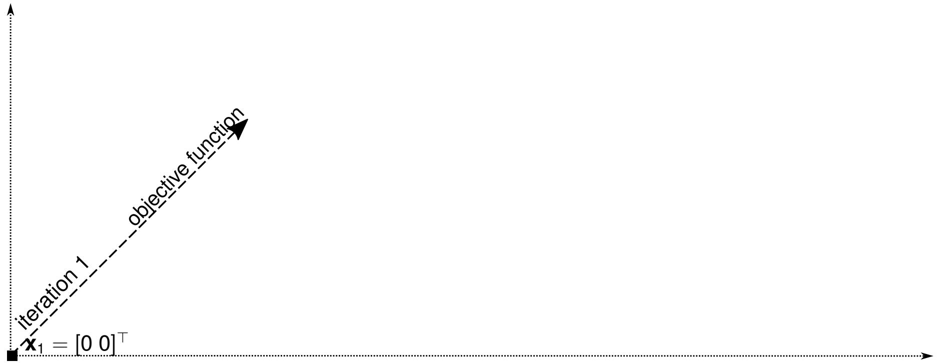
# Semidefinite programming by Projective-Cutting-Planes

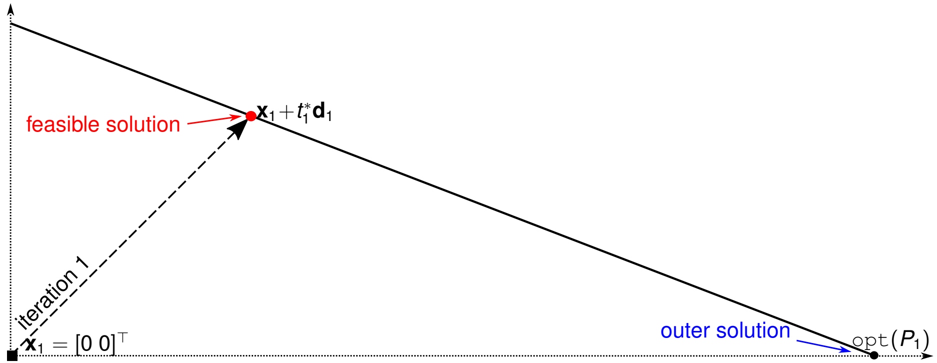
Daniel Porumbel

Conservatoire National des Arts et Métiers, Paris

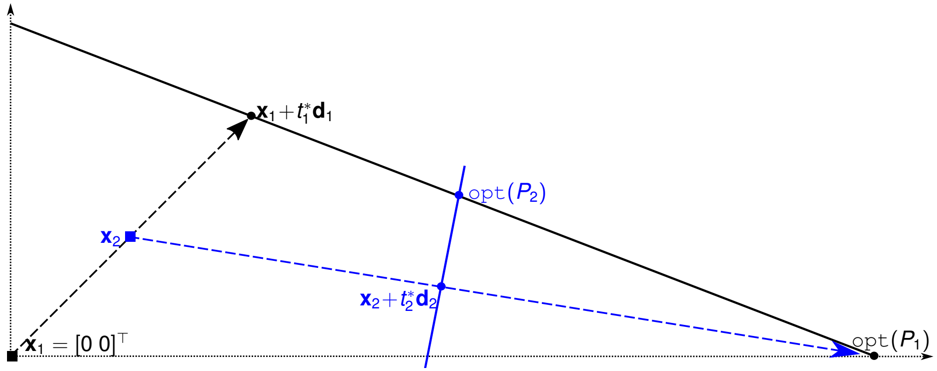
I first worked the idea of *Projective-Cutting-Planes* in a purely linear context



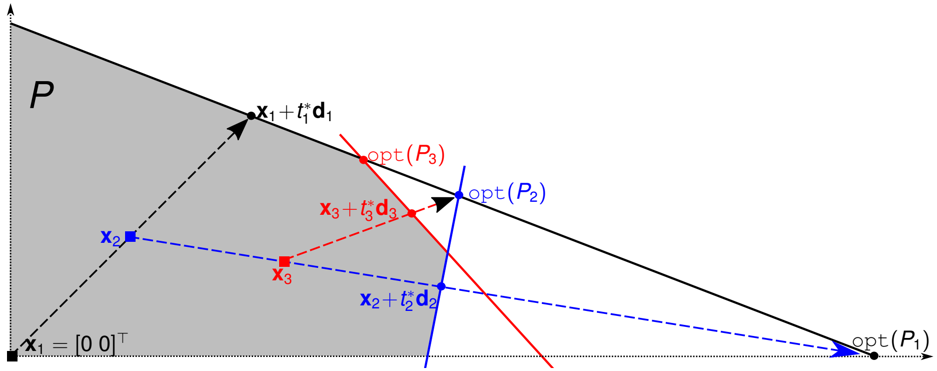
Iteration 1 : uncharted territory, follow objective function, i.e., advance along $\mathbf{x}_1 \rightarrow \mathbf{d}_1$ where $\mathbf{d}_1$ takes the value of the objective function


feasible solution
$\mathbf{x}_1 + t_1^*\mathbf{d}_1$
iteration 1
$\mathbf{x}_1 = [0\ 0]^\top$
outer solution
$\mathtt{opt}(P_1)$

Iteration 1 : found a first outer solution $\mathtt{opt}(P_1)$ and a first inner solution (contact point) $\mathbf{x}_1 + t_1^*\mathbf{d}_1$

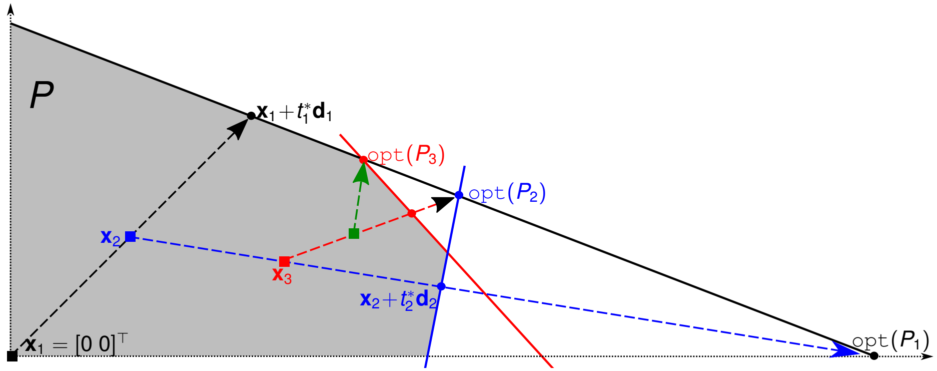

$\mathbf{x}_1 + t_1^*\mathbf{d}_1$
$\mathtt{opt}(P_2)$
$\mathbf{x}_2$
$\mathbf{x}_2 + t_2^*\mathbf{d}_2$
$\mathbf{x}_1 = [0\ 0]^\top$
$\mathtt{opt}(P_1)$

Iteration 2 : an inner feasible solution (contact point) $\mathbf{x}_2 + t_2^*\mathbf{d}_2$ and a new outer solution. We take $\mathbf{d}_2 = \mathtt{opt}(P_1) - \mathbf{x}_2$.



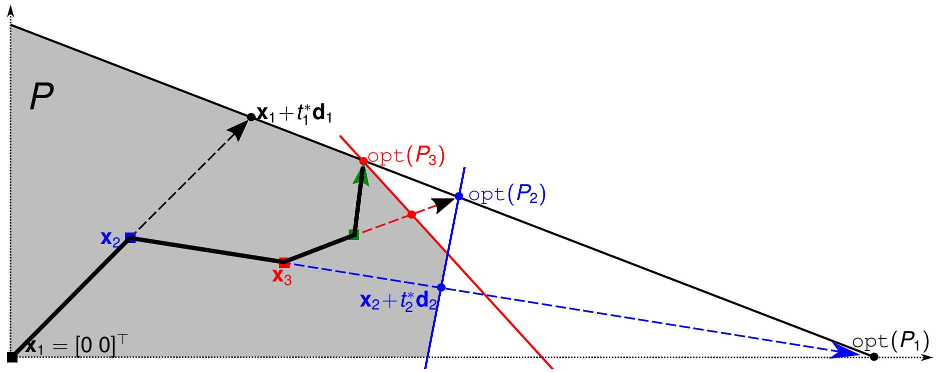
$P$
$\mathbf{x}_1 + t_1^*\mathbf{d}_1$
$\mathtt{opt}(P_3)$
$\mathtt{opt}(P_2)$
$\mathbf{x}_3 + t_3^*\mathbf{d}_3$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\mathbf{x}_2 + t_2^*\mathbf{d}_2$
$\mathbf{x}_1 = [0\ 0]^\top$
$\mathtt{opt}(P_1)$

Iteration 3 : the feasible solution $\mathbf{x}_3 + t_3^*\mathbf{d}_3$ is almost optimal


$P$
$\mathbf{x}_1 + t_1^* \mathbf{d}_1$
$\mathtt{opt}(P_3)$
$\mathtt{opt}(P_2)$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\mathbf{x}_2 + t_2^* \mathbf{d}_2$
$\mathbf{x}_1 = [0\ 0]^\top$
$\mathtt{opt}(P_1)$

Iteration 4 : optimality of $\mathtt{opt}(P_3)$ proved
You can see the proposed method is convergent because it solves a separation problem on $\mathtt{opt}(P_k)$ at each iteration $k$

- The convergence proof takes two lines, cool !



$P$
$\mathbf{x}_1 + t_1^* \mathbf{d}_1$
$\texttt{opt}(P_3)$
$\texttt{opt}(P_2)$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\mathbf{x}_2 + t_2^* \mathbf{d}_2$
$\mathbf{x}_1 = [0\ 0]^\top$
$\texttt{opt}(P_1)$

Building on existing work [1,2], the new method was deliberately designed to be more general and when possible simpler

[1] Daniel Porumbel. Ray projection for optimizing polytopes with prohibitively many constraints in set-covering column generation. *Mathematical Programming*, 155(1) :147–197, 2016.

[2] Daniel Porumbel. Daniel Porumbel. Projective Cutting-Planes, *SIAM Journal on Optimization*, 30(1) : 1007-1032, 2020

We now focus on the following standard (semidefinite programming) SDP problem, where $\boxed{\mathcal{A}^\top \mathbf{y} = \sum_{i=1}^k A_i y_i}$. The implication in the last constraint indicates that the SDP constraint can be expressed using an infinite number of linear cuts.

$$\mathcal{S}_0 \begin{cases} \max & \mathbf{b}^\top \mathbf{y} \\ s.t & X = C - \mathcal{A}^\top \mathbf{y} \\ & X \succeq \mathbf{0} \iff X \bullet \mathbf{s}\mathbf{s}^\top \geq 0 \ \forall \mathbf{s} \in \mathbb{R}^n \end{cases}$$

In some sense, the SDP cone is a polytope described by infinitely-many cuts.

$$\mathcal{S}_0 \begin{cases} \max & \mathbf{b}^\top \mathbf{y} \\ s.t & X = C - \mathcal{A}^\top \mathbf{y} \\ & X \succeq \mathbf{0} \iff X \bullet \mathbf{s}\mathbf{s}^\top \geq 0 \ \forall \mathbf{s} \in \mathbb{R}^n \end{cases}$$

We could address the problem by progressively separating infeasible solutions $\mathbf{y}_{\text{out}} \in \mathbb{R}^n$. This standard `Cutting-Planes` is rarely very effective (at least not in any of my tests).

In *Projective-Cutting-Planes*, we upgrade the separation sub-problem to the projection sub-problem: given feasible $\mathbf{y}$ in a feasible area $\mathcal{S}_0$ and an arbitrary direction $\mathbf{d}$, what is the maximum step-length $t^*$ so that $\mathbf{y} + t^*\mathbf{d} \in \mathcal{S}_0$?

$$\mathcal{S}_0 \begin{cases} \max & \mathbf{b}^\top \mathbf{y} \\ s.t & X = C - \mathcal{A}^\top \mathbf{y} \\ & X \succeq \mathbf{0} \iff X \bullet \mathbf{s}\mathbf{s}^\top \geq 0 \ \forall \mathbf{s} \in \mathbb{R}^n \end{cases}$$

We could address the problem by progressively separating infeasible solutions $\mathbf{y}_{\text{out}} \in \mathbb{R}^n$. This standard `Cutting-Planes` is rarely very effective (at least not in any of my tests).

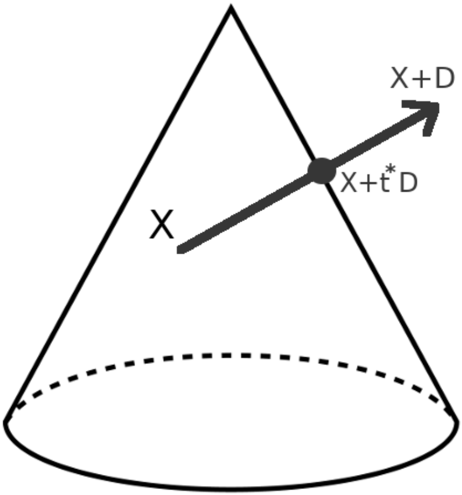
In *Projective-Cutting-Planes*, we upgrade the separation sub-problem to the projection sub-problem: given feasible $\mathbf{y}$ in a feasible area $\mathcal{S}_0$ and an arbitrary direction $\mathbf{d}$, what is the maximum step-length $t^*$ so that $\mathbf{y} + t^*\mathbf{d} \in \mathcal{S}_0$?

In SDP programming, projecting $\mathbf{y} \to \mathbf{d}$ requires solving $t^* = \max\{t : X + tD \succeq \mathbf{0}\}$ for this $X \succeq \mathbf{0}$ and $D$:

- $X = C - \mathcal{A}^\top \mathbf{y}$ is SDP when $\mathbf{y}$ is feasible
- $D = C - \mathcal{A}^\top \mathbf{d}$ may be SDP or not.

We have to project $X \to D$ in the SDP cone, *i.e.*, find:

$$t^* = \max\{t : X + tD \succeq \mathbf{0}\}$$



We have to project $X \to D$ in the SDP cone, *i.e.*, find:

$$t^* = \max\{t : X + tD \succeq \mathbf{0}\}$$

**An easy-to-implement approach**: notice $t^*$ is the generalized eigenvalue of $X$ and $-D$. The corresponding generalized eigenvector $\mathbf{v}$ satisfies $X\mathbf{v} = -t^* D\mathbf{v}$.

I did not have the time to fully investigate this approach, but this is the practical difficulty: we need a very particular generalized eigenvalue, namely, *the lowest real eigenvalue above 0*. With existing software, computing all eigenvalues or only the eigenvalues close to zero seems much too slow.

But there is a huge advantage: using this projection approach, the overall method can be very easily implemented.

---

The main challenge is the speed of the projection algorithm. It should be closer to computing Cholesky or the smallest eigenvalue of matrix. If the speed is closer to computing the whole eigendecomposition, all seems lost.

`And your speed it’s all you’ll ever need.`
`All you’ll ever need to know.`
`You and me we’re going nowhere slowly`
`You go down on the pedal and you’re ready to roll.`
`Meat Loaf - Nowhere slowly`

We have to project $X \to D$ in the SDP cone, *i.e.*, find:

$$t^* = \max\{t : X + tD \succeq \mathbf{0}\}$$

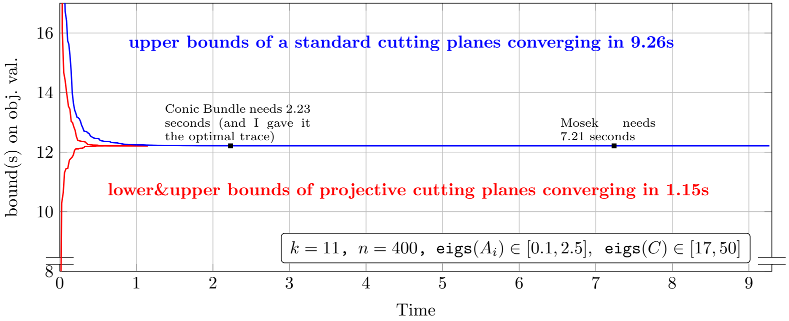
Th projection sub-problem is quite simple if $X \succ \mathbf{0}$. In this case, there is a unique Cholesky decomposition $X = KK^\top$ and $K$ is non-singular. We'll see later that the following projections are equivalent, using $D' =$ the unique solution of $D = KD'K^\top$.

- $X \to D$;
- $I_n \to D'$
  - we can determine $\max\{t : I_n + tD' \succeq \mathbf{0}\}$ by computing $\lambda_{\min}(D')$

Thus, the projection computational cost is: one Cholesky $\oplus$ one minimum eigen-value.

This projection is more difficult if $X$ is not strictly SDP. Yet, the simplified pure SDP case enabled us to solve some particular instances very rapidly.

Instance below was solved by advancing to the optimum trough a sequence of strictly SDP matrices (strictly interior points): $X_0 = \epsilon I_n, X_1, X_2, X_3 \dots$



*Property 1.* We will see that the projection $X \to D$ can be calculated more rapidly if $D$ belongs to the image of $X$. This means that each column (and row) of $D$ can be written as a linear combination of the columns (or rows, resp.) of $X$. We can equivalently say that the null space of $X$ is included in the null space of $D$; thus, $X\mathbf{d} = 0 \implies D\mathbf{d} = 0 \; \forall \mathbf{d} \in \mathbb{R}^n$. We will show below in cases A) and B) how it is easier to project when this property holds; if possible, `Projective Cutting-Planes` should thus adapt its own evolution to seek this property.

A) This case is characterized by $X \succ \mathbf{0}$, *i.e.*, $X$ is non-singular; Prop 1 surely holds because the image of a non-singular $X$ is $\mathbb{R}^n$. We apply the Cholesky decomposition to determine the unique non-singular $K$ such that $X = KK^\top$. We then solve $D = KD'K^\top$ in variables $D'$ by back substitution; this may require $O(n^3)$ in theory, but `Matlab` is able to compute it much more rapidly in practice because $K$ is triangular. Let us re-write (3) as:

$$\max\left\{t : KI_nK^\top + tKD'K^\top \succeq \mathbf{0}\right\}. \tag{4}$$

This is equivalent (by congruence according to Prop 2) to

$$\max\left\{t : I_n + tD' \succeq \mathbf{0}\right\}. \tag{5}$$

The sought step length is $t^* = -\frac{1}{\lambda_{\min}(D')}$, or $t^* = \infty$ if $\lambda_{\min}(D') \geq 0$.

We still have to find a first-hit cut $\mathbf{v} \in \mathbb{R}^n$; in fact, technically, the first-hit cut will be $\left(A_1 \bullet \mathbf{v}\mathbf{v}^\top\right) y_1 + \left(A_2 \bullet \mathbf{v}\mathbf{v}^\top\right) y_2 + \cdots + \left(A_k \bullet \mathbf{v}\mathbf{v}^\top\right) y_k \leq C \bullet \mathbf{v}\mathbf{v}^\top$.

If $\mathbf{v}$ is an eigenvector of $K(I_n + t^* D')K^\top$ with an eigenvalue of 0, this means $\mathbf{v}^\top K(I_n + t^* D')K^\top \mathbf{v} = 0$. Thus, $\mathbf{u} = K^\top \mathbf{v}$ is eigenvector of $I_n + t^* D'$ with an eigenvalue of 0. This latter eigenvector $\mathbf{u}$ can be computed when determining $\lambda_{\min}(D') < 0$ above, because if the eigenvalue of $\mathbf{u}$ with regards to $D'$ is $\lambda_{\min}(D')$ its eigenvalue with regards to $I_n + t^* D'$ is 0 (since recall $t^* = -\frac{1}{\lambda_{\min}(D')}$). The sought $\mathbf{v}$ solves $K^\top \mathbf{v} = \mathbf{u}$ and it can rapidly be computed by back-substitution. We have $\mathbf{u}^\top D' \mathbf{u} < 0 \implies \mathbf{v}^\top K D' K^\top \mathbf{v} < 0 \implies \mathbf{v}^\top D \mathbf{v} < 0$. We thus have $\mathbf{v}^\top (X + t^* D)\mathbf{v} = 0$ and $\mathbf{v}^\top (X + (t^* + \epsilon) D)\mathbf{v} < 0$ for any $\epsilon > 0$.

B) In this case Prop 1 is still satisfied, but $X$ has rank $c < n$. This means $X$ contains $c$ independent rows (and columns by symmetry), referred to as *core* rows (or columns); the other dependent rows (or columns) are *non-core* positions. Using the LDL decomposition of $X$, we will factorize $X = K_{nc}K_{nc}^\top$, where $K_{nc} \in \mathbb{R}^{n \times c}$. The image of $K_{nc}$ is equal to the image of $X$. Since Prop 1 is satisfied, we will see we can still solve $D = K_{nc}D'K_{nc}^\top$ in variables $D'$. A first intuition is to notice that we can project $X \to D$ only over the core rows and columns, because the non-core positions are dependent on the core ones.

But the most difficult task is to determine these core positions. We first apply the LDL decomposition and write $X = L\texttt{diag}(\mathbf{p})L^\top$ with $\mathbf{p} \geq \mathbf{0}_n$. The contribution of each $p_i$ in $L\texttt{diag}(\mathbf{p})L^\top$ is actually $p_i L_i L_i^\top$, where $L_i$ is column $i$ of $L$ ($\forall i \in [1..n]$). If all $n \times n$ elements of $p_i L_i L_i^\top$ are below some precision parameter, we consider $i$ is a non-core position; otherwise, it is a core position. By reducing all non-core positions $p_i$ to zero, we can say that all $n - c$ non core columns of $L$ vanish in the decomposition $X = L\texttt{diag}(\mathbf{p})L^\top$. After removing these vanished $n-c$ columns from $L$ and the corresponding zeros from $\mathbf{p}$, we can write $X = L\texttt{diag}(\mathbf{p})L^\top = L\texttt{diag}(\mathbf{p})^{\frac{1}{2}}\texttt{diag}(\mathbf{p})^{\frac{1}{2}}L^\top = K_{nc}K_{nc}^\top$ with $K_{nc} \in \mathbb{R}^{n\times c}$.

We next solve $D = K_{nc}D'K_{nc}^\top$ in variables $D'$. For this, we first reduce this system to work on $c \times c$ matrices, *i.e.*, we transform it into $D_{cc} = K_{cc}D'K_{cc}$ where $K_{cc}$ is $K_{nc}$ restricted to the $c$ core rows and $D_{cc}$ is $D$ restricted to the $c \times c$ core rows and columns. To solve this square system, we apply back-substitution twice and this is very fast because $K_{cc}$ is lower triangular. If the resulting solution $D'$ also satisfies $D = K_{nc}D'K_{nc}^\top$, then we are surely in case B). We obtained a reduced-size version of (5) working in the space of $c \times c$ matrices:

| Instance | | | | `Projective Cutting-Planes` | | | | | | `ConicBundle` | | `Mosek` |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $n$ | k | Eigs $A_i$'s | Eigs C | Itera-tions | All time | Compute $X$ & $D$ | Proj time | LP time (cplex) | Send data to LP | Trace unknown | Trace provided | |
| 800 | 80 | [-20, 100] | [0,100] | 1108 | 410 | 179 | 44 | 70 | 102 | 1051 | 94 | 320 |
| 600 | 40 | [-20, 100] | [0,100] | 155 | 17 | 4 | 6 | 1 | 3 | 148 | 22 | 72 |
| 400 | 100 | [-20, 100] | [0,100] | 2075 | 572 | 94 | 13 | 384 | 71 | 490 | 42 | 60 |
| Huge instances below have $\mathbf{y} \geq 0$, a random $\mathbf{b}$ and $\frac{n}{5}$ fixed null eigenvectors for all $A_i$'s and $C$ | | | | | | | | | | | | |
| 200 | 2000 | [40, 100] | [10,40] | 31 | 11 | 5 | 0.2 | 0.2 | 5 | timeout | | 717 |
| 200 | 3000 | [40, 100] | [10,40] | 70 | 49 | 27 | 0.4 | 0.7 | 18 | timeout | | 1346 |
| 4000 | 20 | [20,25] | [20,25] | 8 | 76 | 17 | 44 | 0 | 11 | timeout | | timeout |
| 5000 | 20 | [20,25] | [20,25] | 7 | 139 | 27 | 87 | 0 | 18 | timeout | | timeout |

**Table 2.** Seven runs of `Projective Cutting-Planes`, `ConicBundle` and `Mosek` on more varied instances. The last four instances have $\mathbf{y} \geq \mathbf{0}$; such linear constraints on $\mathbf{y}$ simplify the problem for `Projective Cutting-Planes`, but this may be a non-trivial change for `ConicBundle` (or other algorithms that do not embed the SDP problem in a lightweight LP over $\mathbf{y}$).