– Texte –
Recuit simulé et le voyageur de commerce

Mots-clefs : Chaîne de Markov, loi stationnaire, graphe, permutation.

1 Choisir la route la plus courte

Un voyageur de commerce (de luxe et pas écolo) doit se rendre dans plusieurs villes par avion. Il connaît les distances entre toutes les villes et voudrait déterminer l'ordre dans lequel il doit faire ses visites pour minimiser la distance parcourue (et donc le coût de la tournée).

On numérote les villes de 1 à d. On notera $\delta(a,b)$ la distance entre les villes a et b. Ces distances sont connues. À une tournée dans les d villes, on associe (bijectivement) une permutation circulaire σ de $\{1,\dots,d\}$. On note f l'application qui à une permutation σ associe

$$f(\sigma)=\sum_{i=0}^{d-1}\delta(\sigma^i(1),\sigma^{i+1}(1)).$$

La quantité $f(\sigma)$ représente la longueur totale de la tournée dans l'ordre spécifié par σ, avec retour au point de départ. Le problème est donc de trouver une permutation σ qui minimise $f(\sigma)$. Il s'agit d'un problème d'optimisation sur l'ensemble E des permutations de d éléments. Il est certes fini mais devient vite très grand avec d. Évaluer f sur toutes les permutations prendrait un temps immense, en tout cas bien supérieur à la vie du représentant de commerce... Nous allons proposer ici un algorithme probabiliste dit du recuit simulé qui va permettre de répondre à la question de manière approchée en un temps beaucoup plus court.

Le texte est organisé de la façon suivante. La section 2 introduit la notion de graphe sur un ensemble qui permettra de construire des marches aléatoires sur E dans la section 3. La section 4 décrit la classe des mesures dites de Gibbs sur E qui joueront un rôle fondamental dans notre étude. La section 5 explique comment construire une chaîne de Markov admettant pour mesure invariante une mesure de Gibbs donnée. Enfin la section 6 fournit l'algorithme du recuit simulé proprement dit et un résultat de convergence.

Le fichier `aide-recuit.sci` fournit des fonctions auxiliaires qui pourront être utilisées en compléments de l'implémentation de l'algorithme du recuit simulé proprement dit qui est décrit dans la section 6.

2 Structures de graphe sur l'ensemble des permutations

On notera E l'ensemble des permutations de $\{1,\dots,d\}$. Soit A une partie de E^2 telle que si $(\sigma,\sigma')\in A$ alors $(\sigma',\sigma)\in A$. Si $(\sigma,\sigma')\in A$, on dit que σ et σ' sont voisins et l'on note $\sigma\sim\sigma'$. On a ainsi construit un graphe non orienté (E,A) où E est l'ensemble des sommets et A l'ensemble des arêtes.

Si σ et σ' sont deux éléments de E, un chemin de longueur n de σ à σ' est une suite $\sigma_0=\sigma,\sigma_1,\dots,\sigma_{n-1},\sigma_n=\sigma'$ de points de E tels que deux points consécutifs sont voisins.

Dire que le graphe est connexe signifie que pour tout couple de points de E, il existe un chemin de l'un vers l'autre.

La distance de σ à σ' est la plus petite longueur d'un chemin entre σ et σ'.

Le diamètre du graphe est la plus grande distance entre deux points de E. On le notera D.

Exemple 2.1 (Exemple fondamental)**.** Les structures que nous considérerons dans la suite seront d'une forme très particulière. Soit $k \in \{2, \dots, d\}$. On définit l'ensemble d'arêtes A_k de la manière suivante : $(\sigma, \sigma') \in A_k$ s'il existe une permutation η de k éléments telle que $\sigma = \sigma' \circ \eta$. Si $k = 2$, deux permutations sont voisines si et seulement si elles sont égales à une transposition près. Si $k = d$, deux permutations sont toujours voisines, on dit que le graphe est totalement connecté. Dans le graphe (E, A_k), $\sigma \in E$ possède $k!$ voisins.

3 Marches aléatoires sur un graphe

Sur (E, A), on peut définir la marche aléatoire simple $(Y_n)_{n\in\mathbb{N}}$ qui est la chaîne de Markov sur E qui passe de σ à un de ses voisins de manière équiprobable. Plus précisément, la loi de $(Y_n)_{n\in\mathbb{N}}$ est caractérisée par sa loi initiale et sa matrice de transition Q :

$$\forall (\sigma, \sigma') \in E^2, \quad Q_{\sigma,\sigma'} := \mathbb{P}(Y_{n+1} = \sigma' | Y_n = \sigma) = \begin{cases} \dfrac{1}{\#\{\eta,\ \eta \sim \sigma\}} & \text{si } \sigma' \sim \sigma, \\ 0 & \text{sinon.} \end{cases}$$

Définition 3.1. On dit que la mesure ν sur E est réversible pour la matrice de transition Q si

$$\forall (\sigma, \sigma') \in E^2, \quad \nu_\sigma Q_{\sigma,\sigma'} = \nu_{\sigma'} Q_{\sigma',\sigma}.$$

Proposition 3.2. *Pour tout $k \in \{2, \dots, d\}$, la marche aléatoire $(Y_n)_{n\in\mathbb{N}}$ sur le graphe (E, A_k) est une chaîne de Markov irréductible, récurrente, de mesure réversible la probabilité uniforme sur E.*

Remarque 3.3. Si ν est réversible pour Q, alors elle est invariante.

4 Mesures de Gibbs

4.1 Définition et propriétés essentielles

Définition 4.1. On appelle *mesure de Gibbs* associée à la fonction d'énergie f à température $T > 0$ la loi de probabilité $\nu^T = (\nu_\sigma^T)_{\sigma\in E}$ telle que

$$\forall \sigma \in E, \quad \nu_\sigma^T = \frac{1}{Z(T)} \exp\left(-\frac{1}{T} f(\sigma)\right),$$

où la constante de normalisation est

$$Z(T) = \sum_{i\in E} \exp\left(-\frac{1}{T} f(i)\right). \tag{1}$$

La proposition suivante explicite le comportement de la suite de mesures ν^T en basse et haute températures : dans le premier cas elle converge vers la mesure uniforme sur les états de plus basse énergie, dans le second elle converge vers la mesure uniforme sur E.

Proposition 4.2. *Si $E_{min} = \{i \in E \ :\ \forall j \in E,\ f(i) \leqslant f(j)\}$, alors pour tout $\sigma \in E$:*

$$\lim_{T\to 0^+} \nu_\sigma^T = \frac{1}{|E_{min}|} \mathbb{1}_{E_{min}}(\sigma) \quad \textit{et} \quad \lim_{T\to\infty} \nu_\sigma^T = \frac{1}{|E|}.$$

Remarque 4.3. Revenons un instant au problème initial de trouver les chemins les plus courts. Au vu de la proposition ci-dessus, on peut être tenté de définir l'énergie d'un chemin (*i.e* d'une permutation) comme sa longueur totale, puis, pour localiser les minimima de f, de chercher à simuler des réalisations

de la loi ν^T pour une température suffisamment basse. Les valeurs de f sur les états obtenus par une telle simulation seront proches de la valeur minimale de f sur E. Bien entendu, pour des raisons de taille de E, simuler directement une réalisation de ν^T est impossible en pratique. En effet, cela demanderait en particulier le calcul de la constante de normalisation $Z(T)$ de la définition 4.1, ce qui est numériquement impossible [1]. La section 5 va expliquer comment contourner cette difficulté.

4.2 Un peu de physique

Cette section peut être omise sans préjudice pour la suite.

Définition 4.4. L'entropie de la loi de probabilité $\mu = (\mu_i)_{i\in E}$ est définie par

$$H(\mu) = -\sum_{i\in E} \mu_i \log \mu_i,$$

avec la convention $0 \log 0 = 0$.

Lemme 4.5. *L'entropie est toujours positive, maximale pour la loi uniforme sur E et minimale pour les mesures de Dirac.*

L'entropie est une mesure de l'incertitude, de l'aléa contenu dans la loi μ. En physique, à chaque état i d'un système est associé une énergie, représentée dans notre problème par $f(i)$. Un grand principe de la physique prédit que les distributions stables sont celles dont l'entropie est maximale. La distribution μ sur E possède une énergie moyenne égale à

$$\mathbb{E}_\mu(f) = \sum_{i\in E} f(i)\mu_i.$$

Si l'on impose que cette énergie soit constante égale à $\mathcal{E}$, peut-on trouver les distributions qui maximisent l'entropie ? Pour qu'il en existe, il faut bien sûr que $\mathcal{E} \in [\min f, \max f]$. Supposons de plus que f ne soit pas constante (sinon la question n'est pas très intéressante). Dans ce cas, il existe une seule mesure de probabilité répondant à la question, elle est appellée loi de *Boltzmann*.

Lemme 4.6. *Si f n'est pas constante, alors, à énergie moyenne constante égale à $\mathcal{E} \in]\min f, \max f[$, la mesure de probabilité sur E qui maximise l'entropie est de la forme*

$$\forall i \in E, \quad \mu_i = \frac{1}{Z_a} \exp(-af(i)),$$

où $Z_a = \sum_{j\in E} \exp(-af(j))$ et a est l'unique solution de $\mathcal{E} = \sum_{i\in E} f(i)e^{-af(i)}/Z_a$.

Démonstration. Pour déterminer les extrema de la fonction $\mu \mapsto H(\mu)$ sous les contraintes $\sum_{i\in E} \mu_i = 1$ et $\mathbb{E}_\mu(f) = \mathcal{E}$, on peut appliquer le théorème des extrema liées. Celui-ci assure que les extrema sont de la forme

$$\forall i \in E, \quad \mu_i = Ce^{-af(i)}$$

où C et a sont des réels. Le fait que μ soit une mesure de probabilité impose $C = Z_a$. Enfin, la fonction $a \mapsto \sum_{i\in E} f(i)e^{-af(i)}/Z_a$ est une bijection strictement monotone de $\mathbb{R}$ dans $[\min f, \max f]$. □

1. $|E|$ est souvent très grand, de l'ordre de 20! par exemple et chaque terme de la somme (1) est très petit ce qui rend très long et très instable l'évaluation de $Z(T)$.

5 Algorithme de Métropolis

5.1 Une chaîne de Markov associée aux mesures de Gibbs

Le théorème suivant est excessivement malin. Étant donnée une structure de graphe sur E, la fonction d'énergie f et une température T, il définit une chaîne de Markov, réversible par rapport à la mesure de Gibbs ν^T, et dont la matrice de transition s'exprime très simplement en fonction de f et T et ne requiert pas le calcul de $Z(T)$.

Remarque 5.1. Il s'agit d'un cas particulier de l'algorithme de Metropolis qui est décrit de manière plus générale dans le paragraphe 5.2.

Theorème 5.2. *Soit* $P^T = (p^T_{\sigma,\sigma'})_{\sigma,\sigma'\in E}$ *la matrice de transition sur* E *définie par : pour* $\sigma \neq \sigma'$,

$$p^T_{\sigma,\sigma'} = \begin{cases} \dfrac{1}{\#\{\eta,\ \eta\sim\sigma\}} \min\left\{\exp((f(\sigma)-f(\sigma'))/T), 1\right\} & \text{si } \sigma'\sim\sigma, \\ 0 & \text{sinon,} \end{cases} \tag{2}$$

les coefficients diagonaux étant choisis pour que P^T *soit une matrice markovienne. Alors la mesure* ν^T *est réversible pour* P^T*. De plus la chaîne de Markov associée, notée* $(X_n)_{n\geqslant 0}$ *dans la suite, est irréductible, récurrente et apériodique.*

On déduit de ce résultat un algorithme de simulation de la chaîne de Markov de matrice de transition P^T (la fonction **Random** fournit une réalisation de loi uniforme sur $[0,1]$) :

```
1     Initialiser X
2     n <-- 0
3     Répéter
4       i <-- X
5       Choisir j au hasard parmi les voisins de i
6       Si (f(j)<=f(i))
7         alors X <-- j
8         sinon
9           Si (Random<exp((f(i)-f(j))/T)) alors X <-- j
10          finSi
11      FinSi
12      n <-- n+1
13    Jusqu'à (arrêt de la simulation)
```

On peut voir cet algorithme comme une sorte de descente de gradient aléatoire, dont les pas qui vont vers les basses valeurs de la fonction sont automatiquement effectués. S'il n'y avait que ceux-là, l'algorithme resterait piégé dans les minima locaux [2]. Pour sortir de ces pièges, on autorise l'algorithme à effectuer éventuellement des pas dans la mauvaise direction : lorsque le voisin tiré au sort correspond à une énergie plus grande, il n'est choisi qu'avec une certaine probabilité et dans le cas contraire, le processus reste sur place.

Remarque 5.3. Lorsque le paramètre de température T est grand, la chaîne explore bien l'espace (la chaîne sautera souvent) mais ne reste donc pas sur les minima de f. Au contraire, lorsque T est très petit, la chaîne se concentre sur les minima de f mais l'exploration est très lente : il faudra beaucoup de temps pour sortir d'un voisinage d'un minimum local. Il s'agit de tirer avantage des deux comportements en neutralisant les inconvénients de chacun.

2. Un point i est un minimal local si $f(i) \leqslant f(j)$ pour tout j voisin de i.

5.2 Le cas général

Cette section peut être omise sans préjudice pour la suite. La construction de P^T est un cas particulier de l'algorithme de Metropolis.

Proposition 5.4. *Soit $Q = (q_{ij})_{i,j\in E}$ une matrice de transition sur E telle que*

$$\forall (i,j) \in E^2, \quad q_{ij} > 0 \Rightarrow q_{ji} > 0.$$

Soit μ une loi de probabilité chargeant tous les points de E. Définissons sur E une matrice de transition $P = (p_{ij})_{i,j\in E}$ de la façon suivante : pour $i \neq j$,

$$p_{ij} = \begin{cases} q_{ij} \min\left\{ \dfrac{\mu_j q_{ji}}{\mu_i q_{ij}}, 1 \right\} & \text{si } q_{ij} \neq 0, \\ 0 & \text{sinon,} \end{cases}$$

les coefficients diagonaux étant choisis pour que P soit une matrice markovienne. Alors la mesure μ est réversible pour P.

Remarque 5.5. Il est capital de noter que μ n'intervient dans la définition de P que par l'intermédiaire des quantités μ_j/μ_i. Ainsi, il suffit de connaître μ à un facteur de normalisation près...

6 Algorithme du recuit simulé et résultat de convergence

La méthode du recuit simulé tire son nom et son inspiration de la physique des matériaux et plus spécialement de la métallurgie ! Le *recuit* est une opération consistant à laisser refroidir lentement un métal pour améliorer ses qualités. L'idée physique est qu'un refroidissement trop brutal peut bloquer le métal dans un état peu favorable (alors qu'un refroidissement lent permettra aux molécules de s'agencer au mieux dans une configuration stable). C'est cette même idée qui est à la base du recuit simulé. Pour éviter que l'algorithme ne reste piégé dans des minima locaux, on fait en sorte que la température $T = T(n)$ décroisse lentement en fonction du temps.

On peut montrer que la décroissance de T doit être de type logarithmique. Le théorème ci-dessous montre que l'on peut également choisir T constante par morceaux sur des paliers de longueur exponentielle.

Theorème 6.1. *Soit $h > 0$. On choisit T constante par morceaux :*

$$\forall k \in \mathbb{N}^*, \quad \forall n \in]e^{(k-1)h}, e^{kh}], \quad T(n) = \frac{1}{k}.$$

Il existe $h^* > 0$ tel que pour tout $h > h^*$, l'algorithme de recuit converge, c'est-à-dire que

$$\lim_{n\to\infty} \mathbb{P}(X_n \in E_{min}) = 1.$$

Remarque 6.2. La constante h^* s'exprime en fonction de la profondeur des pièges constitués par les minima locaux qui ne sont pas globaux. Elle est donc en pratique inconnue. Le paramètre h est donc un des *boutons de réglage* de l'algorithme qui doit être judicieusement positionné par l'utilisateur : ceci est fait au jugé... S'il est choisi trop petit, l'algorithme peut ne pas converger vers les minima globaux et rester piégé dans un minimum local. S'il est trop grand, la convergence sera très lente.

Remarque 6.3. Un autre bouton de réglage est la structure de graphe choisie sur l'espace. Pour le problème du voyageur de commerce, plusieurs choix sont possibles, notamment les graphes (E, A_k). On pourra opter pour une structure de voisinage qui permette une exploration rapide de l'espace à condition que cela n'alourdisse pas trop l'algorithme.

Voici à présent le cœur de l'algorithme du recuit simulé. Il présuppose connus les éléments suivants :
— un paramètre h pré-défini qui gouverne les paliers de la fonction température [3],
— la fonction **Random** qui fournit une réalisation de la loi uniforme sur $[0,1]$,
— la fonction [4] **Permuter** qui associe à une permutation une permutation aléatoire voisine au sens du graphe choisi,
— la fonction [5] **f** qui associe à une permutation la longueur du circuit à partir de la matrice des distances entre toutes les villes.

```
(Initialisations)
UnsurT ← 0
ordre ← permutation aléatoire de {1, ..., d}
f1 ← f(ordre)
n ← 0
(Boucle principale)
Répéter
  UnsurT ← UnsurT+1
  Palier ← exp(UnsurT * h)
  Répéter
    ordre2 ← permuter(ordre)
    f2 ← f(ordre2)
    Si (f2⩽f1)
      Alors
        ordre ← ordre2
        f1 ← f2
      Sinon
        proba ← exp((f1-f2) * UnsurT)
        Si (Random<proba) alors
          ordre ← ordre2
          f1 ← f2
      FinSi
    FinSi
    n ← n+1
  Jusqu'à (n⩾Palier)
Jusqu'à (arrêt de la simulation)
```

7 Suggestions

1. On pourra présenter la problématique du voyageur de commerce.
2. On pourra démontrer les proposition 4.2 et 3.2.
3. On pourra démontrer les lemmes 4.5 et 4.6.
4. On pourra démontrer le théorème 5.2.
5. On pourra décrire, théoriquement et/ou par la simulation, le comportement asymptotique de la chaîne de Markov de matrice de transition P^T définie par (2).
6. On pourra expliquer (heuristiquement) l'idée du recuit simulé.

3. Comme elle apparaît au dénominateur et qu'elle est de la forme $1/k$, on a choisi d'utiliser le paramètre $1/T$ que l'on nomme UnsurT.
4. Voir fichier aide-recuit.sci ci-joint.
5. Voir fichier aide-recuit.sci ci-joint.

7. On pourra implémenter l'algorithme du recuit simulé et illustrer son efficacité. On utilisera les fonctions annexes **Permuter**, **f**, **Initialise** et **Traceparcours** fournies dans le fichier `aide-recuit.sci`.