CM-TD-TP-08 - Agrégation de Mathématiques - Méthode de Métropolis - Recuit simulé- 2007/2008.

On cherche à simuler un champs de Gibbs noté

$$\pi(x) = \frac{1}{Z} e^{-\beta \mathcal{E}(x)}$$

où Z est une constante de normalisation inconnue (inaccessible numériquement) et $\mathcal{E}$ une énergie quantifiant l'état d'un système, tandis que β est une constante positive. L'approche va consister à construite une chaîne de Markov $(X_n)_{n\in\mathbb{N}}$ de mesure invariante π. On fera l'hypothèse simplificatrice que tout état x appartient à E fini. Il est in-envisageable de simuler directement une telle distribution pour plusieurs raisons :
– La plupart du temps, l'espace d'états est de grande taille et on ne peut calculer $\mathcal{E}$ en tout point.
– La constante de normalisation Z ne peut être calculée exactement.

1 Algorithme de Metropolis

On suppose donnée une matrice de transition P sur E vérifiant l'hypothèse de *Doeblin*

$$\exists l \geq 1, \exists \alpha > 0, \exists c : E \mapsto \mathbb{R} \quad \forall (x,y) \in E^2 \qquad P^l(x,y) \geq \alpha c(y) \tag{1}$$

où c est une distribution de probabilités sur E. On impose en plus l'hypothèse de réversibilité de la chaîne

$$P(x,y) > 0 \Longleftrightarrow P(y,x) > 0 \tag{2}$$

La stratégie adoptée consiste alors à :
– On tire X_0 quelconque dans E.
– Étant donné une réalisation $X_n = x_n$, on tire $Y_{n+1} = y_{n+1}$ selon la loi $P(x_n, .)$.
– On calcule le quotient

$$\rho_n = \frac{\pi(y_{n+1})P(y_{n+1}, x_n)}{\pi(x_n)P(x_n, y_{n+1})}$$

– On choisit alors $X_{n+1} = y_{n+1}$ avec la probabilité $\min(1, \rho_n)$ et sinon on pose $X_{n+1} = x_n$.

1. Montrer que le processus discret (X_n) ainsi défini est une chaîne de Markov.
2. En notant Q la probabilité de transition (homogène) de la chaîne de Markov, démontrer qu'en fait, la probabilité Q est réversible par rapport à π, *i.e* :

$$\pi(x)Q(x,y) = \pi(y)Q(y,x)$$

3. En déduire que la distribution π est invariante pour Q.
4. Dans le cas d'une distribution π donnée par le champ de Gibbs précédent, et dans le cas où la matrice de transition P est symétrique, donner une expression simple de la matrice de transition Q de $(X_n)_{n\in\mathbb{N}}$.
5. Vérifier que si P satisfait une condition du type (**??**), il en est de même pour Q.
6. On définit la norme en variation totale de la façon suivante :

$$\forall (\mu,\nu) \in \mathcal{P}(E) \qquad \|\nu - \mu\| = \sum_{y\in E} |\nu(y) - \mu(y)|$$

Démontrer que toute chaîne de Markov de matrice de transition P vérifiant (**??**) converge en variation vers une mesure de probabilité π unique probabilité invariante de la chaîne.

7. Conclure.

L'objectif d'un tel algorithme est d'obtenir une minimisation asymptotique d'une énergie $\mathcal{E}$. L'idée est de considérer une famille de lois de probabilités sur E qui « convergent »vers des masses de Dirac en les minima de $\mathcal{E}$. Les lois de probabilités sont exactement les π_β introduites dans le précédent paragraphe. On notera m le minimum de $\mathcal{E}$ (pas nécessairement unique).

2.1 Convergence et Algorithme

1. Démontrer que lorsque β tend vers $+\infty$, les distributions π_β chargent principalement les minima de $\mathcal{E}$. On étudiera pour cela la limite :

$$\lim_{\beta \mapsto +\infty} \pi_\beta \left(\mathcal{E} > m + \epsilon \right)$$

où m est le minimum de $\mathcal{E}$ et $\epsilon > 0$ quelconque.

2. Le point difficile est alors d'établir que pour toute loi ν_0 de X chaîne de Markov (non stationnaire) simulée par l'algorithme de Métropolis, on a la convergence de la distribution au temps n vers la distribution « limite »de π_β. Plus précisément, dans les hypothèses (1) et (2), on admettra que :

$$\lim_{n \mapsto +\infty} \| \nu_n - \pi_{\beta_n} \| = 0$$

avec $\beta_n = \gamma \log n$ et ν_n loi de la chaîne de Markov à l'instant n simulée par l'algorithme de Métropolis. En déduire l'écriture d'un algorithme assurant la convergence de X vers le minimum de $\mathcal{E}$.

2.2 Une application

L'objectif de ce TP est de programmer en MATLAB une implémentation du Recuit Simulé adapté au problème du Voyageur de Commerce. Un voyageur de commerce doit visiter N clients situés dans N villes différentes, puis revenir à son point de départ. Pour plus de simplicité de programmation, on fera l'étude où le voyageur de commerce n'est pas obligé de revenir à son point initial, mais il est également possible d'envisager un parcours bouclé.

1. En ne considérant uniquement que des distances euclidiennes entre point du plan, poser le problème du voyageur de commerce comme un problème d'optimisation.
2. Expliquer pourquoi le problème se ramène à optimiser une fonction sur l'ensemble des permutations σ_N. Quel problème algorithmique se pose alors ?
3. Écrire une fonction Matlab permettant de générer aléatoirement un nombre quelconque n de villes dans le plan représenté par le carré de côté 1. La sortie du code pourra être, par exemple, un vecteur v de taille $n \times 2$: $v =$ *initialise(n)*.
4. En identifiant l'itinéraire effectué par le voyageur de commerce comme étant l'ordre des coordonnées dans le vecteur v, écrire un programme Matlab permettant de calculer la distance associé à l'itinéraire ainsi défini. Cette fonction prendra en argument un vecteur v de la forme précédente. On notera cette fonction *d= dist(v)*.
5. On note $P(x, y)$ la matrice de transition utilisée pour construire l'algorithme de Recuit Simulé pour x et y deux permutations de σ_n.
   - Rappeler quelles sont les propriétés devant être satisfaites par P pour assurer la convergence de l'algorithme.
     On rappelle que la formule d'acceptation de transition est donnée par

$$\min \left(1, \frac{\pi(y) P(y, x)}{\pi(x) P(x, y)} \right)$$

     où π est une loi de probabilité que l'on précisera.
   - Quelle est la dimension de P ?
6. Dans la phase cruciale de construction de cette matrice P, on choisit de définir une relation de voisinage sur les itinéraires en imposant que deux itinéraires sont voisins si et seulement si ils sont identiques modulo une permutation de 2 villes dans l'itinéraire. Décrire alors les voisinages obtenus sur σ_n.
7. Voici comment on construit P : partant d'un itinéraire x, on choisit soit un itinéraire voisin différent uniformément parmi l'ensemble de tous les itinéraires voisins, soit de garder le même itinéraire avec probabilité $1/n$. Expliciter précisément $P(x, y)$.

stockées dans v, renvoie 2 entiers candidats i et j décrivant la proposition de modification d'itinéraire *via* P.

10. Écrire un programme *newv=perm(v,i,j)* construisant le nouvel itinéraire newv à partir de la liste v et les deux entiers précédemment proposés i et j.
11. Programmer enfin la fonction *[f,v]=recuit(n,niter,gamma)* prenant comme arguments la liste initiale des villes v, le nombre de pas de l'algorithme niter, ainsi qu'un paramètre γ de température dont on expliquera le rôle. Les sorties seront par exemple f un vecteur contenant la liste des longueurs d'itinéraires successifs et v la liste des villes mises à jour dans le bon ordre, en fin d'algorithme.
12. En utilisant une liste de villes simulées par vos propres soins (cf question 4), construire le trajet obtenu par recuit optimal, représenter graphiquement l'itinéraire initial, final. Construire enfin la courbe donnant la longueur des itinéraires obtenus en fonction de l'itération de l'algorithme.
13. Si vous en avez le courage, adaptez une autre technique d'optimisation stochastique de votre choix pour résoudre ce problème.