Programmation dynamique
Macros générales

Michel DE LARA

October 10, 2017

Contents

1 Programmation dynamique stochastique sur des entiers (cas des matrices de transition)

Le problème d'optimisation considéré ici est

$$\inf_{u_0\in\mathbb{U},\dots,u_{T-1}\in\mathbb{U}} \mathbb{E}\left(\sum_{t=0}^{T-1} L(t,x(t),u(t)) + \Phi(T,x(T))\right), \tag{1}$$

avec

1. *coût instantané*, $L : \{0,...,T-1\}\times\{1,...,n\}\times\{1,...,p\}\to\mathbb{R}$, où
   (a) $n\in\mathbb{N}^*$ est le cardinal de l'*espace d'état* $\mathbb{X} := \{1,...,n\}\subset\mathbb{N}^*$,
   (b) $p\in\mathbb{N}^*$ est le cardinal de l'*espace de commande* $\mathbb{U} := \{1,...,p\}\subset\mathbb{N}^*$,
   (c) $T\in\mathbb{N}^*$ est l'*horizon*,
2. *coût final*, $\Phi : \{1,...,n\}\to\mathbb{R}$,
3. *matrices de transition*, $M : \{1,...,p\}\to\mathcal{M}(n,n)$, application à valeurs dans les matrices (n,n) stochastiques, c'est-à-dire $\forall u = 1,...,p$,

$$\forall (x,x')\in\{1,...,n\}^2,\ M^u(x,x')\geq 0 \quad \text{et} \quad \forall x\in\{1,...,n\},\ \sum_{x'\in\{1,...,n\}} M^u(x,x') = 1;$$

   le terme $M^u(x,x')$ représente la probabilité que, partant de l'état x et appliquant la commande u à l'instant t, le futur état au temps $t+1$ soit x' :

$$M^u(x,x') = \mathbb{P}(x(t+1)=x' \mid x(t)=x, u(t)=u),$$

4. *fonction valeur*

$$V(t,x) := \inf_{u_t\in\mathbb{U},\dots,u_{T-1}\in\mathbb{U}} \mathbb{E}\left(\sum_{s=t}^{T-1} L(s,x(s),u(s)) + \Phi(T,x(T)) \mid x(t)=x\right), \tag{2}$$

   la fonction valeur évaluée sur l'état x à l'instant t,

5. *l'équation de la programmation dynamique*

$$V(t,x) = \min_{u\in\mathbb{U}}\left(L(t,x,u) + \sum_{x'\in\{1,...,n\}} M^u(x,x')V(t+1,x')\right) \tag{3}$$

6. *feedback optimal*

$$u^\sharp(t,x) := \arg\min_{u\in\mathbb{U}}\left(L(t,x,u) + \sum_{x'\in\{1,...,n\}} M^u(x,x')V(t+1,x')\right), \tag{4}$$

   si l'inf est atteint en un unique argument,

7. *trajectoires d'état, de commande, de coût*

$$\begin{cases} (x(0), ..., x(T-1), x(T)) \\ (u(0), ..., u(T-1)) \\ (L(0, x(0), u(0)), ..., \sum_{t=0}^{T-2} L(t, x(t), u(t)), \sum_{t=0}^{T-1} L(t, x(t), u(t)) + \Phi(T, x(T))) \end{cases} \tag{5}$$

2 Programmation dynamique stochastique sur des entiers (cas dynamique bruitée)

Le problème d'optimisation considéré ici identique avec

1. *coût instantané*,
2. *coût final*
3. *dynamique bruitée*, $F : \{0, ..., T-1\} \times \{1, ..., n\} \times \{1, ..., p\} \times \mathbb{W} \to \{1, ..., n\}$, application telle que
$$x(t+1) = F(t, x(t), u(t), w(t)) \tag{6}$$
où $w(0), \ldots, w(T-1)$ est une suite de variables aléatoires i.i.d.
4. *fonction valeur*
$$V(t, x) := \inf_{u_t \in \mathbb{U}, ..., u_{T-1} \in \mathbb{U}} \mathbb{E}\left(\sum_{s=t}^{T-1} L(s, x(s), u(s)) + \Phi(T, x(T)) \mid x(t) = x \right), \tag{7}$$
la fonction valeur évaluée sur l'état x à l'instant t, prise sur une loi produit $\mathbb{P}$ sur $\mathbb{W}^T$;
5. *l'équation de la programmation dynamique*
$$V(t, x) = \min_{u \in \mathbb{U}} \left(L(t, x, u) + \mathbb{E}_{w(t)} \left(V(t+1, F(t, x, u, w(t))) \right) \right) \tag{8}$$
6. *feedback optimal*
$$u^\sharp(t, x) := \arg\min_{u \in \mathbb{U}} \left(L(t, x, u) + \mathbb{E}_{w(t)} \left(V(t+1, F(t, x, u, w(t))) \right) \right) \tag{9}$$
si l'inf est atteint en un unique argument,
7. *trajectoires d'état, de commande, de coût*

3 Transcription en objets Scilab

1. Le **coût instantané** est une **liste** `cout_instantane` à p éléments composée de **matrices** de dimension (n, T) :

$$\texttt{cout\_instantane}(l)(i,j) = L(i,l,j-1)\,, \quad l = 1,...,p\,, \quad i = 1,...,n\,, \quad j = 1,...,T\,, \tag{10}$$

où on notera le décalage dans les indices temporels, et où

   (a) le cardinal n de l'*espace d'état* est un entier `cardinal_etat`

$$\texttt{cardinal\_etat} = \texttt{size(cout\_instantane(1))(1)} = n\,, \tag{11}$$

   (b) le cardinal p de l'*espace de commande* est un entier

$$\texttt{cardinal\_commande} = \texttt{size(cout\_instantane)} = p\,, \tag{12}$$

   (c) l'*horizon* T est un entier `horizon`

$$\texttt{horizon} = 1 + \texttt{size(cout\_instantane(1))(2)} = T \tag{13}$$

2. le **coût final** est un **vecteur** `cout_final` de dimension $(n, 1)$:

$$\texttt{cout\_final}(i) = \Phi(i)\,, \quad l = 1,...,p\,, \tag{14}$$

3. Transitions

   (a) les **matrices de transition** sont une **liste** `mat_transition` à p éléments composée de **matrices** (n, n) stochastiques :

$$\texttt{mat\_transition}(l)(i,i') = M^l(i,i')\,, \quad l = 1,...,p\,, \quad i = 1,...,n\,, \quad i' = 1,...,n\,, \tag{15}$$

   (b) la *dynamique bruitée* est une fonction `dynamique`(t,i,l,w)

4. la **fonction valeur** est une **matrice** `valeur` de dimension $(n, T+1)$:

$$\texttt{valeur}(i,j) = V(i,j-1)\,, \quad i = 1,...,n\,, \quad j = 1,...,T+1\,, \tag{16}$$

où on notera le décalage dans les indices temporels,

5. le **feedback optimal** est une **matrice** `feedback` de dimension (n, T) :

$$\texttt{feedback}(i,j) = u^\sharp(i,j-1)\,, \quad i = 1,...,n\,, \quad j = 1,...,T\,, \tag{17}$$

où on notera le décalage dans les indices temporels,

6. les **trajectoires d'état, de commande, de coût** sont des **vecteurs** `trajectoire_etat`, `trajectoire_commande` et `trajectoire_cout`, de dimensions respectives $(1, T+1)$, $(1, T)$ et $(1, T+1)$:

$$\begin{cases} \texttt{trajectoire\_etat}(j) & = x(j-1)\,, & j = 1, ..., T+1 \\ \texttt{trajectoire\_commande}(j) & = u(j-1)\,, & j = 1, ..., T \\ \texttt{trajectoire\_cout}(j) & = \displaystyle\sum_{t=0}^{j-1} L(x(t), u(t), t)\,, & j = 1, ..., T \\ \texttt{trajectoire\_cout}(T+1) & = \displaystyle\sum_{t=0}^{T} L(x(t), u(t), t) + \Phi(x(T)) & \end{cases} \tag{18}$$

4 Macros Scilab pour la résolution numérique (cas des matrices de transition)

4.1 Résolution de l'équation de la programmation dynamique stochastique

L'astuce employée ici consiste à utiliser un indice dans une matrice pour représenter l'état.

```
function [valeur,feedback]=Bell_stoch(matrice_transition,cout_instantane,cout_final, ...
                                      miniOUmaxi)
  // algorithme de programmation dynamique stochastique
  //     pour une chaîne de Markov sur \{1,2...,cardinal_etat\}
  // matrice_transition = liste à cardinal_commande éléments
  //     composée de matrices de dimension (cardinal_etat,cardinal_etat)
  // cout_instantane = liste à cardinal_commande éléments
  //     composée de matrices de dimension (cardinal_etat,horizon-1)
  // cout_final = vecteur de dimension (cardinal_etat,1)
  // valeur = fonction valeur de Bellman,
  //     matrice de dimension (cardinal_etat,horizon)
  // feedback = commande en feedback sur l'état et le temps
  //     matrice de dimension (cardinal_etat,horizon-1)
  // MINIMISATION si miniOUmaxi=-1
  // MAXIMISATION si miniOUmaxi=1

  ee=size(cout_instantane(1));cardinal_etat=ee(1);
  cardinal_commande=size(cout_instantane);
  hh=size(cout_instantane(1));horizon=1+hh(2);

  valeur=0*ones(cardinal_etat,horizon);
  // tableau ou l'on stocke la fonction de Bellman
  // valeur(:,t) est un vecteur
  // : représente ici le vecteur d'état

  valeur(:,horizon)=cout_final;
  // La fonction valeur au temps T est cout_final

  feedback=0*ones(cardinal_etat,horizon-1);
  // tableau ou l'on stocke le feedback u(x,t)

  // Calcul rétrograde de la fonction de Bellman et
  // de la commande optimale à la date 'temp' par l'équation de Bellman
  // On calcule le coût pour la commande choisie connaissant la valeur de la
  // fonction coût à la date suivante pour l'état obtenu
```

```
  for temp=horizon:-1:2 do
    // équation rétrograde
    // (attention, pour des questions d'indices, bien finir en :2)
    loc=zeros(cardinal_etat,cardinal_commande);
    // variable locale contenant les valeurs de la fonction à minimiser
    // loc est une matrice
    for j=1:cardinal_commande do
      loc(:,j)=matrice_transition(j)*valeur(:,temp)+cout_instantane(j)(:,temp-1);
    end;

    [mmn,jjn]=mini(loc,'c');
    [mmx,jjx]=maxi(loc,'c');
    // mm est l'extremum atteint
    // jj est la commande qui réalise l'extremum

    valeur(:,temp-1)=(1-miniOUmaxi)/2*mmn+(1+miniOUmaxi)/2*mmx;
    // mm;
    // coût optimal
    feedback(:,temp-1)=(1-miniOUmaxi)/2*jjn+(1+miniOUmaxi)/2*jjx;
    //jj;
    // feedback optimal
  end
endfunction
```

4.2 Simulation de trajectoires optimales

```
function z=trajopt(matrice_transition,feedback,cout_instantane,cout_final,etat_initial)
  // z est une liste :
  // z(1) est la trajectoire optimale obtenue partant de etat_initial
  // z(2) est la trajectoire correspondante des commandes optimales
  // z(3) est la trajectoire correspondante des coûts
  // etat_initial = un entier
  // pour le reste, voir la macro Bell_stoch

  ee=size(cout_instantane(1));cardinal_etat=ee(1);
  cardinal_commande=size(cout_instantane);
  hh=size(cout_instantane(1));horizon=1+hh(2);

  z=list();
  x=etat_initial;
  u=[];
  c=[];
```

```
  for j=1:horizon-1 do
    // remplissage des trajectoires : commande, coût, état
    u=[u,feedback(x($),j)];
    c=[c,c($)+cout_instantane(u($))(x($),j)];
    mat_trans=full(matrice_transition(u($)));
    x=[x,grand(1,'markov',mat_trans,x($))];
  end
  c=[c,c($)+cout_final(x($))];

  z(1)=x;z(2)=u;z(3)=c;
endfunction
```

5 Macros Scilab pour la résolution numérique (cas dynamique bruitée)

6 Programmation dynamique stochastique sur un ensemble fini

Le problème d'optimisation est toujours (1), mais avec

1. *coût instantané*, $L : \{x^1, ..., x^n\} \times \{u^1, ..., u^p\} \times \{0, ..., T\} \to \mathbb{R}$, où
   (a) $\mathbb{X} := \{x^1, ..., x^n\}$ est l'*espace d'état*,
   (b) $\mathbb{U} := \{u^1, ..., u^p\}$ est l'*espace de commande*,
   (c) $T \in \mathbb{N}^*$ est l'*horizon*,
2. *coût final*, $\Phi : \{x^1, ..., x^n\} \to \mathbb{R}$,
3. *matrices de transition*

$$M^{u^l}(x^i, x^{i'}) = \mathbb{P}(x(t+1) = x^{i'} \mid x(t) = x^i, u(t) = u^l)\,. \tag{19}$$

On se ramène au cas des entiers en introduisant

1. *nouveau coût instantané*, $\widetilde{L} : \{1, ..., n\} \times \{1, ..., p\} \times \{0, ..., T-1\} \to \mathbb{R}$, avec

$$\widetilde{L}(i, l, t) := L(x^i, u^l, t)\,, \quad l = 1, ..., p\,, \quad i = 1, ..., n\,, \quad t = 0, ..., T-1\,, \tag{20}$$

2. *nouveau coût final*, $\widetilde{\Phi} : \{1, ..., n\} \to \mathbb{R}$, avec

$$\widetilde{\Phi}(i) := \Phi(x^i)\,, \quad i = 1, ..., n\,, \tag{21}$$

3. transition

   (a) *nouvelles matrices de transition* définies par

$$\widetilde{M}^l(i,i') := M^{u^l}(x^i, x^{i'})\,. \tag{22}$$

   (b) dynamique bruitée...

On transcrit ces données $\widetilde{L}$, $\widetilde{\Phi}$ et $\widetilde{M}$ en objets Scilab `cout_instantane`, `cout_final`, `matrice_trans`. L'exécution de la macro `Bell_stoch` retourne en particulier `feedback`.

Pour la simulation de trajectoires optimales, il suffit alors d'introduire un vecteur `etat` de dimension $(1,n)$ et tel que

$$\texttt{etat}(i) = x^i\,, \quad i = 1, ..., n\,, \tag{23}$$

et un vecteur `commande` de dimension $(1,p)$ et tel que

$$\texttt{commande}(l) = u^l\,, \quad l = 1, ..., p\,, \tag{24}$$

et d'exécuter une instruction du type

```
etat_initial=i;
// correspond à l'état original x^i
z=trajopt(matrice_trans,feedback,cout_instantane,cout_final,etat_initial)

zz=list();
zz(1)=etat(z(1));
zz(2)=commande(z(2));
zz(3)=z(3);
```

7 Problèmes de discrétisation

Les problèmes de discrétisation de l'espace d'état se posent assez naturellement quand le problème d'optimisation n'est pas formulé directement avec des matrices de transition, mais plutôt par le biais d'une *fonction de dynamique*. C'est ainsi qu'un problème d'optimisation déterministe est généralement formulé, mais aussi de nombreux problèmes d'optimisation stochastique.

7.1 Dynamique stochastique commandée

Une *dynamique commandée* est une application

$$F : \mathbb{X} \times \mathbb{U} \to \mathbb{X}\,. \tag{25}$$

Une *dynamique stochastique commandée* est une application

$$F : \mathbb{X} \times \mathbb{U} \times \Omega \to \mathbb{X} \tag{26}$$

où $\mathbb{X}$ est un espace mesurable, (Ω, μ) est un espace de probabilité, et où $F(x, u, \cdot)$ est supposée mesurable pour tous $(x, u) \in \mathbb{X} \times \mathbb{U}$.

Dans le cas où $\mathbb{X} = \{x^1, ..., x^n\}$ est un ensemble fini et où

$$F : \{x^1, ..., x^n\} \times \mathbb{U} \times \Omega \to \{x^1, ..., x^n\}\,, \tag{27}$$

il est immédiat de définir une famille de *matrices de transition* sur $\mathbb{X} = \{x^1, ..., x^n\}$ par

$$M^u(x^i, x^{i'}) = \mathbb{P}(x(t+1) = x^{i'} \mid x(t) = x^i, u(t) = u) = \mu(\omega \in \Omega \mid F(x^i, u, \omega) = x^{i'}\}\,. \tag{28}$$

7.2 Discrétisation d'un problème d'optimisation stochastique sur $\mathbb{R}$

Nous supposons donnée une *dynamique stochastique commandée*

$$F : \mathbb{R} \times \mathbb{U} \times \Omega \to \mathbb{R}\,. \tag{29}$$

Nous discrétisons l'espace d'état $\mathbb{R}$ en $\{x^1, ..., x^n\}$, avec $x^1 < \cdots < x^n$. Nous discrétisons également l'espace de commande $\mathbb{U}$ en $\{u^1, ..., u^l\}$.

Nous définissons les applications *prédécesseur* $\Pi : \mathbb{R} \to \{x^1, ..., x^n\}$ et *successeur* $\Sigma : \mathbb{R} \to \{x^1, ..., x^n\}$ par

$$\forall x \in \mathbb{R} \left\{ \begin{array}{lll} \Pi(x) & = \sup & \{x' \in \{x^1, ..., x^n\} \mid x' \leq x\} \\ \Sigma(x) & = \inf & \{x' \in \{x^1, ..., x^n\} \mid x' > x\} \end{array} \right. \tag{30}$$

avec la convention que $\sup \emptyset = x^1$ et $\inf \emptyset = x^n$.

Nous allons distribuer aléatoirement tout x vers son prédécesseur et vers son successeur par la fonction $\psi : \mathbb{R} \times [0, 1] \to \{x^1, ..., x^n\}$ définie par

$$\left\{ \begin{array}{ll} \Pi(x) = \Sigma(x) & \Rightarrow \psi(x, \omega') = \quad \Pi(x) = \Sigma(x) \\ & \\ \Pi(x) < \Sigma(x) & \Rightarrow \psi(x, \omega') = \left\{ \begin{array}{lll} \Pi(x) & \text{si} & \omega' > \dfrac{x - \Pi(x)}{\Sigma(x) - \Pi(x)} \\ & & \\ \Sigma(x) & \text{si} & \omega' \leq \dfrac{x - \Pi(x)}{\Sigma(x) - \Pi(x)}\,. \end{array} \right. \end{array} \right. \tag{31}$$

On discrétise alors la dynamique stochastique F par

$$\widehat{F} : \{x^1, ..., x^n\} \times \mathbb{U} \times (\Omega \times [0, 1], \mu \otimes d\omega') \to \{x^1, ..., x^n\} \tag{32}$$

où

$$\widehat{F}(x^i, u, \omega, \omega') = \psi(F(x^i, u, \omega), \omega')\,. \tag{33}$$

7.3 Discrétisation d'un problème d'optimisation stochastique sur $\mathbb{R}^n$

7.4 Discrétisation d'un problème d'optimisation déterministe sur $\mathbb{R}$ et mise sous forme stochastique

Nous reprenons ce qui est fait ci-dessus, mais avec F déterministe.

La dynamique commandée `dynamique_commandee` est transformée en une famille de matrices de transition `discretisation(etat,commande,dynamique_commandee)` : partant d'un état x^i et appliquant la commande u^l, on transite vers les deux voisins encadrant `dynamique_commandee`(x^i, u^l) (ou un seul si on est en deçà ou au delà de $\{x^1, ..., x^n\}$) avec une probabilité égale à la distance normalisée à ces points (ou égale à 1).

La macro Scilab suivante`predec_succes` utilise la fonction `dsearch`, pas forcément disponible sur les versions les plus anciennes de Scilab. En cas de problème, la remplacer par le code à télécharger ici.

```
function indices_image_discretisee=predec_succes(etat,image)
  //etat = vecteur ordonné
  //image = vecteur
  // indices_image_discretisee = liste
  //indices_image_discretisee(1)(i)=j
  // SSI etat(j) est le prédécesseur de image(i)
  //indices_image_discretisee(2)(i)= j
  // SSI etat(j) est le successeur de image(i)
  //indices_image_discretisee(3)(i)=
  //probabilité d'aller vers le prédécesseur de image(i)

  cardinal_etat=prod(size(etat));
  ind=dsearch(image,etat);
  // image(i) \in [etat(ind(i)), etat(ind(i)+1)[ sauf si
  // ind(i)=0, auquel cas image(i)<etat(1) ou image(i)>etat(£)
  //ind_nd=ind(image_ind);
  image_ind_h=find(image > etat($));
  ind(image_ind_h)=cardinal_etat;
  // ind(i)=0 ssi image(i)<etat(1)
  // ind(i)=n ssi image(i)>etat(£)
  image_ind_m=find(ind > 0 & ind < cardinal_etat);
  // indices du milieu : image(image_ind_m) \subset [etat(1),etat(£)]
  // ind(image_ind_m) \subset ]0,n[
  image_ind_b=find(ind==0);
  // indices du bas : image(image_ind_b) <etat(1)
  image_ind_h=find(ind==cardinal_etat);
  // indices du haut : image(image_ind_h) >etat(£)
```

```
  ind1=zeros(image);ind2=ind1;
  proba=zeros(image);

  ind1(image_ind_h)=ind(image_ind_h);
  // envoie les indices pour lesquels l'image de l'etat correspond est
  // >etat(£) vers cardinal_etat, dernier indice de etat
  ind2(image_ind_h)=ind1(image_ind_h);
  // prédécesseur = successeur = etat(£)
  proba(image_ind_h)=ones(image_ind_h);
  // probabilité 1 d'aller vers prédécesseur = successeur

  ind1(image_ind_m)=ind(image_ind_m);
  ind2(image_ind_m)=1+ind(image_ind_m);
  proba(image_ind_m)=(etat(1+ind(image_ind_m))-image(image_ind_m)) ./ ...
                     (etat(1+ind(image_ind_m))-etat(ind(image_ind_m)));

  ind1(image_ind_b)=ind(image_ind_b)+1;
  // +1 à cause de ind(image_ind_b) composé de 0
  // envoie les indices pour lesquels l'image de l'etat correspond est
  // <etat(1) vers 1, premier indice de etat
  ind2(image_ind_b)=ind1(image_ind_b);
  // prédécesseur = successeur = etat(1)
  proba(image_ind_b)=ones(image_ind_b);
  // probabilité 1 d'aller vers prédécesseur = successeur

  indices_image_discretisee=list();
  indices_image_discretisee(1)=ind1;
  indices_image_discretisee(2)=ind2;
  indices_image_discretisee(3)=proba;
endfunction


function z=egal(x,y)
  z=bool2s(x==y)
endfunction


function matrice_transition=discretisation(etat,commande,dynamique_commandee)
  matrice_transition=list()
  //etat = vecteur ordonné
  //commande = vecteur
```

```
  //dynamique_commandee = fonction(x,u)
  //matrice_transition = liste de matrices de transition
  //  sur les indices de etat
  cardinal_etat=prod(size(etat));
  for l=1:prod(size(commande)) do
    //la liste matrice_transition est indicée par les indices du vecteur commande
    image=dynamique_commandee(etat,commande(l));
    // vecteur des images du vecteur etat
    indices_image_discretisee=predec_succes(etat,image)
    indices1=indices_image_discretisee(1);
    indices2=indices_image_discretisee(2);
    probabilites=indices_image_discretisee(3);

    M1=zeros(cardinal_etat,cardinal_etat);
    M2=zeros(M1);
    for i=1:cardinal_etat do
      M1(i,indices1(i))=probabilites(i);
      M2(i,indices2(i))=1-probabilites(i);
    end
    matrice_transition(l)=M1+M2

    //coincidmoins=feval(zmoins(1,:),etat,egal)
    // coincidmoins(i,j)=1 ssi zmoins(1,i) == etat(j)
    // ssi partant de etat(i), etat(j) est visité
    //coincidplus=feval(zplus(1,:),etat,egal)
    //coincidmoins=1-sign ( abs ( log( ( exp(-zmoins(1,:)') ) * exp(etat) )));
    //coincidplus=1-sign ( abs ( log( ( exp(-zplus(1,:)') ) * exp(etat) ) ) );
    //matrice_transition(l)= coincidmoins .* ( ones(etat)' * zmoins(2,:) )' +...
    //coincidplus .* ( ones(etat)' * zplus(2,:) )'
    // affectation des probabilités
  end
endfunction
```

7.5 Discrétisation d'un problème d'optimisation stochastique sur $\mathbb{R}^N$, $N \geq 2$

On peut généraliser ce qui a été fait pour $N = 1$, en discrétisant $\mathbb{R}^N$ sur une grille finie

$$\mathbb{X} = \{x_1^1, ..., x_1^{n_1}\} \times \cdots \times \{x_N^1, ..., x_N^{n_N}\} \tag{34}$$

et en définissant des voisins $\mathcal{V}(x) \subset \mathbb{X}$ pour tout point $x \in \mathbb{R}^N$, avec des probabilités de transiter vers ces voisins.

On est alors ramené à une chaîne de Markov sur un espace fini $\mathbb{X}$, qui a la particularité

d'être un espace produit. Il reste alors à numéroter les éléments de $\mathbb{X}$ pour se ramener à un problème d'optimisation stochastique sur des entiers.

Bien sûr, en terme de programmation informatique, ceci n'est pas si simple...