First-Order Algorithms for CLP

Yinyu Ye
Department of Management Science and Engineering
Stanford University
Stanford, CA 94305, U.S.A.

http://www.stanford.edu/~yyye

(LY: Chapters 5.5, 8.2, 8.5, 12.1)

First-Order Method/Value-Iteration for MDP/RL I

In contrast to the second-order methods such as Newton's method, the first-order methods are typically using just matrix-vector multiplications in each step (e.g., in evaluating the gradient method).

Recall the Fixed-Point Model:

$$y_i = \min_{j \in \mathcal{A}_i} \{c_j + \gamma \mathbf{p}_j^T \mathbf{y}\}, \ \forall i$$

and the LP formulation

$$\begin{aligned} \text{maximize}_{\mathbf{y}} \quad & \sum_{i=1}^m y_i \\ \text{subject to} \quad & y_i - \gamma \mathbf{p}_j^T \mathbf{y} \leq c_j, \ j \in \mathcal{A}_i, \ \forall i. \end{aligned}$$

First-Order Method/Value-Iteration for MDP/RL II

The Value-Iteration (VI) Method: starting from any $\mathbf{y}^0$,

$$y_i^{k+1} = \min_{j \in \mathcal{A}_i} \{c_j + \gamma \mathbf{p}_j^T \mathbf{y}^k\}, \ \forall i.$$

Contraction:

$$\|\mathbf{y}^{k+1} - \mathbf{y}^*\|_\infty \leq \gamma \|\mathbf{y}^k - \mathbf{y}^*\|_\infty, \ \forall k.$$

where $\mathbf{y}^*$ is the fixed-point or optimal value vector, that is,

$$y_i^* = \min_{j \in \mathcal{A}_i} \{c_j + \gamma \mathbf{p}_j^T \mathbf{y}^*\}, \ \forall i.$$

Monotonicity: If we start from a vector such that

$$y_i^0 < \min_{j \in \mathcal{A}_i} \{c_j + \gamma \mathbf{p}_j^T \mathbf{y}^0\}, \ \forall i$$

($\mathbf{y}^0$ in the interior of the feasible region), then

$$\mathbf{y}^* \geq \mathbf{y}^{k+1} \geq \mathbf{y}^k, \ \forall k.$$

Dimension Reduction for MDP/RL III

Target Action Sampling: select important actions to update cost-to-go values during the VI process?

Online State-Aggregation: group states into a single "super"-state with similar cost-to-go values during the VI process?

First-Order Method/Value-Iteration for MDP/RL IV

One can apply the barrier function to solving the MDP problem, that is, to maximize the berried objective for a fixed μ as unconstrained optimization:

$$\max_{\mathbf{y}} \quad b_\mu(\mathbf{y}) = \mathbf{b}^T\mathbf{y} + \mu \sum_j \log(c_j - \mathbf{a}_j^T\mathbf{y}).$$

Starting an initial interior-feasible solution $\mathbf{y}^0$, apply the steepest-ascent algorithm to maximize the objective.

After the gradient values become “small”, decrease μ by a fixed factor and start the steepest ascent again – First-Order Path-Following.

One may also directly apply the First-Order Potential-Reduction.

First-Order Algorithm: the Steepest Descent Method (SDM)

Let f be a differentiable function and assume we can compute gradient (column) vector ∇f . We want to solve the unconstrained minimization problem

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

In the absence of further information, we seek a first-order KKT or stationary point of f, that is, a point $\mathbf{x}^*$ at which $\nabla f(\mathbf{x}^*) = \mathbf{0}$. Here we choose direction vector $\mathbf{d}^k = -\nabla f(\mathbf{x}^k)$ as the search direction at $\mathbf{x}^k$, which is the direction of steepest descent.

The number $\alpha^k \geq 0$, called step-size, is chosen "appropriately" as

$$\alpha^k \in \arg\min f(\mathbf{x}^k - \alpha\nabla f(\mathbf{x}^k)).$$

Then the new iterate is defined as $\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha^k\nabla f(\mathbf{x}^k)$.

In some implementations, step-size α^k is fixed through out the process – independent of iteration count k

Step-Size of the SDM for Minimizing Lipschitz Functions

Let $f(\mathbf{x})$ be differentiable every where and satisfy the (first-order) β-Lipschitz condition, that is, for any two points $\mathbf{x}$ and $\mathbf{y}$

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \leq \beta\|\mathbf{x} - \mathbf{y}\| \tag{1}$$

for a positive real constant β. Then, we have

Lemma 1 *Let f be a β-Lipschitz function. Then for any two points $\mathbf{x}$ and $\mathbf{y}$*

$$f(\mathbf{x}) - f(\mathbf{y}) - \nabla f(\mathbf{y})^T(\mathbf{x} - \mathbf{y}) \leq \frac{\beta}{2}\|\mathbf{x} - \mathbf{y}\|^2. \tag{2}$$

At the kth step of SDM, we have

$$f(\mathbf{x}) - f(\mathbf{x}^k) \leq \nabla f(\mathbf{x}^k)^T(\mathbf{x} - \mathbf{x}^k) + \frac{\beta}{2}\|\mathbf{x} - \mathbf{x}^k\|^2.$$

The left hand strict convex quadratic function of $\mathbf{x}$ establishes a upper bound on the objective reduction.

Let us minimize the quadratic function

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \; \nabla f(\mathbf{x}^k)^T(\mathbf{x} - \mathbf{x}^k) + \frac{\beta}{2}\|\mathbf{x} - \mathbf{x}^k\|^2,$$

and let the minimizer be the next iterate. Then it has a close form:

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \frac{1}{\beta}\nabla f(\mathbf{x}^k)$$

which is the SDM with the fixed step-size $\frac{1}{\beta}$. Then

$$f(\mathbf{x}^{k+1}) - f(\mathbf{x}^k) \le -\frac{1}{2\beta}\|\nabla f(\mathbf{x}^k)\|^2, \quad \text{or} \quad f(\mathbf{x}^k) - f(\mathbf{x}^{k+1}) \ge \frac{1}{2\beta}\|\nabla f(\mathbf{x}^k)\|^2.$$

Then, after $K(\ge 1)$ steps, we must have

$$f(\mathbf{x}^0) - f(\mathbf{x}^K) \ge \frac{1}{2\beta}\sum_{k=0}^{K-1}\|\nabla f(\mathbf{x}^k)\|^2. \tag{3}$$

Theorem 1 *(Error Convergence Estimate Theorem) Let the objective function* $p^* = \inf \; f(\mathbf{x})$ *be finite and let us stop the SDM as soon as* $\|\nabla f(\mathbf{x}^k)\| \le \epsilon$ *for a given tolerance* $\epsilon \in (0\;1)$*. Then the SDM* *terminates in* $\frac{2\beta(f(\mathbf{x}^0)-p^*)}{\epsilon^2}$ *steps.*

Proof: From (3), after $K = \frac{2\beta(f(\mathbf{x}^0)-p^*)}{\epsilon^2}$ steps

$$f(\mathbf{x}^0) - p^* \geq f(\mathbf{x}^0) - f(\mathbf{x}^K) \geq \frac{1}{2\beta} \sum_{k=0}^{K-1} \|\nabla f(\mathbf{x}^k)\|^2.$$

If $\|\nabla f(\mathbf{x}^k)\| > \epsilon$ for all $k = 0, ..., K-1$, then we have

$$f(\mathbf{x}^0) - p^* > \frac{K}{2\beta}\epsilon^2 \geq f(\mathbf{x}^0) - p^*$$

which is a contradiction.

Corollary 1 *If a minimizer* $\mathbf{x}^*$ *of* f *is attainable, then the SDM terminates in* $\frac{\beta^2\|\mathbf{x}^0-\mathbf{x}^*\|^2}{\epsilon^2}$ *steps.*

The proof is based on Lemma 1 with $\mathbf{x} = \mathbf{x}^0$ and $\mathbf{y} = \mathbf{x}^*$ and noting $\nabla f(\mathbf{y}) = \nabla f(\mathbf{x}^*) = \mathbf{0}$:

$$f(\mathbf{x}^0) - p^* = f(\mathbf{x}^0) - f(\mathbf{x}^*) \leq \frac{\beta}{2}\|\mathbf{x}^0 - \mathbf{x}^*\|^2.$$

Forward and Backward Tracking Step-Size Method

In most real applications, the Lipschitz constant β is unknown. Furthermore, we like to use a smaller and localized Lipschitz constant β^k, assuming it is bounded away from 0, at iteration k such that the inequality

$$f(\mathbf{x}^k + \alpha \mathbf{d}^k) - f(\mathbf{x}^k) - \nabla f(\mathbf{x}^k)^T(\alpha \mathbf{d}^k) \le \frac{\beta^k}{2}\|\alpha \mathbf{d}^k\|^2$$

holds, where $\mathbf{d}^k = -\nabla f(\mathbf{x}^k)$, to decide the step-size $\alpha = \frac{1}{\beta^k}$.

Consider the following step-size strategy: stat at a good step-size guess $\alpha > 0$:

(1): If $\alpha \le \frac{2(f(\mathbf{x}^k) - f(\mathbf{x}^k + \alpha \mathbf{d}^k))}{\|\mathbf{d}^k\|^2}$ then doubling the step-size: $\alpha \leftarrow 2\alpha$, stop as soon as the inequality is reversed and select the latest α with $\alpha \le \frac{2(f(\mathbf{x}^k) - f(\mathbf{x}^k + \alpha \mathbf{d}^k))}{\|\mathbf{d}^k\|^2}$;

(2): Otherwise halving the step-size: $\alpha \leftarrow \alpha/2$; stop as soon as $\alpha \le \frac{2(f(\mathbf{x}^k) - f(\mathbf{x}^k + \alpha \mathbf{d}^k))}{\|\mathbf{d}^k\|^2}$ and return it.

Prove that the selected step-size

$$\alpha \ge \frac{1}{2\beta^k}.$$

First-Order Algorithms for Conic Constrained Optimization (CCO)

Consider the conic nonlinear optimization problem: $\min \ f(\mathbf{x}) \quad \text{s.t.} \quad \mathbf{x} \in K.$

- Nonnegative Linear Regression: given data $A \in R^{m \times n}$ and $\mathbf{b} \in R^m$

$$\min \ f(\mathbf{x}) = \frac{1}{2}\|A\mathbf{x} - \mathbf{b}\|^2 \text{ s.t. } \mathbf{x} \geq \mathbf{0}; \quad \text{where } \nabla f(\mathbf{x}) = A^T(A\mathbf{x} - \mathbf{b}).$$

- Semidefinite Linear Regression: given data $A_i \in S^n$ for $i = 1, ..., m$ and $\mathbf{b} \in R^m$

$$\min \ f(X) = \frac{1}{2}\|\mathcal{A}X - \mathbf{b}\|^2 \text{ s.t. } X \succeq \mathbf{0}; \quad \text{where } \nabla f(X) = \mathcal{A}^T(\mathcal{A}X - \mathbf{b}).$$

$$\mathcal{A}X = \begin{pmatrix} A_1 \bullet X \\ \dots \\ A_m \bullet X \end{pmatrix} \quad \text{and} \quad \mathcal{A}^T\mathbf{y} = \sum_{i=1} y_i A_i.$$

Suppose we start from a feasible solution $\mathbf{x}^0$ or X^0.

Descent-First and Feasible-Second I

- $\hat{\mathbf{x}}^{k+1} = \mathbf{x}^k - \frac{1}{\beta}\nabla f(\mathbf{x}^k)$
- $\mathbf{x}^{k+1} = \text{Proj}_K(\hat{\mathbf{x}}^{k+1})$: Solve $\min_{\mathbf{x}\in K} \ \|\mathbf{x} - \hat{\mathbf{x}}^{k+1}\|^2$.

For examples:

- if $K = \{\mathbf{x} : \ \mathbf{x} \geq \mathbf{0}\}$, then

$$\mathbf{x}^{k+1} = \text{Proj}_K(\hat{\mathbf{x}}^{k+1}) = \max\{\mathbf{0}, \ \hat{\mathbf{x}}^{k+1}\}.$$

- If $K = \{X : \ X \succeq \mathbf{0}\}$, then factorize $\hat{X}^{k+1} = \sum_{j=1}^{n} \lambda_j \mathbf{v}_j \mathbf{v}_j^T$ and let

$$X^{k+1} = \text{Proj}_K(\hat{X}^{k+1}) = \sum_{j:\lambda_j>0} \lambda_j \mathbf{v}_j \mathbf{v}_j^T.$$

(The drawback is that the total eigenvalue-factorization may be costly...)

Does the method converge? What is the convergence speed?

Descent-First and Feasible-Second II

Consider the conic nonlinear optimization problem: $\min \ f(\mathbf{x}) \quad \text{s.t.} \quad A\mathbf{x} = \mathbf{b}.$ that is $K = \{\mathbf{x} : \ A\mathbf{x} = \mathbf{b}\}$.

The projection method becomes, starting from a feasible solution $\mathbf{x}^0$ and let direction

$$\mathbf{d}^k = -(I - A^T(AA^T)^{-1}A)\nabla f(\mathbf{x}^k)$$

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k \mathbf{d}^k; \tag{4}$$

where the stepsize can be chosen from line-search or again simply let

$$\alpha^k = \frac{1}{\beta}$$

and β is the (global) Lipschitz constant.

Does the method converge? What is the convergence speed? See more details in HW3.

Descent-First and Feasible-Second III

- $K \subset R^n$ whose support size is no more than $d(< n)$: $\mathbf{x} = \text{Proj}_K(\hat{\mathbf{x}})$ contains the largest d absolute entries of $\hat{\mathbf{x}}$ and set the rest of them to zeros.
- $K \subset R^n_+$ and its support size is no more than $d(< n)$: $\mathbf{x} = \text{Proj}_K(\hat{\mathbf{x}})$ contains the largest no more than d positive entries of $\hat{\mathbf{x}}$ and set the rest of them to zeros.
- $K \subset S^n$ whose rank is no more than $d(< n)$: factorize $\hat{X} = \sum_{j=1}^n \lambda_j \mathbf{v}_j \mathbf{v}_j^T$ with $|\lambda_1| \geq |\lambda_2| \geq ... \geq |\lambda_n|$ then $\text{Proj}_K(\hat{X}) = \sum_{j=1}^d \lambda_j \mathbf{v}_j \mathbf{v}_j^T$.
- $K \subset S^n_+$ whose rank is no more than $d(< n)$: factorize $\hat{X} = \sum_{j=1}^n \lambda_j \mathbf{v}_j \mathbf{v}_j^T$ with $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n$ then $\text{Proj}_K(\hat{X}) = \sum_{j=1}^d \max\{0, \lambda_j\} \mathbf{v}_j \mathbf{v}_j^T$.

Does the method converge? What is the convergence speed? What if $f(\cdot)$ is not a convex function?

Multiplicative-Update I: "Mirror" SDM for CCO

At the kth iterate with $\mathbf{x}^k > \mathbf{0}$:

$$\mathbf{x}^{k+1} = \mathbf{x}^k .* \exp(-\frac{1}{\beta}\nabla f(\mathbf{x}^k))$$

Note that $\mathbf{x}^{k+1}$ remains positive in the updating process.

The classical Projected SDM update can be viewed as

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}\geq\mathbf{0}} \nabla f(\mathbf{x}^k)^T\mathbf{x} + \frac{\beta}{2}\|\mathbf{x}-\mathbf{x}^k\|^2.$$

One can choose any strongly convex function $h(\cdot)$ and define

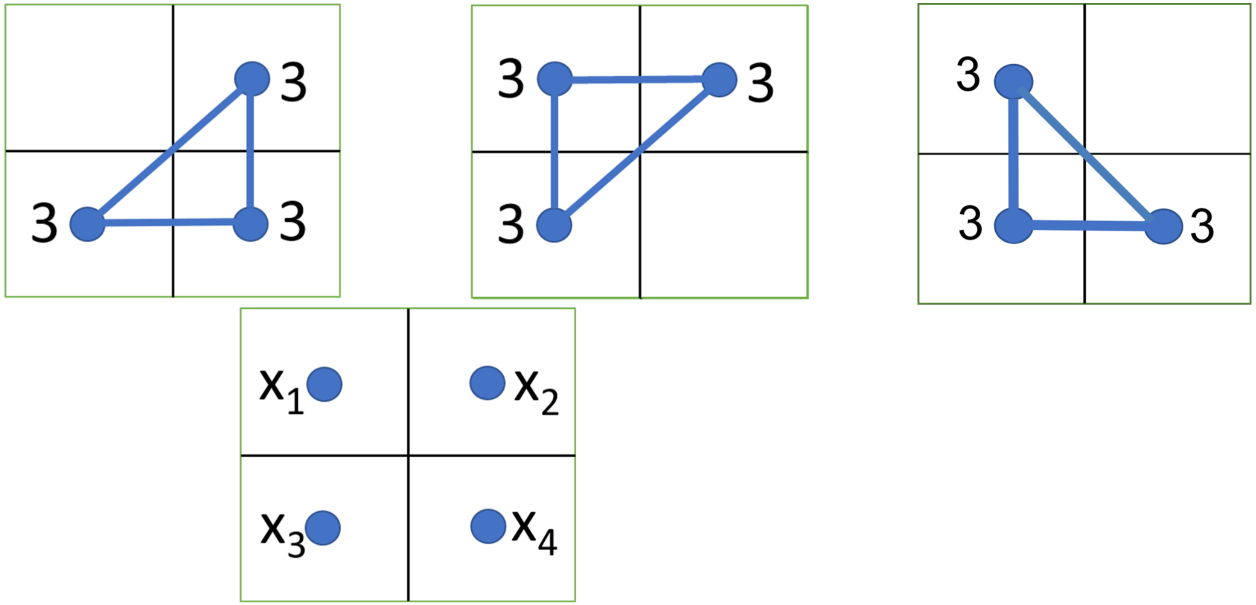
$$\mathcal{D}_h(\mathbf{x},\mathbf{y}) = h(\mathbf{x}) - h(\mathbf{y}) - \nabla h(\mathbf{y})^T(\mathbf{x}-\mathbf{y})$$

and define the update as

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}\geq\mathbf{0}} \nabla f(\mathbf{x}^k)^T\mathbf{x} + \beta\mathcal{D}_h(\mathbf{x},\mathbf{x}^k).$$

The update above is the result of choosing (negative) entropy function $h(\mathbf{x}) = \sum_j x_j\log(x_j)$.

The Wassestein Barycenter Problem



Find distribution of $x_i, i = 1, 2, 3, 4$ to minimize

$$\begin{array}{ll} \min & WD_l(\mathbf{x}) + WD_m(\mathbf{x}) + WD_r(\mathbf{x}) \\ \text{s.t.} & x_1 + x_2 + x_3 + x_4 = 9, \qquad x_i \geq 0,\ i = 1, 2, 3, 4. \end{array}$$

The objective is a nonlinear function, but its gradient vector $\nabla WD_l(\mathbf{x})$, $\nabla WD_m(\mathbf{x})$ and $\nabla WD_l(\mathbf{x})$ are shadow prices of the three sub-transportation problems –popularly used in Hierarchy Optimization.

(Projects #4 on WBC)

Multiplicative-Update II: Affine Scaling SDM for CCO

At the kth iterate with $\mathbf{x}^k > \mathbf{0}$, let D^k be a diagonal matrix such that

$$D^k_{jj} = x^k_j, \ \forall j$$

and

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x} \geq \mathbf{0}} \ \nabla f(\mathbf{x}^k)^T \mathbf{x} + \frac{\beta}{2} \|(D^k)^{-1}(\mathbf{x} - \mathbf{x}^k)\|^2,$$

or

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha_k (D^k)^2 \nabla f(\mathbf{x}^k) = \mathbf{x}^k. * (\mathbf{e} - \alpha_k \nabla f(\mathbf{x}^k). * \mathbf{x}^k)$$

where variable step-sizes can be

$$\alpha^k = \min\{\frac{1}{\beta \max(\mathbf{x}^k)^2}, \ \frac{1}{2\|\mathbf{x}^k. * \nabla f(\mathbf{x}^k)\|_\infty}\}.$$

Is $\mathbf{x}^k > \mathbf{0}, \ \forall k$? Does it converge? What is the convergence speed? See more details in HW.

Geometric Interpretation: inscribed ball vs inscribed ellipsoid.

Affine Scaling for SDP Cone?

At the kth iterate with $X^k \succ \mathbf{0}$. the new SDM iterate would be

$$X^{k+1} = X^k - \alpha_k X^k \nabla f(X^k) X^k = X^k(I - \alpha_k \nabla f(X^k) X^k).$$

Choose step-size is chosen such that the smallest eigenvalue of X^{k+1} is at most a fraction from the one of X^k?

Does it converge? What is the convergence speed?

First-Order Potential Reduction for Linear Least-Squares

Let us solve

$$\begin{array}{ll} \min & \|A\mathbf{x}-\mathbf{b}\|^2 \\ \text{s.t.} & \mathbf{x} \ge \mathbf{0} \end{array}$$

Consider the potential function

$$\psi_{n+\rho}(\mathbf{x}) := (n+\rho)\log(\|A\mathbf{x}-\mathbf{b}\|^2) - \sum_{j=1}^{n} \log(x_j).$$

Starting from an interior-point solution $\mathbf{x}) > \mathbf{0}$, we apply the SDM method to minimize the potential function.

Can use the preconditioning to improve the performance.

First-Order Potential Reduction for LP I

Recall that the joint primal-dual potential function is defined by

$$\psi_{n+\rho}(\mathbf{x},\mathbf{s}) := (n+\rho)\log(\mathbf{x}^T\mathbf{s}) - \sum_{j=1}^{n}\log(x_j s_j).$$

At the kth iteration, we compute the direction vectors $(\mathbf{d}_x, \mathbf{d}_y, \mathbf{d}_s)$ using the steepest descent direction:

$$\begin{array}{lll} \min & \nabla_x\phi(\mathbf{x}^k,\mathbf{s}^k)^T\mathbf{d}_x + \nabla_s\phi(\mathbf{x}^k,\mathbf{s}^k)^T\mathbf{d}_s & \\ \text{s.t.} & A\mathbf{d}_x & = \mathbf{0} \\ & A^T\mathbf{d}_y + \mathbf{d}_s & = \mathbf{0}, \end{array}$$

where

$$\nabla_x\phi(\mathbf{x}^k,\mathbf{s}^k)^T = \frac{n+\rho}{(\mathbf{x}^k)^T\mathbf{s}^k}\mathbf{s}^k - (X^k)^{-1}\mathbf{e}$$

and

$$\nabla_s\phi(\mathbf{x}^k,\mathbf{s}^k)^T = \frac{n+\rho}{(\mathbf{x}^k)^T\mathbf{s}^k}\mathbf{x}^k - (S^k)^{-1}\mathbf{e}.$$

First-Order Potential Reduction for LP II

More precisely, we have

$$\begin{aligned}
\mathbf{d}_x &= -(I - A^T(AA^T)^{-1}A)\nabla_x \phi(\mathbf{x}^k, \mathbf{s}^k), \\
\mathbf{d}_y &= A\nabla_s \phi(\mathbf{x}^k, \mathbf{s}^k), \\
\mathbf{d}_s &= -A^T A\nabla_s \phi(\mathbf{x}^k, \mathbf{s}^k).
\end{aligned}$$

Then, we let

$$\begin{aligned}
\mathbf{x}^{k+1} &= \mathbf{x}^k + \alpha \mathbf{d}_x, \\
\mathbf{y}^{k+1} &= \mathbf{y}^k + \alpha \mathbf{d}_y, \\
\mathbf{s}^{k+1} &= \mathbf{s}^k + \alpha \mathbf{d}_s,
\end{aligned}$$

for some step-size α such that the potential value is minimized along the directions.

SDP Cone?

Alternating Primal-Dual Direction Method I

The joint primal-dual potential function can be written as

$$\begin{aligned}\psi_{n+\rho}(\mathbf{x},\mathbf{s}) &= (n+\rho)\log(\mathbf{x}^T\mathbf{s}) - \textstyle\sum_{j=1}^n \log(x_j s_j)\\ &= (n+\rho)\log(\mathbf{c}^T\mathbf{x}-\mathbf{b}^T\mathbf{y}) - \textstyle\sum_{j=1}^n \log(x_j) - \sum_{j=1}^n \log(s_j)\\ &= (n+\rho)\log(\mathbf{c}^T\mathbf{x}-\mathbf{b}^T\mathbf{y}) - \textstyle\sum_{j=1}^n \log(x_j) - \sum_{j=1}^n \log(c_j - \mathbf{a}_j^T\mathbf{y})\end{aligned}$$

since $\mathbf{s} = \mathbf{c} - A^T\mathbf{y}$. Then let

$$\phi(\mathbf{x}^k,\mathbf{y}^k) = (n+\rho)\log(\mathbf{c}^T\mathbf{x}-\mathbf{b}^T\mathbf{y}) - \sum_{j=1}^n \log(x_j) - \sum_{j=1}^n \log(c_j - \mathbf{a}_j^T\mathbf{y}),$$

$$\nabla_x\phi(\mathbf{x}^k,\mathbf{y}^k)^T = \frac{n+\rho}{\mathbf{c}^T x^k - \mathbf{b}^T\mathbf{y}^k}\mathbf{c} - (X^k)^{-1}\mathbf{e},$$

and

$$\nabla_y\phi(\mathbf{x}^k,\mathbf{y}^k)^T = -\frac{n+\rho}{\mathbf{c}^T x^k - \mathbf{b}^T\mathbf{y}^k}\mathbf{b} - A(S^k)^{-1}\mathbf{e}$$

Alternating Primal-Dual Direction Method II

At the kth iteration, we fix $(\mathbf{s}^k, \mathbf{y}^k)$ and compute an approximate minimizer as $\mathbf{x}^{k+1}$ using any iterative method starting from $\mathbf{x}^k$:

$$\begin{array}{lll} \min_{\mathbf{x}} & \phi(\mathbf{x}, \mathbf{y}^k) & \\ \text{s.t.} & A\mathbf{x} & = \mathbf{b}. \end{array}$$

One would reduce the potential function by a fixed amount after updating from $\mathbf{x}^k$ to $\mathbf{x}^{k+1}$ while keep $(\mathbf{y}^{k+1}, \mathbf{s}^{k+1}) = (\mathbf{y}^k, \mathbf{s}^k)$:

$$\psi_{n+\rho}(\mathbf{x}^{k+1}, \mathbf{s}^{k+1}) - \psi_{n+\rho}(\mathbf{x}^k, \mathbf{s}^k) \leq -\delta.$$

Then we update the dual iterate $(\mathbf{y}^k, \mathbf{s}^k)$, and do these updates alternatively.

Alternating Primal-Dual Direction Method III

When fix $\mathbf{x}^k$, we compute an approximate minimizer as $\mathbf{y}^{k+1}$ using any iterative method starting from $\mathbf{y}^k$:

$$\min_{\mathbf{y}} \quad \phi(\mathbf{x}^k, \mathbf{y})$$

which is an unconstrained minimization.

Again, one would reduce the potential function by a fixed amount after updating from $\mathbf{y}^k$ to $\mathbf{y}^{k+1}$ ($\mathbf{s}^{k+1} = \mathbf{c} - A^T\mathbf{y}^{k+1}$) while keep $\mathbf{x}^{k+1} = \mathbf{x}^k$:

$$\psi_{n+\rho}(\mathbf{x}^{k+1}, \mathbf{s}^{k+1}) - \psi_{n+\rho}(\mathbf{x}^k, \mathbf{s}^k) \le -\delta.$$

Many iterative methods can be considered: the Steepest Descent, Conjugate Gradient, Quasi-Newton, Stochastic Gradient, etc.