagonal pre-conditioning

- heuristic: choose diagonal D, E so that $\sigma_i(DAE) \approx 1$
- supported by (some) theory, numerical experiments
- heuristic for heuristic: *equilibrate* DAE
  i.e., choose D and E so that rows (and columns) have same norm:

$$\sum_{j=1}^{n} (D_{ii} A_{ij} E_{jj})^2 = n\alpha, \qquad \sum_{i=1}^{m} (D_{ii} A_{ij} E_{jj})^2 = m\alpha$$

- find D and E by minimizing convex function

$$\sum_{i=1}^{m} \sum_{j=1}^{n} A_{ij}^2 e^{u_i + v_j} - n\mathbf{1}^T u - m\mathbf{1}^T v$$

  by (simple) coordinate minimization; take $D_{ii} = e^{u_i/2}$, $E_{jj} = e^{v_j/2}$
  (recovers Sinkhorn-Knopp algorithm)

Outline

Graph form problem

Dual and optimality conditions

Algorithm

Pre-conditioning

POGS

Proximal Graph Solver (POGS)

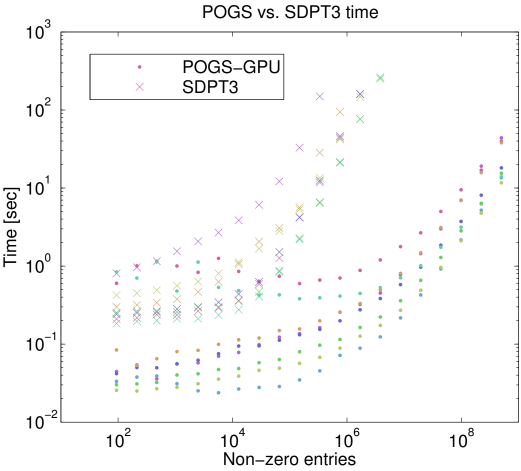
- developed by Chris Fougner
- open source C++ implementation, on `github`
- targets CPUs and GPUs, with various wrappers
- handles sparse and dense A, direct and indirect solvers
- for now, only fully separable f and g
- includes proximal operator library; easy to extend
- algorithm only slightly more complicated than description above (*e.g.*, adaptive ρ-update, regularized equilibration)

Testing

- POGS was tested on many problem instances
  - from many application areas
  - of varying dimensions
  - of varying difficulty
- results verified against (high accuracy) interior-point method (where possible)
- since we want a general solver, *no tuning of any POGS algorithm parameters*
- timing includes transfer to/from GPU, factorization, ...

POGS-GPU versus SDPT3

results for 3GHz Core i7, Nvidia K40

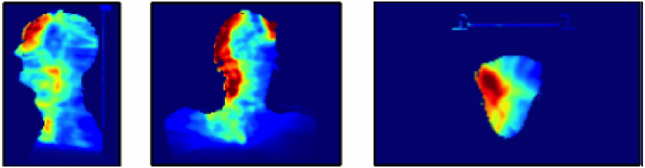


Performance summary

POGS-GPU versus SDPT3

- POGS solves problems $1000\times$ larger in same time
- POGS solves same problems $100\times$ (or more) faster
- limitation is GPU memory

Radiation treatment planning



- 0.4 GB problem, $m = 360000$ voxels, $n = 360$ beams
- checked against interior-point method and actual treatment plan used
- solve times
  - conventional method: 8 hours
  - ECOS (interior-point method): 1 hour
  - POGS (cold start): 5 seconds
  - POGS (warm start): 2 seconds
- enables real-time treatment planning