ALGORITHMS FOR TWO-STAGE SP: IMPLEMENTATION

Claudia Sagastizábal

BAS Lecture 11, April 14, 2016, IMPA

SVAN 2016
Set YouTube resolution to 480p for best viewing
Decomposition of 2SLP

1st-stage problem

\[ \min c^T x + \phi(x) \]
\[ x \in X \]

2nd-stage subproblems

\[ \phi(x^k) \text{ and a sub-gradient} \]

\[ Q(x^k, \xi^1) \]

\[ Q(x^k, \xi^S) \]
L-shaped method $k$th iteration

The 1st-stage problem has the form

$$
\begin{align*}
\min_{x \in X} & \quad c^\top x + r \\
\text{s.t.} & \quad r \geq O - \text{cut}^i(x) \quad \text{for } i \in J_{\text{Obj}}^{k-1} \\
& \quad 0 \geq F - \text{cut}^{s,i}(x) \quad \text{for } i \in J_{\text{Feas}}^{s,k-1} \text{ and } s = 1, \ldots, S
\end{align*}
$$

We defined

$$
\begin{align*}
O - \text{cut}^i(x) &= \sum_{s=1}^{S} p_s \pi^{s,i} \top (h^s - T^s x) \quad \text{if } x^i \in \text{dom} \phi \\
F - \text{cut}^{s,i}(x) &= \eta^{s,i} \top (h^s - T^s x) \quad \text{if } x^i \notin \text{dom} Q(\cdot, \xi^s), \\
\text{and } J_{\text{Obj}}^k &= \{i < k : x^i \in \text{dom} \phi\} \\
J_{\text{Feas}}^{s,k} &= \{i < k : x^i \notin \text{dom} Q(\cdot, \xi^s) \text{ for } s = 1, \ldots, S\}
\end{align*}
$$
L-shaped method \( k \)th iteration

\[
\min_{x \in X} c^T x + r \\
r \geq 0 - \text{cut}^i(x) \\
0 \geq F - \text{cut}^{s,i}(x)
\]

and a sub-
\[
\phi(x^k)
\]

gradient

\[
\begin{align*}
\mathbf{Q}(x^k, \xi^1) \\
\text{or} \\
U(x^k, \xi^s) \\
\max_{\pi} \pi^T (h^s - T^s x^k) : W^T \pi \leq q^s \\
\text{or} \\
\max_{\eta} \eta^T (h^s - T^s x^k) : W^T \eta \leq 0, ||\eta||_* \leq 1
\end{align*}
\]
L-shaped method \( k \)th iteration

1st-stage problem

\[
\min_{x \in X} c^\top x + r \\
\quad r \geq O - \text{cut}^i(x) \\
\quad 0 \geq F - \text{cut}^{s,i}(x)
\]

there is also a multi-cut variant

\[\phi(x^k)\] gradient

\[Q(x^k, \xi^1)\]

\[\text{or}\]

\[U(x^k, \xi^s)\]

\[Q(x^k, \xi^s)\]

\[\max \pi^\top (h_s - T^s x^k) : W^\top \pi \leq q^s\]

or

\[\max \eta^\top (h_s - T^s x^k) : W^\top \eta \leq 0, \|\eta\|_* \leq 1\]
Application: Cash-Matching Problem

Over the next $t = 1, \ldots, T$ years a company plans to make payments $d_t$, that will be financed by buying $i = 1, \ldots, n$ bonds with known return $r_{it}$ and cost $c_i$. For a capital $K$ and each $i = 1, \ldots, n$, we need to determine $x_i$, the number of bonds of type $i$ to buy today so that at the end of the horizon we maximize our money in a manner that in no period we are in red.
Mathematical Formulation

Introducing cumulative gains and losses

\[ \text{$IN$} \quad \text{Return of bond } i \text{ until time } t \quad a_{it} = \sum_{\tau=1}^{t} r_{i\tau} - c_i \]

\[ \text{$OUT$} \quad \text{Payments until time } t \quad h_t = \sum_{\tau=1}^{t} d_{\tau} - K \]
Mathematical Formulation

Introducing cumulative gains and losses

$\text{IN}$ Return of bond $i$ until time $t$ \hspace{1cm} $a_{it} = \sum_{\tau=1}^{t} r_{i\tau} - c_i$

$\text{OUT}$ Payments until time $t$ \hspace{1cm} $h_t = \sum_{\tau=1}^{t} d_{\tau} - K$

the problem can be written as follows:

$$\begin{aligned}
\max_{\chi} & \quad \sum_{i=1}^{n} a_{iT} \chi_i \\
\text{s.t.} & \quad \sum_{i=1}^{n} a_{it} \chi_i \geq h_t \text{ for } t = 1, \ldots, T
\end{aligned}$$
Mathematical Formulation

Introducing cumulative gains and losses

$\text{IN}$ Return of bond $i$ until time $t$

$a_{it} = \sum_{\tau=1}^{t} r_{i\tau} - c_i$

$\text{OUT}$ Payments until time $t$

$h_t(\omega) = \sum_{\tau=1}^{t} d_\tau - K$

the problem can be written as follows:

\[
\begin{align*}
\max_{\chi} & \quad \sum_{i=1}^{n} a_{iT} x_i \\
\text{s.t.} & \quad \sum_{i=1}^{n} a_{it} x_i \geq h_t \quad \text{for } t = 1, \ldots, T
\end{align*}
\]
Mathematical Formulation

\[
\begin{align*}
\max_{\chi} \quad & \sum_{i=1}^{n} a_{i\tau} \chi_i \\
\text{s.t.} \quad & \sum_{i=1}^{n} a_{it} \chi_i \geq h_t(\omega) \quad \text{for } t = 1, \ldots, T
\end{align*}
\]

**Difficulty:** exact payments are not known in advance, \(d_\tau\) and \(h_t\) are random

**What about a stochastic programming approach?**
Mathematical Formulation

\[
\begin{align*}
\max_{\chi} & \quad \sum_{i=1}^{n} a_{iT} x_{i} \\
\text{s.t.} & \quad \sum_{i=1}^{n} a_{it} x_{i} \geq h_{t}(\omega) \quad \text{for } t = 1, \ldots, T
\end{align*}
\]

**Difficulty:** exact payments are not known in advance, \( d_{\tau} \) and \( h_{t} \) are random

What about a stochastic programming approach?

Let’s formulate a 2-stage model with recourse
Decision stages $\neq$ time steps

For each $i = 1, \ldots, n$, we need to determine $x_i$, the number of bonds of type $i$ to buy today so that at the end of the horizon we maximize our money in a manner that in no period we are in red.
Decision stages ≠ time steps

For each \( i = 1, \ldots, n \), we need to determine \( x_i \), the number of bonds of type \( i \) to buy today so that at the end of the horizon we maximize our money in a manner that in no period we are in red.

This is a formulation without recourse
Decision stages \(\neq\) time steps

For each \(i = 1, \ldots, n\), we need to determine \(x_i\), the number of bonds of type \(i\) to buy today so that at the end of the horizon we maximize our money in a manner that in no period we are in red.

This is a formulation without recourse

Instead, we can buy \(x_i\) today \((t = 1)\) and allow for wait-and-see adjustments \(y_i(\omega)\) in the future \((t = 2)\)
Decision stages ≠ time steps

For each $i = 1, \ldots, n$, we need to determine $x_i$, the number of bonds of type $i$ to buy today so that at the end of the horizon we maximize our money in a manner that in no period we are in red. This is a formulation without recourse.

Instead, we can buy $x_i$ today ($t = 1$) and allow for wait-and-see adjustments $y_i(\omega)$ in the future ($t = 2$). This is a formulation with recourse.
Model with recourse

If there is recourse of buying $y(\omega)$ at $t = 2$:

$$GAIN_{t = 1} = (r_{i1} - c_i)x_i$$

$$GAIN_{t \geq 2} = (r_{i1} - c_i)x_i + (\sum_{\tau=2}^{t} r_{i\tau} - c_i)(x_i + y_i(\omega))$$

$$OUT_t(\omega) = \sum_{\tau=1}^{t} d_{\tau}(\omega) - K$$
Model with recourse

If there is recourse of buying $y(\omega)$ at $t = 2$:

$GAIN \ t = 1 \quad (r_{i1} - c_i) x_i$

$GAIN \ t \geq 2 \quad (r_{i1} - c_i) x_i + (\sum_{\tau=2}^{t} r_{i\tau} - c_i)(x_i + y_i(\omega))$

$\quad = a_{it} x_i + (a_{it} - r_{i1}) y_i(\omega)$

$OUT \quad h_t(\omega) = \sum_{\tau=1}^{t} d_{\tau}(\omega) - K$
Cash-matching problem: recourse model

\[
\begin{align*}
\max & \quad \sum_{i=1}^{n} a_{iT} x_i + \mathbb{E} \left[ \sum_{i=1}^{n} (a_{iT} - r_{i1}) y_i(\omega) \right] \\
\text{s.t.} & \quad 0 \leq x \leq K \text{ and } 0 \leq y(\omega) \leq K \text{ a.e. } \omega \\
& \quad \sum_{i=1}^{n} a_{i1} x_i \geq h_1 \\
& \quad \sum_{i=1}^{n} a_{it} x_i + \sum_{i=1}^{n} (a_{it} - r_{i1}) y_i(\omega) \geq h_t(\omega) \text{ a.e. } \omega \\
& \quad \text{for } t = 1, \ldots, T
\end{align*}
\]
Preparing the implementation: data generation

function [data]=genCMData(Nscen, Resample)
data.Nscen=Nscen; data.resample=Resample;
[data]=CashMatchingData;
[data.scen, data.lb_scen, data.resample]=CMGenScen(data);
Preparing the implementation: data generation

genCMData.m

function [data]=genCMData(Nscen,Resample)
data.Nscen=Nscen;data.resample=Resample;
[data]=CashMatchingData;
[data.scen,data.lb_scen,data.resample]=CMGenScen(data);

CashMatchingData.m

n=3;T=15;K = 250000;
c = [980;970;1050];
d = 1000*[11 12 14 15 16 18 20 21 22 24 25 30 31 31 31]';
r= [ 0 0 0
    60 65 75
    60 65 75
    60 65 75
    60 65 75]
\[ \begin{bmatrix}
60 & 65 & 75 \\
0 & 65 & 75 \\
0 & 65 & 75 \\
0 & 65 & 75 \\
0 & 65 & 75 \\
0 & 65 & 75 \\
0 & 65 & 75 \\
0 & 0 & 75 \\
0 & 0 & 75 \\
0 & 0 & 1075 \\
\end{bmatrix}; \\
\]

\[
\sigma = 500 \ast [1:T]';
\]
A_scen = zeros(n,T);
    for i=1:n
        for j=1:T
            s=sum(r(1:j,i));
            A_scen(i,j) = s - c(i);
        end
    end
    c_scen = A_scen(:,T);c_scen = -c_scen; %max problem
    A_scen = A_scen';

data.n=n;data.T=T;data.K=K;data.c=c;data.d=d;data.r=r;
data.sigma=sigma;data.A_scen=A_scen;data.c_scen=c_scen;
end
Oracle: your turn!

function [intercept, slope, data] = CashMatchingOracle(xk, data)

% Output: intercept and slope of cut at xk
% its type: Optimality cut or Feasibility cut
% setting data.Infeas=0 if optimality cut
% or data.Infeas=[s_1; s_2; ...],
% the involved scenarios

intercept=[]; slope=[]; data.Infeas=[];
intercept = ...;
slope = ...;
data.Infeas = ...;
return