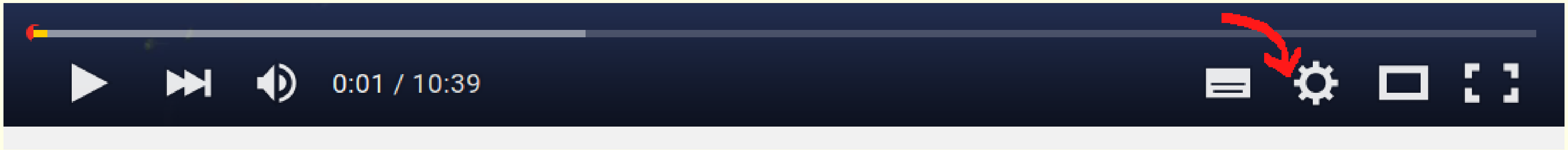
ALGORITHMS FOR TWO-STAGE SP: IMPLEMENTATION

Claudia Sagastizábal

BAS Lecture 11, April 14, 2016, IMPA

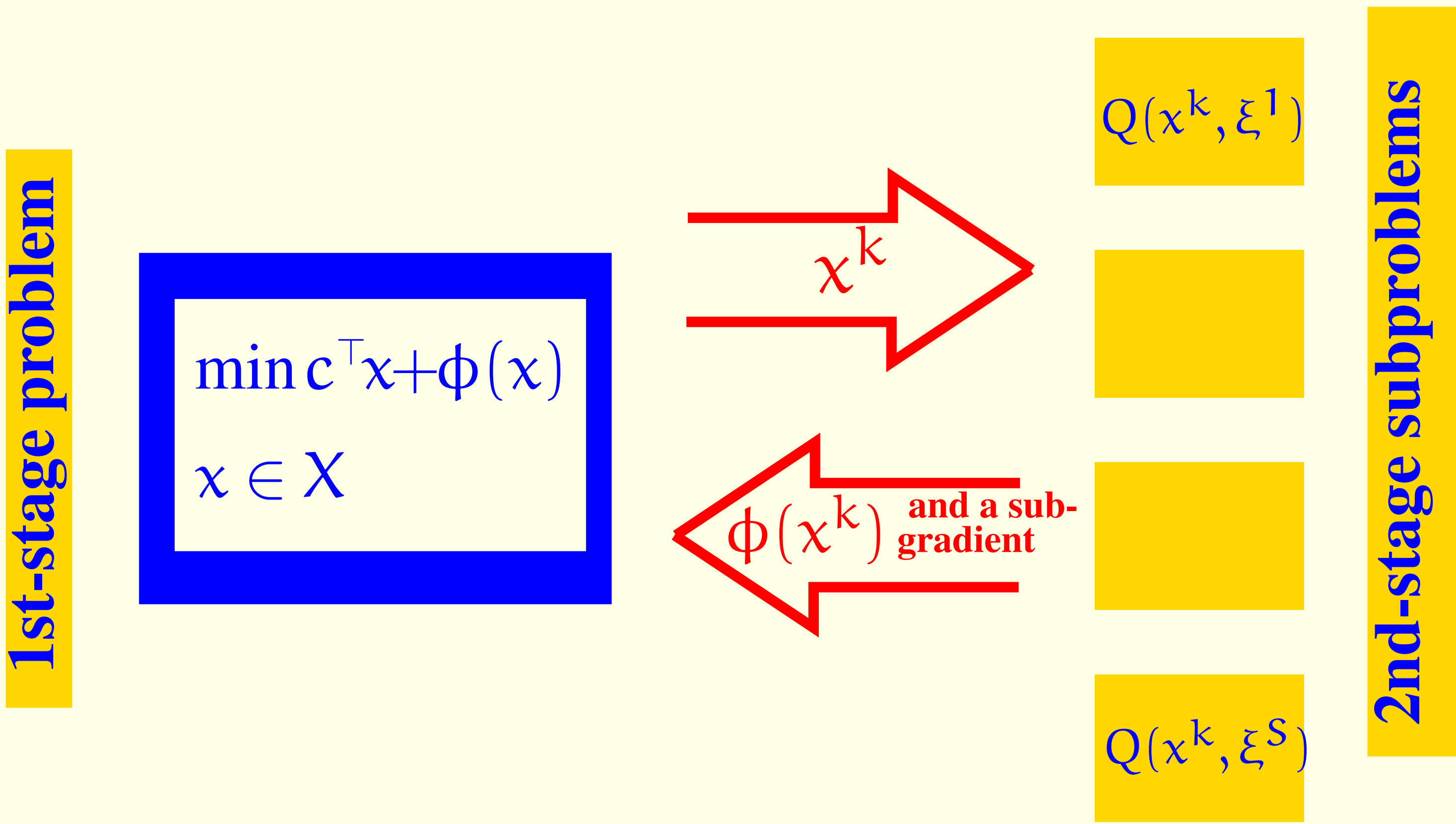
SVAN 2016

Set YouTube resolution to 480p



for best viewing

Decomposition of 2SLP



L-shaped method kth iteration

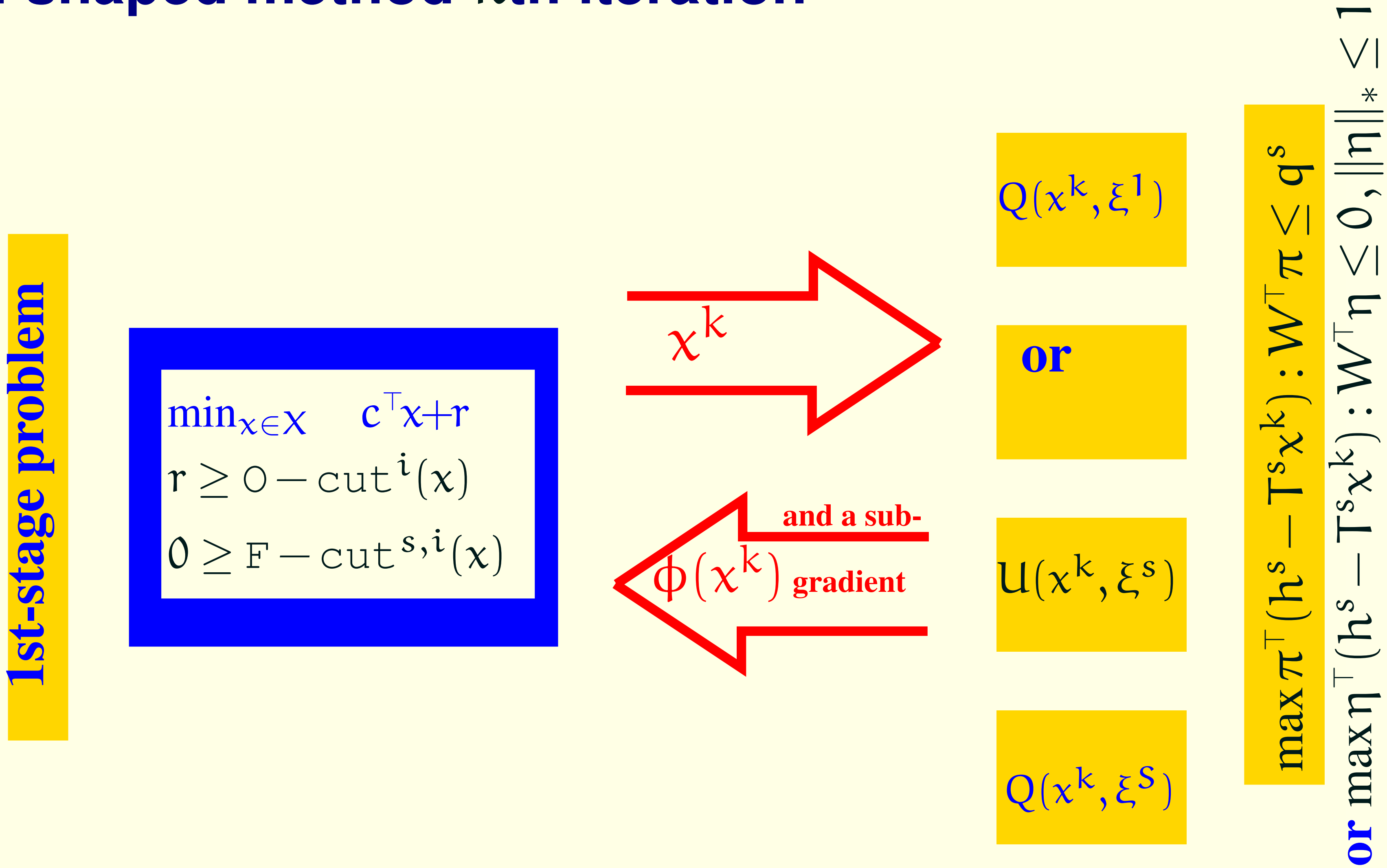
The 1st-stage problem has the form

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

We defined

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

L-shaped method kth iteration



L-shaped method kth iteration

1st-stage problem

$$\min_{x \in X} \quad c^\top x + r$$

$$r \geq 0 - \mathrm{cut}^i(x)$$

$$0 \geq F - \mathrm{cut}^{s,i}(x)$$

x^k

$\phi(x^k)$ **and a sub-gradient**

$Q(x^k, \xi^1)$

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 $\max \eta^\top (h^s - T^s x^k) : W^\top \eta \leq 0, \|\eta\|_* \leq 1$

there is also a multi-cut variant

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

\$ IN	Return of bond i until time t	$a_{it} = \sum_{\tau=1}^{t} r_{i\tau} - c_i$
\$ OUT	Payments until time t	$h_t = \sum_{\tau=1}^{t} d_\tau - K$

Mathematical Formulation

Introducing cumulative gains and losses

\$ IN	Return of bond i until time t	$a_{it} = \sum_{\tau=1}^{t} r_{i\tau} - c_i$
\$ OUT	Payments until time t	$h_t = \sum_{\tau=1}^{t} d_\tau - K$

the problem can be written as follows:

$$\begin{cases} \max_x & \displaystyle\sum_{i=1}^{n} a_{iT} x_i \\ \text{s.t.} & \displaystyle\sum_{i=1}^{n} a_{it} x_i \geq h_t \text{ for } t = 1, \ldots, T \end{cases}$$

Mathematical Formulation

Introducing cumulative gains and losses

\$ IN	Return of bond i until time t	$a_{it} = \sum_{\tau=1}^{t} r_{i\tau} - c_i$
\$ OUT	Payments until time t	$h_t(\omega) = \sum_{\tau=1}^{t} \boxed{d_\tau} - K$

the problem can be written as follows:

$$\begin{cases} \max_x \ \displaystyle\sum_{i=1}^{n} a_{iT} x_i \\ \text{s.t.} \ \displaystyle\sum_{i=1}^{n} a_{it} x_i \geq \boxed{h_t} \text{ for } t = 1, \ldots, T \end{cases}$$

Mathematical Formulation

$$\begin{cases} \max_x & \sum_{i=1}^{n} a_{iT} x_i \\ \text{s.t.} & \sum_{i=1}^{n} a_{it} x_i \geq \boxed{h_t(\omega)} \text{ for } t = 1, \ldots, T \end{cases}$$

Difficulty: exact payments are not known in advance, d_τ and h_t are random

What about a stochastic programming approach?

Mathematical Formulation

$$\begin{cases} \max_x \ \sum_{i=1}^{n} a_{iT} x_i \\ \text{s.t.} \quad \sum_{i=1}^{n} a_{it} x_i \geq \boxed{h_t(\omega)} \text{ for } t = 1, \ldots, T \end{cases}$$

Difficulty: exact payments are not known in advance, d_τ 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 $\neq$ 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

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 adjustements $y_i(\omega)$ in the future $(t = 2)$

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 adjustements $y_i(\omega)$ 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 $t = 1 \quad (r_{i1} - c_i)x_i$

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

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

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

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

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

Cash-matching problem: recourse model

$$
\begin{cases}
\max & \displaystyle\sum_{i=1}^{n} a_{iT} x_i + \mathbb{E}\Big[\sum_{i=1}^{n} (a_{iT} - r_{i1}) y_i(\omega)\Big] \\
\text{s.t.} & 0 \le x \le K \text{ and } 0 \le y(\omega) \le K \text{ a.e.} \omega \\
& \displaystyle\sum_{i=1}^{n} a_{i1} x_i \ge h_1 \\
& \displaystyle\sum_{i=1}^{n} a_{it} x_i + \sum_{i=1}^{n} (a_{it} - r_{i1}) y_i(\omega) \ge h_t(\omega) \text{ a.e.} \omega \\
& \hfill \text{for } t = 1, \ldots, T
\end{cases}
$$

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
 1060 65 75
    0 65 75
    0 65 75
    0 65 75
    0 65 75
    0 65 75
    0 1060 75
    0  0 75
    0  0 75
    0  0 1075];


sigma = 500*[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
```