Linear Programming: Barrier and First Order Methods

Joachim Dahl & Qi Huangfu

Cardinal Operations

CO@Work - September 2024

杉数科技
Cardinal Operations

Linear Programming

Standard primal and dual linear programming problems:

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax = b \\ & x \geqslant 0, \end{array} \qquad \begin{array}{ll} \text{maximize} & b^T y \\ \text{subject to} & c - A^T y = s \\ & s \geqslant 0. \end{array}$$

Optimality conditions:

$$\begin{bmatrix} 0 & A \\ -A^T & 0 \end{bmatrix} \begin{bmatrix} y \\ x \end{bmatrix} + \begin{bmatrix} -b \\ c \end{bmatrix} = \begin{bmatrix} 0 \\ s \end{bmatrix},$$
$$x \geqslant 0, \quad s \geqslant 0, \quad Xs = 0.$$

Consequently,

$$c^T x - b^T y = 0.$$

Barrier methods

Primal barrier problem

$$\begin{array}{ll} \text{minimize} & c^T x - \mu \sum_i \log x_i \\ \text{subject to} & Ax = b \\ & x \geqslant 0, \end{array}$$

for a centrality parameter $\mu > 0$. Lagrange function:

$$L(x, y) = c^T x - \mu \sum_i \log x_i - y^T (Ax - b)$$

Optimality conditions:

$$Ax = b, \quad x \geqslant 0, \quad c - A^T y = \mu X^{-1} e.$$

Dual barrier problem

$$\begin{array}{ll} \text{maximize} & b^T y + \mu \sum_i \log s_i \\ \text{subject to} & c - A^T y = s \\ & s \geqslant 0. \end{array}$$

has same optimality conditions.

Primal-dual methods and the central path

Perturbed optimality conditions:

$$\begin{bmatrix} 0 & A \\ -A^T & 0 \end{bmatrix} \begin{bmatrix} y \\ x \end{bmatrix} + \begin{bmatrix} -b \\ c \end{bmatrix} = \begin{bmatrix} 0 \\ s \end{bmatrix},$$

$$x \geqslant 0, \quad s \geqslant 0, \quad Xs = \mu e.$$

Linearized central path:

$$A(x+\Delta x) = b, \quad c-A^T(y+\Delta y) = s+\Delta s, \quad X\Delta s+S\Delta x = \mu e-Xs$$

or equivalently

$$\begin{bmatrix} 0 & A \\ -A^T & X^{-1}S \end{bmatrix} \begin{bmatrix} \Delta y \\ \Delta x \end{bmatrix} = -\begin{bmatrix} r_p \\ r_d + X^{-1} r_c \end{bmatrix}$$

where

$$r_p := Ax - b, \quad r_d := c - A^T y - s, \quad r_c = Xs - \mu e.$$

A practical primal-dual algorithm

Step 1. Select interior starting point. E.g.,

$$x = s = e, \qquad y = 0.$$

Step 2. Compute residuals and check termination.

$$r_p := Ax - b, \quad r_d := c - A^T y - s$$

Terminate if r_p, r_d and $x^T s$ are all small.

Step 3. Compute affine search direction.

$$\begin{bmatrix} 0 & A \\ -A^T & X^{-1}S \end{bmatrix} \begin{bmatrix} \Delta y_{\text{aff}} \\ \Delta x_{\text{aff}} \end{bmatrix} = - \begin{bmatrix} r_p \\ r_d + s \end{bmatrix}$$

$$S\Delta x_{\text{aff}} + X\Delta s_{\text{aff}} = -Xs.$$

and step-sizes

$$\alpha^P_{\text{aff}} := \max\{\alpha \in [0,1] \mid x + \alpha\Delta x_{\text{aff}} \geqslant 0\}$$
$$\alpha^D_{\text{aff}} := \max\{\alpha \in [0,1] \mid s + \alpha\Delta s_{\text{aff}} \geqslant 0\}.$$

A practical primal-dual algorithm

Step 4. Select centering parameter.

$$\sigma := \left(\frac{(x + \alpha^P_{\text{aff}} \Delta x_{\text{aff}})^T (s + \alpha^D_{\text{aff}} \Delta s_{\text{aff}})}{x^T s} \right)^3 .$$

Step 5. Compute combined search direction.

$$\begin{bmatrix} 0 & A \\ -A^T & X^{-1}S \end{bmatrix} \begin{bmatrix} \Delta y \\ \Delta x \end{bmatrix} = - \begin{bmatrix} r_p \\ r_d + X^{-1} r_c' \end{bmatrix}$$

$$S\Delta x + X \Delta s = -Xs + \sigma \mu e - \Delta X_{\text{aff}} \Delta s_{\text{aff}} =: -r_c'.$$

Step 6. Compute step-sizes and update iterates.

$$\alpha^P := \max\{\alpha \in [0,1] \mid x + \alpha \Delta x \geqslant 0\}$$
$$\alpha^D := \max\{\alpha \in [0,1] \mid s + \alpha \Delta s \geqslant 0\}.$$

$$x := x + 0.99\alpha^P \Delta x, \quad s := s + 0.99\alpha^D \Delta s, \quad y := y + 0.99\alpha^D \Delta y.$$

A practical primal-dual algorithm.

The system

$$\begin{bmatrix} 0 & A \\ -A^T & X^{-1}S \end{bmatrix} \begin{bmatrix} \Delta y \\ \Delta x \end{bmatrix} = \begin{bmatrix} q_p \\ q_d \end{bmatrix}$$

is typically solved by block elimination

$$AXS^{-1}A^T\Delta y = q_p - AXS^{-1}q_d, \quad \Delta x = XS^{-1}(q_d + A^T\Delta y).$$

- We use a sparse Cholesky factorization to solve for Δy.
- Important to reorder rows and columns to reduce fill-in using minimum-degree or nested dissection reordering.
- Important to handle dense columns in A separately.
- **Algorithm assumes primal-dual feasibility.**

The simplified homogeneous self-dual embedding

Using $(\tau, \kappa) \geqslant 0$, we embed the optimality conditions into

$$G(z) := \begin{bmatrix} 0 & A & -b \\ -A^T & 0 & c \\ b^T & -c^T & 0 \end{bmatrix} \begin{bmatrix} y \\ x \\ \tau \end{bmatrix} - \begin{bmatrix} 0 \\ s \\ \kappa \end{bmatrix},$$

where $z := (x, \tau, s, \kappa, y)$. The system $G(z) = 0$ always has solution, and

$$\langle x, s \rangle + \tau\kappa = 0.$$

Assume $G(z) = 0$ and $\tau + \kappa > 0$.

- If $\tau > 0$ then $(x, s, y)/\tau$ is primal-dual optimal.
- If $\kappa > 0$ and $b^T y > 0$ then the primal problem is infeasible.
- If $\kappa > 0$ and $c^T x < 0$ then the dual problem is infeasible.

Redefined central path

We redefine the central path as solutions to

$$G(z) = 0, \quad Xs = \mu e, \quad \tau\kappa = \mu.$$

Linearized central path equations:

$$G(\Delta z) = -G(z),$$
$$S\Delta x + X\Delta s = \mu e - Xs, \quad \kappa\Delta\tau + \tau\Delta\kappa = \mu - \kappa\tau.$$

Update using a single step size:

$$z := z + \alpha\Delta z$$

where

$$(x, s, \tau, \kappa) + \alpha(\Delta x, \Delta s, \Delta\tau, \Delta\kappa) \geqslant 0.$$

A practical algorithm for HSD

Step 1. Select interior starting point.

$$x = s = e, \quad y = 0, \quad \tau = \kappa = 1.$$

Step 2. Compute residuals and check termination.

$$r_p := Ax - b\tau, \quad r_d := c\tau - A^T y - s, \quad r_g = b^T x - c^T y - \kappa$$

Terminate if r_p, r_d and r_g are all small.

Step 3. Compute affine search direction.

$$G(\Delta z_{\text{aff}}) = -G(z),$$
$$S\Delta x_{\text{aff}} + X\Delta s_{\text{aff}} = -Xs, \quad \kappa\Delta\tau_{\text{aff}} + \tau\Delta\kappa_{\text{aff}} = -\tau\kappa$$

and step size

$$\alpha_{\text{aff}} := \max\{\alpha \in [0,1] \mid (x,s,\tau,\kappa)+\alpha(\Delta x_{\text{aff}}, \Delta z_{\text{aff}}, \Delta\tau_{\text{aff}}, \Delta\kappa_{\text{aff}}) \geqslant 0\}.$$

A practical algorithm for HSD

Step 4. Select centering parameter.

$$\sigma := (1 - \alpha_{\text{aff}})^3.$$

Step 5. Compute combined search direction.

$$G(\Delta z) = -(1 - \sigma)G(z),$$
$$S\Delta x + X\Delta s = -Xs + \sigma\mu e - \Delta X_{\text{aff}}\Delta s_{\text{aff}},$$
$$\kappa\Delta\tau + \tau\Delta\kappa = -\tau\kappa + \sigma\mu - \Delta\tau_{\text{aff}}\Delta\kappa_{\text{aff}}.$$

Step 6. Compute step-size and update iterates.

$$\alpha := \max\{\alpha \in [0, 1] \mid (x, s, \tau, \kappa) + \alpha(\Delta x_{\text{aff}}, \Delta z_{\text{aff}}, \Delta\tau_{\text{aff}}, \Delta\kappa_{\text{aff}}) \geqslant 0\}.$$

$$z := z + 0.99\alpha\Delta z.$$

A practical algorithm for HSD

We solve

$$\begin{bmatrix} 0 & A & -b \\ -A^T & 0 & c \\ b^T & -c^T & 0 \end{bmatrix} \begin{bmatrix} \Delta y \\ \Delta x \\ \Delta \tau \end{bmatrix} - \begin{bmatrix} 0 \\ \Delta s \\ \Delta \kappa \end{bmatrix} = - \begin{bmatrix} r_p \\ r_d \\ r_g \end{bmatrix}$$

$$S\Delta x + X\Delta s = -r_c, \quad \kappa\Delta\tau + \tau\Delta\kappa = -r_{\tau\kappa}$$

by eliminating Δs and $\Delta\kappa$,

$$\begin{bmatrix} 0 & A & -b \\ -A^T & X^{-1}S & c \\ b^T & -c^T & \kappa/\tau \end{bmatrix} \begin{bmatrix} \Delta y \\ \Delta x \\ \Delta \tau \end{bmatrix} = - \begin{bmatrix} r_p \\ r_d + X^{-1}r_c \\ r_g + r_{\tau\kappa}/\tau \end{bmatrix}.$$

A practical algorithm for HSD

We solve

$$\begin{bmatrix} 0 & A & -b \\ -A^T & X^{-1}S & c \\ b^T & -c^T & \kappa/\tau \end{bmatrix} \begin{bmatrix} \Delta y \\ \Delta x \\ \Delta \tau \end{bmatrix} = \begin{bmatrix} q_p \\ q_d \\ q_g \end{bmatrix}$$

by first solving two simpler systems

$$\begin{bmatrix} 0 & A \\ -A^T & X^{-1}S \end{bmatrix} \begin{bmatrix} \Delta y_1 & \Delta y_2 \\ \Delta x_1 & \Delta x_2 \end{bmatrix} = \begin{bmatrix} -b & q_p \\ c & q_d \end{bmatrix}.$$

Then

$$\Delta \tau = \frac{q_g - b^T \Delta y_2 + c^T \Delta x_2}{\kappa/\tau - b^T \Delta y_1 + c^T \Delta x_1},$$

and

$$(\Delta x, \Delta y) = (\Delta x_2, \Delta y_2) - \Delta \tau (\Delta x_1, \Delta y_1).$$

Nesterov-Todd search-directions for symmetric cones

The Nesterov-Todd scaling point w defines a primal-dual mapping,

$$F''(w)x = s, \qquad F''(w)F_*'(s) = F'(x),$$

where $F(x)$ and $F_*(s)$ are primal and dual barrier functions.

For $F''(w) = W^T W$, equivalent centrality conditions are

$$Wx \circ W^{-T}s = \mu e,$$

and linearization gives the (affine) Nesterov-Todd search direction

$$G(\Delta z_{\text{aff}}) = -G(z),$$
$$W\Delta x_{\text{aff}} + W^{-T}\Delta s_{\text{aff}} = -W^{-T}s, \quad \kappa\Delta\tau_{\text{aff}} + \tau\Delta\kappa_{\text{aff}} = -\tau\kappa.$$

Pioneered by SeDuMi and SDPT3, and basis of all commercial second-order cone solvers.

Simplex vs Barrier

Simplex Method	Barrier Method
Suitable for	
Very sparse LP problems	Dense LP problems
Huge scale LP problems	Degenerated LP problems
MIP reoptimization	Numerically hard problems
Implementation techniques	
Presolving	Remove linear dependencies
LU factorization and update	Cholesky decomposition
Exploiting hyper-sparsity	Crossover (basic solution)
Handling numerical issues	Efficient parallelization
Handling degeneracy	Handling infeasibility
Our highlights	
Parallel (30% speedup)	#threads-independent
Quad-precision support	Smart crossover

The primal-dual hybrid gradient method

A first order splitting method for solving problems[1]

$$\begin{array}{ll} \text{minimize}_{x \in X} & f(x) \\ \text{subject to} & Ax = b, \end{array}$$

where f is a closed convex function. The Lagrange function:

$$L(x, y, z) = f(x) - y^T(Ax - b).$$

Then the dual problem becomes

$$\text{maximize} - f^*(A^T y) + b^T y,$$

with

$$f^*(u) := \sup_x \{u^T x - f(x)\}.$$

[1]Special case of $\min_{x \in X} f(x) + g(Ax)$.

The primal-dual hybrid gradient method

Solution is a saddle-point of

$$\min_{x \in X} \max_{y \in \mathbf{R}^m} L(x, y) = f(x) - y^T A x + b^T y,$$

and the optimality conditions are

$$A^T y \in \partial f(x), \qquad Ax = b$$

where $\partial f(x)$ is the subdifferential,

$$\partial f(x) := \{g \mid f(y) - f(x) \geqslant g^T (y - x), \forall y \in \mathrm{dom}(f)\}.$$

Update equations:

$$\begin{aligned} x_{k+1} &= \mathrm{prox}_{\tau f}(x_k + \tau A^T y_k) \\ y_{k+1} &= y_k + \sigma(b - A(2x_{k+1} - x_k)). \end{aligned}$$

where τ and κ are fixed step-sizes. Converges if $\tau\sigma\|A\|_2^2 \leqslant 1$.

PDHG for linear programming

$$f(x) := c^T x + \delta_C(x)$$

where δ_C is the indicator for $C = \mathbf{R}_+$. Then

$$\mathrm{prox}_{\tau f}(u) := \arg\min_{x \in C} \{\tau c^T x + (1/2)\|u - x\|^2\} = P_C(u - \tau c)$$

resulting in update equations

$$\begin{aligned} x_{k+1} &= P_C(x_k - \tau(c - A^T y_k)), \\ y_{k+1} &= y_k + \sigma(b - A(2x_{k+1} - x_k)). \end{aligned}$$

Since $\partial f(x) = c + N_c(x)$, where

$$N_c(x) = \{d \mid d^T(z - x) \leqslant 0, \forall z \in C\}$$

is the normal-cone, we get the usual optimality conditions

$$c - A^T y = s, \quad s \geqslant 0, \quad x^T s = 0, \quad Ax = b.$$

PDHG - cuPDLP-C

cuPDLP-C: a tuned GPU implementation in C (open-source).

https://github.com/COPT-Public/cuPDLP-C

- Developed by Cardinal Operations and Haihao Lu's team.
- Highly optimized.
- Can solve certain large sparse problem much faster then either simplex or primal-dual methods. Solves `zib03` to 10^{-6} accuracy in 15 minutes using an NVIDIA H100 card. In contrast, the COPT barrier solver spends about 20 hours.

Simplex vs Barrier vs PDLP - a computational comparison

Instances	Rows	Columns	Non-Zeros
Open pit mining problems			
rmine11	97389	12292	241240
rmine13	197155	23980	485784
rmine15	358395	42438	879732
rmine21	1441651	162547	3514884
rmine25	2953849	326599	7182744
Satellite schedule problems			
satellites2-25	20916	35378	283668
satellites2-40	20916	35378	283668
satellites2-60-fs	16516	35378	125048
satellites3-25	44804	81681	698176
satellites4-25	51712	95637	821192
Unit commitment problems			
uccase7	47132	33020	335644
uccase8	53709	37413	214625
uccase9	49565	33242	332316
uccase10	196498	110818	787045
uccase12	121161	62529	419447

Table: Group of MIPLIB 2017 instances where root LP is non-trivial.

Simplex vs Barrier vs PDLP - a computational comparison

Instances	Simplex	Barrier	PDLP(CPU)	PDLP(GPU)
rmine11	3.82	2.07	41.19	4.04
rmine13	6.85	4.36	56.61	7.70
rmine15	28.07	13.08	137.99	8.30
rmine21	848.69	187.13	1531.02	111.71
rmine25	> 3600.00	1600.66	> 3600.00	681.90
satellites2-25	36.87	5.18	27.24	3.80
satellites2-40	33.90	4.86	20.61	3.51
satellites2-60-fs	4.23	0.68	30.23	6.87
satellites3-25	91.97	27.45	71.48	4.72
satellites4-25	180.60	34.68	87.77	7.64
uccase7	5.91	1.49	103.96	13.15
uccase8	1.63	1.10	9.34	2.65
uccase9	2.99	1.64	72.78	16.30
uccase10	1.30	1.21	4.15	12.16
uccase12	0.62	0.66	90.69	13.07

Table: COPT solution time comparison (seconds). Time limit: 3600s.
Setting: only LP presolving is performed, all solved to optimal basis.
Hardware: AMD 7900X with 128G and NVIDIA 4090 with 24G memory.

Active research areas

Interior-point methods.

- Theory and algorithms for non-symmetric cones.
- GPU implementations.
- Exploiting sparse structure in semidefinite optimization.

First-order methods.

- Theory and algorithms.
  Improve step-size and restart strategies.
- Further investigate alternative first-order methods.
- GPU implementations.

Try COPT - quick starts

Python

pip install coptpy

Julia

Pkg.add("COPT")

Trial license

- No need to apply for a license for non-commercial use.
- Within 10000 constraints and variables for LP.
- Within 2000 constraints and variables for other problem types.

Online documentation:

https://guide.coap.online/copt/en-doc/

Applegate, David, Oliver Hinder, Haihao Lu, and Miles Lubin. "Faster first-order primal-dual methods for linear programming using restarts and sharpness". In: *Mathematical Programming* 201.1 (2023), pp. 133–184.

Chambolle, Antonin and Thomas Pock. "A first-order primal-dual algorithm for convex problems with applications to imaging". In: *Journal of mathematical imaging and vision* 40 (2011), pp. 120–145.

Sturm, Jos F. "Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones". In: *Optimization methods and software* 11.1-4 (1999), pp. 625–653.

Vandenberghe, Lieven. *Lecture notes for EE236C - Optimization Methods for Large-Scale Systems*.

Wright, Stephen J. *Primal-dual interior-point methods*. SIAM, 1997.