Optimal scheduling for open pit mine extraction

Michel De Lara

October 10, 2017

Contents

1 A two-dimensional open pit mine optimal extraction model

Open pit mines are generally designed with the help of so-called *block models*, or *resource models*. These latter represent the material inside the pit using millions of blocks with given shape. Open pit mines extraction is submitted to physical restrictions: only blocks at the surface may be extracted; a block cannot be extracted if the slope made with one of its neighbors is too high, due to geotechnical constraints on mine wall slopes.

In essence, exploiting an open pit mine can be seen as selecting an *admissible extraction sequence* of blocks where, by admissible, we mean that the above physical restrictions are satisfied. Among all admissible extraction sequences, of particular economic interest are those which are the more profitable.

Block model

For simplicity, we shall consider a two-dimensional mine [1] as in Fig. 1. Each block is a two-dimensional rectangle identified by its vertical position $d \in \{1, \ldots, D\}$ (d for depth) and by its horizontal position $c \in \{1, \ldots, C\}$ (c for column).

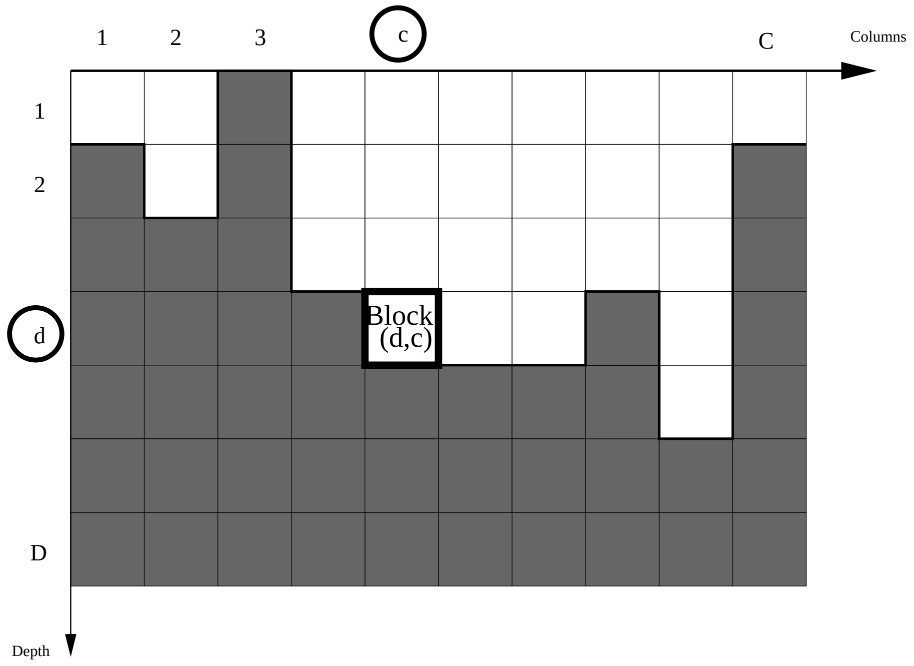

Figure 1: An extraction profile in an open pit mine

Dynamics

We assume that blocks are extracted sequentially under the following hypothesis [2, 1]:

- it takes one time unit to extract one block;
- only blocks at the surface may be extracted;
- a block cannot be extracted if the slope made with one of its two neighbors is too high, due to geotechnical constraints on mine wall slopes (the rule is mathematically decribed in the sequel);
- a retirement option is available where no block is extracted.

States and admissible states

Denote discrete time by $t = 0, 1, \ldots, T$, where an upper bound for the number of extraction steps is the number $C \times D$ of blocks (it is in fact strictly lower due to slope constraints). At time t, the *state* of the mine is a *profile*

$$x(t) = \big(x_1(t), \ldots, x_C(t)\big) \in \mathbb{X} = \{1, \ldots, D+1\}^C$$

where $x_c(t) \in \{1, \ldots, D+1\}$ is the vertical position of the top block with horizontal position $c \in \{1, \ldots, C\}$ (see Fig. 1). The initial profile is $x(0) = (1, 1, \ldots, 1)$ while the mine is totally exhausted in state $x = (D+1, D+1, \ldots, D+1)$.

An admissible profile is one that respects local angular constraints at each point, due to physical requirements. A state $x = (x_1, \dots, x_C)$ is said to be *admissible* if the geotechnical slope constraints are respected in the sense that

$$\begin{cases} x_1 = 1 \quad \text{or} \quad 2 \quad \text{(border slope)} \\ |x_{c+1} - x_c| \leq 1 \,, \quad \text{for} \quad c = 1, \dots, C-1 \\ x_C = 1 \quad \text{or} \quad 2 \quad \text{(border slope).} \end{cases} \tag{1}$$

Denote by $\mathbb{A} \subset \{1, \dots, D+1\}^C$ the set of admissible states satisfying the above slope constraints (1).

Controlled dynamics

A decision is the selection of a column in $\mathbb{C} = \{1, \dots, C\}$, the top block of which will be extracted. A decision may also be the retirement option, that we shall identify with an additional fictituous column denoted ∞. Thus, a decision u is an element of the set $\overline{\mathbb{C}} = \mathbb{C} \cup \{\infty\} = \{1, \dots, C, \infty\}$.

At time t, if a column $u(t) \in \{1, \dots, C\}$ is chosen at the surface of the open pit mine, the corresponding block is extracted and the profile $x(t) = \big(x_1(t), \dots, x_C(t)\big)$ becomes

$$x_c(t+1) = \begin{cases} x_c(t) + 1 & \text{if} \quad c = u(t) \\ x_c(t) & \text{else.} \end{cases}$$

In case of retirement option $u(t) = \infty$, then $x(t+1) = x(t)$ and the profile does not change. In other words, the dynamics is given by $x(t+1) = \mathsf{DYN}(x, u)$ where

$$\mathsf{DYN}_c(x, u) = \begin{cases} x_c + 1 & \text{if} \quad c = u \in \{1, \dots, C\} \\ x_c & \text{if} \quad c \neq u \quad \text{or} \quad c = \infty \,. \end{cases} \tag{2}$$

Indeed, the top block of column c is no longer at depth $x_c(t)$ but at $x_c(t) + 1$, while all other top blocks remain.

Starting from state $x = (2, 3, 1, 4, 3)$ in Figure 2 and applying control $u = 3$, one obtains the following state $(2, 3, 2, 4, 3)$ as in Figure 3.

Decision constraints

Not all decisions $u(t) = c$ are possible either because there are no blocks left in column c ($x_c = D + 1$) or because of slope constraints.

When in state $x \in \mathbb{A}$, the decision $u \in \overline{\mathbb{C}}$ is *admissible* if the future profile $\mathsf{DYN}(x, u) \in \mathbb{A}$, namely if it satisfies the geotechnical slope constraints. This may easily be transformed into a condition $u \in \mathbb{B}(x)$, where

$$\mathbb{B}(x) := \{u \in \overline{\mathbb{C}} \mid \mathsf{DYN}(x, u) \in \mathbb{A}\} \,. \tag{3}$$

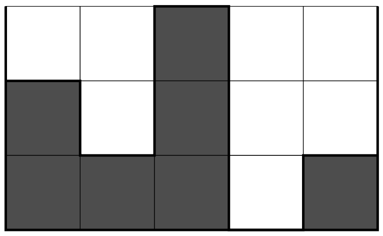

Figure 2: State (2,3,1,4,3)

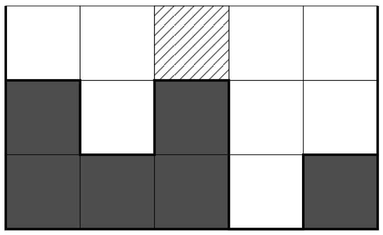

Figure 3: State (2,3,2,4,3)

Intertemporal profit maximization

The open pit mine optimal scheduling problem consists in finding a path of admissible blocks which maximizes an intertemporal discounted extraction profit. It is assumed that the value of blocks differs in depth and column because richness of the mine is not uniform among the zones as well as costs of extraction. The profit model states that each block has an economic value $\mathsf{Rent}(d,c) \in \mathbb{R}$, supposed to be known.[1] By convention $\mathsf{Rent}(d,\infty) = 0$ when the retirement option is selected. Selecting a column $u(t) \in \mathbb{C}$ at the surface of the open pit mine, and extracting the corresponding block[2] at depth $x_{u(t)}(t)$ yields the value $\mathsf{Rent}\big(x_{u(t)}(t), u(t)\big)$. With discount rate $r_f \geq 0$ and discount factor $0 \leq \frac{1}{1+r_f} \leq 1$, the optimization problem is $\sup_{u(\cdot)} \sum_{t=0}^{+\infty} (\frac{1}{1+r_f})^t \mathsf{Rent}\big(x_{u(t)}(t), u(t)\big)$. Notice that the sum is in fact finite, bounded above by the number $T-1$ of blocks. Thus, we shall rather consider[3]

$$\sup_{u(0),\ldots,u(T-1)} \sum_{t=0}^{T-1} (\frac{1}{1+r_f})^t \mathsf{Rent}\big(x_{u(t)}(t), u(t)\big) . \tag{4}$$

[1]Partly determined by "carrots" extraction, this economic value may more generally be assumed either deterministic or random.

[2]When $u(t) = \infty$, there is no corresponding block and the following notation $x_{u(t)}(t) = x_\infty(t)$ is meaningless, but this is without incidence since the value $\mathsf{Rent}\big(x_\infty(t), \infty\big) = 0$.

[3]If we account for transportation costs, we may subtract to Rent a term proportional to $\delta(u(t), u(t-1))$, measuring the distance between two subsequent extraction columns.

Dynamic programming equation

The maximization problem (4) may theoretically be solved by dynamic programming. The value function $\mathcal{V}(t,x)$ solves $\mathcal{V}(T,x)=0$ and

$$\mathcal{V}(t,x)=\max_{u\in\mathbb{B}(x)}\Big((\frac{1}{1+r_f})^t\mathsf{Rent}(x_u,u)+\mathcal{V}\big(t+1,\mathsf{DYN}(x,u)\big)\Big)\,. \tag{5}$$

However, due to numerical considerations (curse of dimensionality), we shall have to introduce a new, and more parcimonious, state before applying the dynamic programming algorithm.

2 Dynamic programming algorithm

Numerical considerations

To give a flavor of the complexity of the problem, notice that 4 columns ($C=4$) each consisting of nine blocks ($D=9$) give 10 000 states ($\mathbb{X}^\sharp=10^4$), while this raises to $10^{20\ 000}$ if we assume that the surface consists of 100×100 columns ($C=10\ 000$) with ninety-nine blocks ($D=99$) .

However, the set $\mathbb{A}$ of acceptable states has a cardinal $\mathbb{A}^\sharp$ which is generally much smaller than $\mathbb{X}^\sharp$. To see this, let us introduce the mapping $x=(x_1,\ldots,x_C)\mapsto\varphi(x)=(x_1,x_2-x_1,\ldots,x_C-x_{C-1})$. Let $x\in\mathbb{A}$ and $y=\varphi(x)$. Since x satisfies the admissibility condition (1), y necessarily satisfies $y_1\in\{1,2\}$ and $\sup_{c=2,\ldots,C}|y_c|\leq1$. Hence, $\text{card}(\mathbb{A})\leq2\times3^{C-1}$ and the dynamic programming algorithm will be released with the new state $y=(y_1,\ldots,y_C)\in\{1,2\}\times\{-1,0,1\}^{C-1}$ corresponding to the increments of the state x. Table 2 provides some figures.

C	D	$\mathbb{X}^\sharp$	$\mathbb{A}^\sharp$	$\mathbb{A}^\sharp/\mathbb{X}^\sharp$
4	9	10^4	≤ 270	≤ 3 %
10 000	99	$10^{20\ 000}$	$\leq 2\times10^{10\ 000}$	$2\times\leq10^{-10\ 000}$

Table 1: Cardinals of states and acceptable states sets

A new incremental state

The dynamic programming algorithm will be released with the new state

$$y=(y_1,\ldots,y_C)\in\mathbb{Y}:=\{1,2\}\times\{-1,0,1\}^{C-1} \tag{6}$$

corresponding to the increments of the state x given by the inverse mapping

$$y=(y_1,\ldots,y_C)\in\mathbb{Y}\mapsto\varphi^{-1}(y)=(y_1,y_1+y_2,\ldots,y_1+y_2+\cdots+y_C)\in\mathbb{N}^C\,. \tag{7}$$

By construction, $x = \varphi^{-1}(y)$ always satisfies the slope constraints (1). However, it does not always correspond to a mine profile state. Indeed, some $y \in \mathbb{Y}$ give $x = \varphi^{-1}(y) \notin \mathbb{X} = \{1, \dots, D+1\}^C$: for instance $y = (1, -1, -1, \dots, -1)$ yields $x = (1, 0, -1, -2, \dots, C-2)$.

The dynamics $(x, u) \mapsto \mathsf{DYN}(x, u)$ now becomes $(y, u) \mapsto G(y, u) = \varphi\big(\mathsf{DYN}(\varphi^{-1}(y), u)\big)$:

$$G(y, u)_c = \begin{cases} y_c + 1 & \text{if} \quad c = u \\ y_c - 1 & \text{if} \quad c = u + 1 \\ y_c & \text{else.} \end{cases} \tag{8}$$

Labelling the states

The new state set $\mathbb{Y}$ is a product space: we shall label its elements with integers. Let $Q \in \{-1, 0, 1, 2, \dots\}$ be the unique integer such that

$$3^{Q+1} < D + 1 \leq 3^{Q+2} \,. \tag{9}$$

Consider the following *labelling* mapping[4]

$$y = (y_1, \dots, y_C) \in \mathbb{Y} \mapsto \lambda(y) := (y_1 - 1) + \sum_{i=2}^{C} (y_i + 1) 3^{Q+i} \in \mathbb{N} \,. \tag{10}$$

The inverse mapping of λ is given as follows. Any integer $z \in \{0, 1, \dots, 3^{Q+C+1}\}$ may be decomposed in the basis $\{1, 3^{Q+2}, \dots, 3^{Q+C}\}$ as

$$z = z_1 + \sum_{i=2}^{C} z_i 3^{Q+i} \qquad \text{with} \qquad z_1 \in \{0, 1, \dots, 3^{Q+2} - 1\} \qquad \text{and} \qquad z_i \in \{0, 1, 2\} \tag{11}$$

for $i = 2, \dots, C$. Thus $(y_1, \dots, y_C) = \lambda^{-1}(z)$ is given by $y_1 = z_1 + 1 \in \{1, \dots, 3^{Q+2}\}$ and $y_i = z_i - 1 \in \{-1, 0, 1\}$ for $i = 2, \dots, C$. Denote by $\mathbb{Z}_a \subset \{0, 1, \dots, 3^{Q+C+1}\}$ those z to which corresponds an admissible state $x = \varphi^{-1}\big(\lambda^{-1}(z)\big) \in \mathbb{A}$.

The dynamics (8) is translated in

$$(z, u) \in \mathbb{N} \times \{1, \dots, C\} \mapsto H(z, u) := \begin{cases} z + 1 - 3^{Q+2} & \text{if} \quad u = 1 \\ z + 3^{Q+u} - 3^{Q+u+1} & \text{if} \quad u \in \{2, \dots, C-1\} \\ z + 3^{Q+C} & \text{if} \quad u = C \end{cases} \tag{12}$$

because of the expression (11) for z.

Let $z \in \mathbb{Z}_a$. Of course, $z' = H(z, u)$ corresponds to an admissible state ($z' \in \mathbb{Z}_a$) if and only if u is admissible for the state $x = \varphi^{-1}\big(\lambda^{-1}(z)\big) \in \mathbb{A}$, property that we shall denote by $u \in \mathbb{U}_a(z) \subset \{1, \dots, C\}$.

[4]The justification of the terms $(y_1 - 1)$ and $(y_i + 1)$ is as follows. Recall that $(y_1, \dots, y_C) \in \mathbb{Y} = \{1, 2\} \times \{-1, 0, 1\}^{C-1}$. Thus, $y_1 \in \{1, \dots, 3^{Q+2}\}$ implies that $(y_1 - 1) \in \{0, 1, \dots, 3^{Q+2} - 1\}$, while $y_i \in \{-1, 0, 1\}$ implies that $(y_i + 1) \in \{0, 1, 2\}$.

3 Simulating extraction strategies

3.1 Data

Copy the file `"marvin_7440_valores.txt"` in your local directory. Then, create a file `"data.sce"` in your local directory containing the following code.

```
// ------------------------------------------

// 2D agregated Marvin mine
RICHNESS=zeros(5,11);
richness=fscanfMat("marvin_7440_valores.txt");
for i=1:5 do
  for j=1:11 do
    RICHNESS(i,j)=sum(richness([((i-1)*4+1):mini((i*4),17)],[((j-1)*6+1):mini((j*6),61)]
  end
end
richness=RICHNESS;

// ------------------------------------------
// RICHNESS MATRIX SIZE
// ------------------------------------------

[HH,NN]=size(richness);
TT=HH*(HH+2)+1;// horizon
TT=HH*NN;// horizon


// ------------------------------------------
// DISCOUNT FACTOR
// ------------------------------------------

interest=10;
annual_discount=1/(1+(interest/100));

discount=(annual_discount)^(1/1500);
// 1500 extractions a year

discount=(annual_discount)^(1/(6*4));
// 6*4 extractions a year
```

3.2 Upper and lower bounds for the value of the mine

The so called *value of the mine* is $\mathcal{V}(0, x(0))$ given by (4).

Question 1 *Rearrange the elements of the matrix* Rent(d, c) *and deduce an upper bound for the value of the mine.*

```
//------------------------------------------
// UPPER BOUND VALUE OF THE MINE
//------------------------------------------

blocks=prod(size(richness));

rich=matrix(richness,1,blocks);
sorted_richness=sort(rich);// richness sorted by decreasing order
sorted_richness=maxi(0,sorted_richness);

UB=sum((discount .^(0:(blocks-1))) .*sorted_richness);
```

3.3 Optimal trajectories simulation

State labelling

Copy the following code in a file "`mine_DP_label.sce`"

```
// ------------------------------------------
// LABELLING STATES AND DYNAMICS
// ------------------------------------------

MM=3^{NN};// number of states

Integers=(1:MM)';// labelled states

State=zeros(MM,NN);
// Will contain, for each integer z in Integers,
// a sequence s=(s_1,...,s_NN)  with
// s_1 \in \{1,2\}
// and s_k \in \{0,1,2\} for k \geq 2, such that
// 1) z = s_1 + s_2*3^{1} + ... + s_NN*3^{NN-1}
// 2) a mine profile p_1,...,p_NN is given by
// p_k = y_1 + ... + y_k where y_1= s_1-1 \in \{0,1\}
// and y_j = s_j - 1 \in \{-1,0,1\} for j>1.

Increments=zeros(MM,NN);
// Will contain, for each integer z in Integers,
// the sequence y=(y_1,...,y_NN).

// The initial profile is supposed to be p(0)=(0,0,...,0)
// to which corresponds y(0)=(0,0,...,0) and
// s(0)=(1,1,...,1) and z(0)=1+ 3^{1} + ... + 3^{NN-1}.

Partial_Integer=zeros(MM,NN);
// Will contain, for each integer z in Integers,
// a lower aproximation of z in the basis
// 1, 3^{1},..., 3^{NN-1}
// Partial_Integer(z,k)=s_1 + s_2*3^{1} +...+ s_k*3^{k-1}.

Future_Integer=zeros(MM,NN);
// Will contain, for each integer z in Integers,
// the image by the dynamics under the control consisting in
// extracting block in the corresponding column.

State(:,1)=pmodulo(Integers,3^{1});
// State(z,1)=s_1
Partial_Integer(:,1)=State(:,1);
// Partial_Integer(z,1)=s_1
Future_Integer(:,1)=maxi(1,Integers+1-3^{1});
// Dynamics (with a "maxi" because some integers in
// Integers+1-3^{1} do not correspond to "mine profiles").
Increments(:,1)=State(:,1)-1;

for k=2:NN do
  remainder=(Integers-Partial_Integer(:,k-1))/3^{k-1};
  // s_{k} + s_{k+1}*3 + ...
  State(:,k)=pmodulo(remainder,3);
  // State(:,k)=s_{k}
  Increments(:,k)=State(:,k)-1;
  Partial_Integer(:,k)=Partial_Integer(:,k-1)+3^{k-1}*State(:,k);
  // Partial_Integer(z,k)=s_1+s_2*3^{1}+...+s_k*3^{k-1}
  Future_Integer(:,k)=maxi(1,Integers+3^{k-1}-3^{k});
  // Dynamics (with a "maxi" because some integers
  // in Integers+3^{k-1}-3^{k}
  // do not correspond to "mine profiles")
end

Future_Integer(:,NN)=mini(MM,Integers+3^{NN-1});
// Correction for the dynamics
// when the last column NN is selected.
// Dynamics (with a "mini" because some integers in
// Integers+3^{NN-1} do not correspond to "mine profiles").
// ---------------------------------------------
// FROM PROFILES TO INTEGERS
// ---------------------------------------------

function z=profile2integer(p)
  // p : profile as a row vector
  yy=p-[0,p(1:($-1))];
  ss=yy+1;
  z=sum(ss .*[1,3 .^{[1:(NN-1)]}]);
endfunction


// ---------------------------------------------
// ADMISSIBLE INTEGERS
// ---------------------------------------------

// Mine profiles are those for which
// HH \geq p_1 \geq 0,..., HH \geq p_NN \geq 0
// that is, HH \geq y_1 + ... + y_k \geq 0 for all k
// Since, starting from the profile p(0)=(0,0,...,0)),  the
// following extraction rule will always give "mine profiles",
// we shall not exclude other unrealistic profiles.

Admissible=zeros(MM,NN);

Profiles=cumsum(Increments,"c");
// Each line contains a profile, realistic or not.

adm_bottom=bool2s(Profiles < HH);
// A block at the bottom cannot be extracted:
// an element in adm_bottom is one if and only if
// the top block of the column is not at the bottom.

// Given a mine profile, extracting one block at the surface
// is admissible if the slope is not too high.
//
// Extracting block in column 1 is admissible
// if and only if p_1=0.
//
```

```
// Extracting block in column j 1<j<NN) is not admissible
// if and only if y_j=1 or y_{j+1}=-1 that is,
// (s_j-1)=1 or (s_{j+1}-1)=-1.
// Extracting block in column j is admissible
// if and only if s_j < 2 and s_{j+1} > 0.
//
// Extracting block in column NN is admissible
// if and only if p_{NN}=0.

Admissible(:,1)=bool2s(Profiles(:,1)==0);
Admissible(:,NN)=bool2s(Profiles(:,NN)==0);
// Corresponds to side columns 1 and NN, for which only the
// original top block may be extracted:
// an element in columns 1 and NN of Admissible is one
// if and only if the pair (state,control) is admissible.

Admissible(:,2:($-1))=bool2s(State(:,2:($-1)) < 2 & State(:,3:$) > 0);
// An element in column 1<j<NN of AA is one if and only
// s_j < 2 and s_{j+1} > 0.

Admissible=Admissible .*adm_bottom;
// An element in column j of admissible is zero
// if and only if
// extracting block in column j is not admissible,
// else it is one.


Stop_Integers=Integers(prod(1-Admissible,"c")==1);
// Labels of states for which no decision is admissible,
// hence the decision is the retirement option


// -----------------------------------------
// INSTANTANEOUS GAIN
// -----------------------------------------

Forced_Profiles=mini(HH,maxi(1,Profiles));
// Each line contains a profile, forced to be realistic.
// This trick is harmless and useful
// to fill in the instantaneous gain matrix.

instant_gain=zeros(MM,NN);
```

```
for uu=1:NN do
  instant_gain(:,uu)=Admissible(:,uu) .* ...
                   richness(Forced_Profiles(Future_Integer(:,uu),uu),uu)+ ...
                   (1-Admissible(:,uu))*bottom;
end
// When the control uu is admissible,
// instant_gain is the richness of the top block of column uu.
// When the control uu is not admissible, instant_gain
// has value "bottom", approximation of -infinity.
```

Dynamic programming algorithm

Copy the following code in a file "mine_DP_algorithm.sce"

```
// -----------------------------------------
// DYNAMIC PROGRAMMING ALGORITHM
// -----------------------------------------

VV=zeros(MM,TT);// value functions in a matrix
// The final value function is zero.
UUopt=(NN+1)*ones(MM,TT);// optimal controls in a matrix

for t=(TT-1):(-1):1 do
  loc=[];
  // will contain the vector to be maximized
  loc_bottom=mini(VV(:,t+1));
  // The value attributed to the value function VV(:,t)
  // when a control is not admissible.
  //
  for uu=1:NN do
    loc=[loc, ...
         Admissible(:,uu) .* ...
         (discount^t*instant_gain(:,uu)+VV(Future_Integer(:,uu),t+1)), ...
         (1-Admissible(:,uu)) .*(discount^t*bottom+loc_bottom)];
  end
  // When the control uu is admissible,
  // loc is the usual DP expression.
  // When the control uu is not admissible,
  // loc is the  DP expression
  // with both terms at the lowest values.
  //
  loc=[loc,VV(:,t+1)+discount^t*0];
  // Adding an extra control/column which provides zero
  // instantaneous gain and does not modify the state:
  // retire option.
  //
  [lhs,rhs]=maxi(loc,"c");// DP equation
  VV(:,t)=lhs;
  UUopt(:,t)=rhs;
  //
  UUopt(Stop_Integers,t)=(NN+1)*ones(Stop_Integers);
  // retire option
end
```

Visualization

Copy the following code in a file "`mine_visualize.sce`"

```
// exec mine_visualize.sce

lines(0);

[HH,NN]=size(richness);

// -----------------------------------------
// Visualizing the richness matrix
// -----------------------------------------

Margin=0.01;
// Margin=0.5 ;

range=6;
range=16;
range=64;

rich_color=floor((range-1)*(maxi(richness)-richness) ./(maxi(richness)-mini(richness))+
// varies between 1 and range:
// maxi(richness) corresponds to 1
// mini(richness) corresponds to range

// To change the color convention, do
// rich_color=range-rich_color +1 ;
// gives too much color
prop=1/2;
prop=0.2;
```

```
rich_color=prop*range+(1-prop)*rich_color;


vect_richness=sort(matrix(rich_color,HH*NN));
// xset("window",10) ;xbasc(); plot(vect_richness);
// xtitle("Distribution of colors");

visual_range=range;
visual_range=2*range;// to avoid white
xset("colormap",hotcolormap(visual_range));
// table of colors

xset("window",0);
xbasc();plot2d2(-Margin+0:(NN+2),-[0,zeros(1,NN),0,0],rect = [0,-HH-Margin,NN+1,Margin])
Matplot1(rich_color,[0,-HH-Margin,NN+1,Margin]);
xtitle("Mine richness");


function display_profile(xx,time)
  // Displays graphics of mine profiles
  xset("window",time);
  xset("colormap",hotcolormap(visual_range));
  // xset("colormap",hotcolormap(64));
  xbasc();
  plot2d2(-Margin+0:(NN+2),-[0,xx(time,:),0,0],rect = [0,-HH-Margin,NN+1,Margin]);
  // Just to to set the frame
  Matplot1(rich_color,[0,-HH-Margin,NN+1,Margin]);
  // The value of the mine blocks in color.
  // Will be in background.
  plot2d2(-Margin+0:(NN+2),-[0,xx(time,:),0,0],rect = [0,-HH-Margin,NN+1,Margin]);
  e=gce();
  p=e.children;
  p.thickness=3;
endfunction
```

Optimal trajectories simulation

Copy the following code in a file "`mine_DP_trajectories.sce`"

```
// exec mine_DP_trajectories.sce

// -----------------------------------------
// OPTIMAL TRAJECTORIES
// -----------------------------------------

xx=zeros(TT,NN);
zz=zeros(TT,1);
uu=(NN+1)*ones(TT,1);
vv=0;

xx(1,:)=zeros(1,NN);
// initial profile

xset("window",999);
// xset("colormap",hotcolormap(64));
xset("colormap",hotcolormap(visual_range));
// table of colors
xbasc();plot2d2(-Margin+0:(NN+2),-[0,xx(1,:),0,0],rect = [0,-HH-Margin,NN+1,Margin])
Matplot1(rich_color,[0,-HH-Margin,NN+1,Margin]);

t=1;last_control=0;
//
while last_control < NN+1 do
  zz(t)=profile2integer(xx(t,:));
  uu(t)=UUopt(zz(t),t);
  xx(t+1,:)=xx(t,:);
  if uu(t) < NN+1 then
    xx(t+1,uu(t))=xx(t,uu(t))+1;
    vv=vv+discount^t*richness(xx(t+1,uu(t)),uu(t));
  end
  //  halt()
  //  xbasc();
  //  Matplot1(rich_color,[0,-HH-Margin,NN+1,Margin]);
  //  plot2d2(-Margin + 0:(NN+2),-[0 xx(t+1,:) 0 0],...
  //                                  rect=[0,-HH-Margin,NN+1,Margin]);
  //  e=gce();
  //  p=e.children;
  //  p.thickness=3;
  // xtitle("Optimal extraction profile for annual interest rate "...
  //         +string(interest)+"%")
  //  xpause(400000);
  //
  last_control=uu(t);
  t=t+1;
end
```

```
//xtitle("Optimal extraction profile for annual interest rate "...
//       +string(interest)+"%")

final_time=t;

time=1+final_time;
display_profile(xx,time)
xtitle("Optimal final extraction profile at time t="+string(time-1)+ ...
       " for annual interest rate "+string(interest)+"%")

for time=1:final_time do
  display_profile(xx,time)
  xtitle("Optimal extraction profile at time t="+string(time)+" for annual interest rate
         string(interest)+"%")
  xs2eps(time,'optimal_'+string(time)+'.eps');
end
```

Question 2 *Copy the following code in a file* "mine_DP.sce"*, then execute it. Change the economic model and the discount rate.*

```
// exec mine_DP.sce

lines(0);

stacksize(2*10^8);
// try higher values

exec data.sce

// ------------------------------------------
// DYNAMIC PROGRAMMING ALGORITHM
// ------------------------------------------

exec mine_DP_label.sce
exec mine_DP_algorithm.sce
// do it once, then comment the above line

// ------------------------------------------
// DYNAMIC PROGRAMMING STRATEGY TRAJECTORIES
// ------------------------------------------

exec mine_visualize.sce
exec mine_DP_trajectories.sce
```

References

[1] M. De Lara and L. Doyen. *Sustainable Management of Natural Resources. Mathematical Models and Methods*. Springer-Verlag, Berlin, 2008.

[2] G.C. Goodwin, M.M. Seron, R.H. Middleton, M. Zhang, B.F. Hennessy, P.M. Stone, and M. Menabde. Receding horizon control applied to optimal mine planning. *Automatica*, 42(8):1337–1342, 2006.