proxDykstra Toolbox

Necoara Ion
ion.necoara@acse.pub.ro

Bob Cristian
b.cristian.cb@gmail.com

2020

Contents

1 Introduction

The *proxDykstra* algorithm aims to solve **SOCP** problems of the following form:

$$\begin{aligned} \min_{x} \quad & c^T x \\ \text{s.t.} \quad & Ax = b \\ & \|Q_i x + q_i\| \leq f_i^T x + d_i \forall i = 1 : m \end{aligned} \tag{1}$$

$$A \in \mathbb{R}^{m_0 \times n}, Q_i \in \mathbb{R}^{m_i \times n}$$

In order to make the problem more compact, we take the following notations:

$$\begin{aligned} X &= \begin{bmatrix} x \\ x_0 \end{bmatrix}, \quad C = \begin{bmatrix} c \\ 0 \end{bmatrix}, \\ Q_0 &= \begin{bmatrix} A & 0 \\ 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} b \\ 0 \end{bmatrix}, \quad Q_i = \begin{bmatrix} Q_i & q_i \\ f_i & d_i \end{bmatrix}. \end{aligned}$$

Then the problem is rewritten in an equivalent form as:

$$\begin{aligned} \min_{x} \quad & C^T X \\ \text{s.t.} \quad & Q_0 x = B \\ & Q_i X \in K_i = \{ \begin{pmatrix} u_i \\ u_0 \end{pmatrix} \in \mathbb{R}^{m_i+1} \mid \|u_i\| \leq u_0 \} \quad \forall i = 1\text{:m} \end{aligned} \tag{2}$$

In order to solve (2), the *Proximal point* algorithm will be used which has the following iteration:

$$X^{k+1} = \begin{cases} \arg\min & C^T X + \frac{1}{2\gamma_k} \left\|X - X^k\right\|^2 \\ \text{s.t.} & Q_i X \in K_i \quad \forall i = 0\text{:m} \end{cases} \tag{3}$$

where γ_k is an increasing sequence and $K_0 = \{B\}$

2 Proximal point Algorithm

The *Proximal point* algorithm is presented in **Algorithm 1**. As you can see, at each step the algorithm needs to solve sub-problem (3) which can be written equivalently as (4):

$$\begin{aligned} \min_{\overline{X}} \quad & \frac{1}{2} \left\|\overline{X} - (X - \gamma C)\right\|^2 \\ \text{s.t.} \quad & Q_i \overline{X} \in K_i \quad \forall i = 0\text{:m} \end{aligned} \tag{4}$$

In order to solve the equation (4) the *Dykstra* algorithm will be used. But first, the problem needs to be written into its **dual form**.

Algorithm 1: Proximal point Algorithm

Input : maxiter, γ_0, ϵ

$\gamma \leftarrow \gamma_0$

$X \leftarrow zeros(n+1, 0)$

$iter \leftarrow 0$

$crtout \leftarrow 1$

$fval \leftarrow 0$

while $iter \leq maxiter$ and $crtout > \epsilon$ **do**

$$X_{iter+1} = \begin{cases} argmin & C^T X + \frac{1}{2\gamma_k} \|X - X_{iter}\|^2 \\ \text{s.t.} & Q_i X \in K_i \quad \forall i = 0\text{:m} \end{cases}$$

$new\_fval \leftarrow C^T X_{iter+1}$

$crtout \leftarrow \|new\_fval - fval\|$

$fval \leftarrow new\_fval$

$iter \leftarrow iter + 1$

$\gamma \leftarrow 10^{iter}$

end while

Result: The solution of the **SOCP** problem

3 Writing the dual problem

The following notation will be considered:

$$v = X - \gamma C$$

Now, the problem (4) becomes:

$$\begin{aligned} \min_{\overline{X}} \quad & \frac{1}{2} \left\| \overline{X} - v \right\|^2 \\ \text{s.t.} \quad & Q_i \overline{X} \in K_i \quad \forall i = 0\text{:m} \end{aligned} \tag{5}$$

The dual form of (5) is computed as follows:

$$Q_i \overline{X} = X_i \quad \forall i = 0 : m$$

$$\begin{aligned} \max_{y_0, y_1, \ldots, y_m} \quad \min_{\overline{X}, X_0, X_1, \ldots, X_m} \quad & \frac{1}{2} \left\| \overline{X} - v \right\|^2 + \sum_{i=0}^{m} \langle Y_i, Q_i \overline{X} - X_i \rangle \\ \text{s.t.} \quad & X_i \in K_i \quad \forall i = 0 : m \\ = \max_{y_0, y_1, \ldots, y_m} \quad & \sum_{i=0}^{m} - \max_{X_i \in K_i} \langle X_i, Y_i \rangle - \frac{1}{2} \left\| \sum_{i=0}^{m} Q_i^T Y_i \right\|^2 + \left(\sum_{i=0}^{m} Q_i^T Y_i \right)^T v \\ = - \min_{y_0, y_1, \ldots, y_m} \quad & \left(\frac{1}{2} \left\| \sum_{i=0}^{m} Q_i^T Y_i \right\|^2 - \left(\sum_{i=0}^{m} Q_i^T Y_i \right)^T v + \sum_{i=0}^{m} \text{supp}_{K_i}(Y_i) \right) \end{aligned}$$

We will use a *Dykstra* type algorithm to solve the following optimization sub-problem:

$$\min_{y_0, y_1, \ldots, y_n} \left(\frac{1}{2} \left\| \sum_{i=0}^{m} Q_i^T Y_i \right\|^2 - \left(\sum_{i=0}^{m} Q_i^T Y_i \right)^T v + \sum_{i=0}^{m} \operatorname{supp}_{K_i} Y_i \right) \tag{6}$$

4 Dykstra Algorithm

Before presenting the *Dykstra* algorithm (**Algorithm 2**), we will introduce the notation $\Pi_{K_i}(v)$ as being the projection of vector v on the cone K_i. This is computed in the following way:

$$\Pi_{K_i}(v) = \Pi_{K_i}\left(\begin{bmatrix} g \\ r \end{bmatrix}\right) = \begin{cases} v & \|g\| \leq r \\ 0 & \|g\| \leq -r \\ \frac{\|g\|+r}{2} \begin{bmatrix} \frac{g}{\|g\|} \\ 1 \end{bmatrix} & otherwise \end{cases} \tag{7}$$

NB: The *Dykstra* algorithm computes the *crtout* once in a while because it is a computational expensive calculus.

Algorithm 2: Dykstra Algorithm

Input : maxiter, ϵ, v, B, $Q_i \quad \forall i = 0 : m$

$iter \leftarrow 0$
$crtout \leftarrow 1$
$res \leftarrow \sum_{i=0}^{m} \left(Q_i^T Y_i\right) - v$
$L \leftarrow [L_0, L_1, ..., L_m] = \left[\lambda_{max}(Q_0 Q_0^T), \lambda_{max}(Q_1 Q_1^T), ..., \lambda_{max}(Q_m Q_m^T)\right]$
$X \leftarrow 0$
while $iter \leq maxiter$ and $crtout > \epsilon$ **do**
  $i \leftarrow random\_int(0, m)$
  $\delta_i \leftarrow Q_i res$
  $stepgrad \leftarrow Y_i - \frac{1}{L_i}\delta_i$
  $prj \leftarrow 0$
  if i==0 **then**
    $prj \leftarrow \frac{1}{L_i} B$
  else
    $prj \leftarrow \frac{1}{L_i} \Pi_{K_i}(L_i stepgrad)$
  end if
  $new\_Y_i \leftarrow stepgrad - prj$
  $res = res + Q_i^T (new\_Y_i - Y_i)$
  $Y_i \leftarrow new\_Y_i$
  $X \leftarrow -res$
  $iter \leftarrow iter + 1$
  if $iter \quad mod \quad m == 0$ **then**
    $crtout = \|Q_0 X - B\|$
    for $i = 1 : m$ **do**
      $crtout = max(crtout, max(\left\|\overline{Q_i} x + q_i x_0\right\| - (f_i^T x + d_i x_0), 0))$
    end for
  end if
end while

Result: The solution of the optimization sub-problem (6)