Simulation des chaînes de Markov

Ana Bušić

INRIA - ENS

http://www.di.ens.fr/~busic/

ana.busic@inria.fr

Master COSY - UVSQ

Versailles, février 2011

Rappels

Nous avons vu :

- Répresentation des chaînes de Markov par des SED.
- Simulation MCMC.
- Simulation parfaite par couplage depuis le passé.
- Cas monotone (et anti-monotone).

Rappels

Nous avons vu :

- Répresentation des chaînes de Markov par des SED.
- Simulation MCMC.
- Simulation parfaite par couplage depuis le passé.
- Cas monotone (et anti-monotone).

Quelques remarques/problèmes :

- Temps de couplage ?
- Garder la suite aléatoire en mémoire ?
- SED non-monotones ?

Temps de couplage

En général un problème très difficile.

Quelques résultats théoriques dans des files d'attente :

- Une file M/M/1/C. Pire cas : $\lambda = \mu$. $O(C^2)$.
- Réseaux de Jakson avec les capacités finies :

  Pour un réseau acyclique de K files M/M/1/C, on peut montrer que $\mathbb{E}\tau^b \leq \alpha(\lambda, \mu) K C^2$ [Dopper, Gaujal , Vincent, 2006], alors que la taille de l'espace d'état est $N = C^K$.

Read-once randomness (Wilson 1999)

Algorithme Propp & Wilson : on recommence les trajectoires en partant des temps $-N_1, -N_2, \ldots$ (avec $N_1 < N_2 < \ldots$) jusqu'à j tel que $|S_{N_j}| = 1$, où $S_{N_j} \stackrel{\text{def}}{=} \mathcal{X} \cdot a_{N_j \to 1}$.

Problème : garder la suite aléatoire en mémoire peut être très cher (dépends du temps de couplage !)

Read-once randomness (Wilson 1999)

Algorithme Propp & Wilson : on recommence les trajectoires en partant des temps $-N_1, -N_2, \ldots$ (avec $N_1 < N_2 < \ldots$) jusqu'à j tel que $|S_{N_j}| = 1$, où $S_{N_j} \stackrel{\text{def}}{=} \mathcal{X} \cdot a_{N_j \to 1}$.

Problème : garder la suite aléatoire en mémoire peut être très cher (dépends du temps de couplage !)

En pratique : garder les "seed" utilisées pour générer les sous-séquences - peut mener à toute sorte des comportements inattendus et génants (à éviter ! !)

Wilson's modification

[Wilson 1999] Idée : une sorte de couplage en avant mais où on n'arrete pas la simulation au moment même du couplage - on continue encore pendant un certain temps aléatoire.

Wilson's modification

[Wilson 1999] Idée : une sorte de couplage en avant mais où on n'arrete pas la simulation au moment même du couplage - on continue encore pendant un certain temps aléatoire.

Comment définir ce temps ?

Wilson's modification

[Wilson 1999] Idée : une sorte de couplage en avant mais où on n'arrete pas la simulation au moment même du couplage - on continue encore pendant un certain temps aléatoire.

Comment définir ce temps ?

On commence par faire quelques observations sur le couplage depuis le passé...

Observation I

Pour le cas monotone, on a utilisé les temps $(N_1, N_2, N_3, N_4...) = (1, 2, 4, 8, ...)$ mais toute suite strictement croissante marche aussi.

Observation I

Pour le cas monotone, on a utilisé les temps $(N_1, N_2, N_3, N_4 ...) = (1, 2, 4, 8, ...)$ mais toute suite strictement croissante marche aussi.

On peut même prendre une suite aléatoire strictement croissante !

Observation I

Pour le cas monotone, on a utilisé les temps $(N_1, N_2, N_3, N_4...) = (1, 2, 4, 8, ...)$ mais toute suite strictement croissante marche aussi.

On peut même prendre une suite aléatoire strictement croissante !

Preuve : soit $N_1 < N_2 < ...$ une suite aléatoire d'enties, indépendante de la suite aléatoire des événements tirés. Alors sachant $(N_1, N_2, ...)$ l'algorithme renvoie un échantillon non-biaisé de π. Cela est vrai pour toute réalisation de $(N_1, N_2, ...)$, ainsi l'algorithme (avec les temps $(N_1, N_2, ...)$ aléatoires) donne un échantillon non-biaisé de π.

Observation II

Continuer l'algorithme depuis encore plus loin dans le passé n'est pas génant !

(à part pour le temps supplémentaire d'exécution de l'algorithme)

Observation II

Continuer l'algorithme depuis encore plus loin dans le passé n'est pas génant !

(à part pour le temps supplémentaire d'exécution de l'algorithme)

Preuve : si $|S_n| = 1$ pour un $n \in \mathbb{N}$, alors pour tout $m > n$, on a $S_m = S_n$.

Choix de la suite aléatoire

Comment choisir la suite aléatoire strictement croissante $N_1 < N_2 < \ldots$?

Soit $(T_1, T_2, T_3, \ldots)$ une suite aléatoire d'entiers i.i.d. avec la distribution égale à celle du couplage en avant.

Choix de la suite aléatoire

Comment choisir la suite aléatoire strictement croissante $N_1 < N_2 < \ldots$?

Soit $(T_1, T_2, T_3, \ldots)$ une suite aléatoire d'entiers i.i.d. avec la distribution égale à celle du couplage en avant.

Question

Comment générer $(T_1, T_2, T_3, \ldots)$ *?*

Choix de la suite aléatoire

Comment choisir la suite aléatoire strictement croissante $N_1 < N_2 < \ldots$?

Soit $(T_1, T_2, T_3, \ldots)$ une suite aléatoire d'entiers i.i.d. avec la distribution égale à celle du couplage en avant.

Question

Comment générer $(T_1, T_2, T_3, \ldots)$ *?*

On fixe :

$$\begin{aligned} N_1 &= T_1 \\ N_2 &= T_1 + T_2 \\ N_3 &= T_1 + T_2 + T_3 \\ &\vdots \quad\quad \vdots \\ N_n &= N_{n-1} + T_n \end{aligned}$$

Observation III

La probabilité que l'algorithme Propp & Wilson en partant du temps initial $-N_1 = -T_1$ se termine (couplage avant le temps 0) est au moins $1/2$.

Observation III

La probabilité que l'algorithme Propp & Wilson en partant du temps initial $-N_1 = -T_1$ se termine (couplage avant le temps 0) est au moins $1/2$.

Preuve : Soit M_1 le temps nécessaire pour coupler en partant du temps $-N_1$ (et dépassant le temps 0 si besoin !)
Alors M_1 et N_1 ont la même distribution et ils sont indépendants, donc :

$$\mathbb{P}(M_1 \leq N_1) = \mathbb{P}(M_1 \geq N_1).$$

Observation III

La probabilité que l'algorithme Propp & Wilson en partant du temps initial $-N_1 = -T_1$ se termine (couplage avant le temps 0) est au moins $1/2$.

Preuve : Soit M_1 le temps nécessaire pour coupler en partant du temps $-N_1$ (et dépassant le temps 0 si besoin !)
Alors M_1 et N_1 ont la même distribution et ils sont indépendants, donc :

$$\mathbb{P}(M_1 \leq N_1) = \mathbb{P}(M_1 \geq N_1).$$

Aussi :

$$\begin{aligned}
\mathbb{P}(M_1 \leq N_1) + \mathbb{P}(M_1 \geq N_1) &= 1 - \mathbb{P}(M_1 > N_1) + 1 - \mathbb{P}(M_1 < N_1) \\
&= 2 - (\mathbb{P}(M_1 > N_1) + \mathbb{P}(M_1 < N_1)) \\
&= 2 - (\mathbb{P}(M_1 \neq N_1)) \\
&\geq 2 - 1 = 1.
\end{aligned}$$

Donc $\mathbb{P}(M_1 \leq N_1) \geq 1/2$.

Observation III - suite

On peut continuer le raisonement...

Si l'algorithme ne termine pas en partant de $-N_1$, alors avec la probabilité au moins $1/2$ on observe le couplage avant le temps $-N_1$ en partant du temps $-N_2 = -(N_1 + T_2)$. Plus généralement, avec la probabilité au moins $1/2$ on observe le couplage avant le temps $-N_{j-1}$ en partant du temps $-N_j = -(N_{j-1} + T_j)$.

Observation III - suite

On peut continuer le raisonement...

Si l'algorithme ne termine pas en partant de $-N_1$, alors avec la probabilité au moins $1/2$ on observe le couplage avant le temps $-N_1$ en partant du temps $-N_2 = -(N_1 + T_2)$. Plus généralement, avec la probabilité au moins $1/2$ on observe le couplage avant le temps $-N_{j-1}$ en partant du temps $-N_j = -(N_{j-1} + T_j)$.

On va dire qu'on a un succès à l'étape j si, en partant du temps $-N_j$, on observe le couplage avant le temps $-N_{j-1}$.

Observation III - suite

On peut continuer le raisonement...

Si l'algorithme ne termine pas en partant de $-N_1$, alors avec la probabilité au moins $1/2$ on observe le couplage avant le temps $-N_1$ en partant du temps $-N_2 = -(N_1 + T_2)$. Plus généralement, avec la probabilité au moins $1/2$ on observe le couplage avant le temps $-N_{j-1}$ en partant du temps $-N_j = -(N_{j-1} + T_j)$.

On va dire qu'on a un succès à l'étape j si, en partant du temps $-N_j$, on observe le couplage avant le temps $-N_{j-1}$.

Et pour avoir la probabilité de succès égale à $1/2$, au cas ou on observe le couplage exactement au temps $-N_{j-1}$, on tire une piece non-biaisé et avec proba $1/2$ on déclare le succès, et sinon l'échec.

Modification de Wilson

L'algorithme (juste, mais un peu artificiel !) qui consiste à recommencer depuis les temps $-N_j$ jusqu'au succès donne bien un échantillon non-biaisé de π.

Modification de Wilson

L'algorithme (juste, mais un peu artificiel !) qui consiste à recommencer depuis les temps $-N_j$ jusqu'au succès donne bien un échantillon non-biaisé de π.

Notons le nombre (aléatoire) d'échec avant le succès par Y. Alors Y est une variable aléatoire géométrique de paramètre $p = 1/2$:

$$\mathbb{P}(Y = n) = p(1-p)^n.$$

L'itération finale (et succès) commence en $-N_{Y+1}$.

Modification de Wilson

L'algorithme (juste, mais un peu artificiel !) qui consiste à recommencer depuis les temps $-N_j$ jusqu'au succès donne bien un échantillon non-biaisé de π.

Notons le nombre (aléatoire) d'échec avant le succès par Y. Alors Y est une variable aléatoire géométrique de paramètre $p = 1/2$:

$$\mathbb{P}(Y = n) = p(1-p)^n.$$

L'itération finale (et succès) commence en $-N_{Y+1}$.

Idée de modification de Wilson : trouver un moyen de simuler d'abord les trajectoires depuis $-N_{Y+1}$ jusqu'à $-N_Y$, puis de $-N_Y$ jusqu'à $-N_{Y-1}$... et cela jusqu'au temps 0.

Twin run

Executer deux copies indépendantes de couplage en avant - On s'arrête quand les deux couplent !

On déclare la copie qui a couplé d'abord le gagnant et l'autre le perdant (avec une pièce non-biaisée pour départager si égalité).

Twin run

Executer deux copies indépendantes de couplage en avant - On s'arrête quand les deux couplent !

On déclare la copie qui a couplé d'abord le gagnant et l'autre le perdant (avec une pièce non-biaisée pour départager si égalité).

Observation : l'évolution de nos trajectoires depuis le passé en partant de $-N_{Y+1}$ jusqu'à $-N_Y$ a justement la même distribution que celle du gagnant.

Algorithme

- Choisir Y selon la loi géometrique avec $p = 1/2$.
- Si $Y = 0$ on a fini. On renvoie la valeur finale du gagnant au moment de couplage du perdant.
- Sinon on doit continuer la simulation depuis $-N_Y$ jusqu'à 0. Pour simuler les trajectoires de $-N_Y$ jusqu'à $-N_{Y-1}$, on fait un autre twin run et en prend l'évolution du perdant, ou le perdant évolue de 0 jusqu'au couplage du gagnant !

  Puis on continue avec un autre twin run indépendant pour $-N_{Y-1}$ jusqu'à $-N_{Y-2}$...

  On renvoie la valeur finale du perdant du dernier twin run (au moment du couplage du gagnant du même twin run).

Le cas non-monotone

- [Kendall 1998] et [Häggström and Nelander 1998] une idée similaire pour le cas anti-monotone :

$$x \preceq y \Rightarrow x \cdot a \succeq y \cdot a, \forall a \in A.$$

- [Kendall and Møller 2000] ont introduit une première idée des processus bornants qui enveloppent toutes les trajectores.
- [Huber 2004] une idée similaire des processus pour détecter le couplage, mais sans introduire une notion d'ordre sur les états.

Envelope Perfect Sampling Algorithm

[B., Gaujal, Vincent 2008]

- Hypothèse : $(\mathcal{X}, \preceq)$ est un treillis.
- Pour $m, M \in \mathcal{X}$, $[m, M] \stackrel{\text{def}}{=} \{x \in \mathcal{X} \ : \ m \preceq x \preceq M\}$ un intervalle entre m et M.
- $\mathcal{I} = \{[m, M] \ : \ m, M \in \mathcal{X}, \ m \preceq M\}$ l'ensemble des intervalles non-vides :
- Nouvelle fonction de transition $\boxdot : \mathcal{I} \times A \to \mathcal{I}$ appellée transition enveloppe : pour tout $[m, M] \in \mathcal{I}$ et $a \in A$

$$[m, M] \boxdot a \stackrel{\text{def}}{=} \left[\inf_{m \preceq x \preceq M} \{x \cdot a\}, \ \sup_{m \preceq x \preceq M} \{x \cdot a\} \right].$$

Extention aux mots finis :
$[m, M] \boxdot u_{1 \to n} \stackrel{\text{def}}{=} [m, M] \boxdot u_1 \boxdot u_2 \boxdot \cdots \boxdot u_n$.

Envelope Perfect Sampling Algorithm

Soient $\bot \stackrel{\text{def}}{=} \inf \mathcal{X}$ et $\top \stackrel{\text{def}}{=} \sup \mathcal{X}$. Le processus $([m_n, M_n])_{n\in\mathbb{N}}$

$$[m_n, M_n] \stackrel{\text{def}}{=} [\bot, \top] \boxdot u_{1\to n}$$

est une chaîne de Markov sur l'espace d'états $\mathcal{X} \times \mathcal{X}$, appellée la chaîne enveloppe.

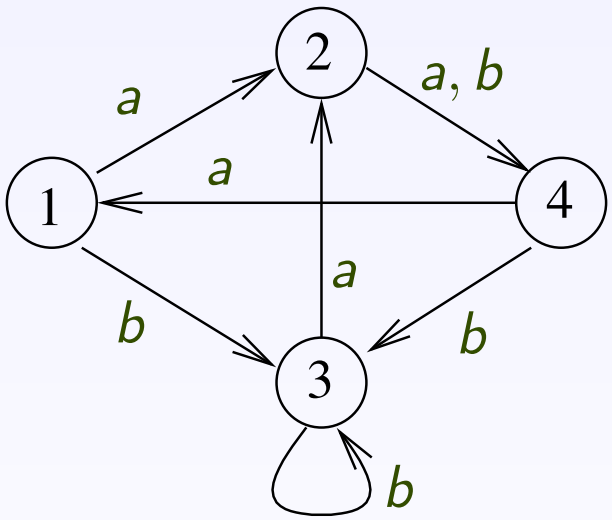
Théorème

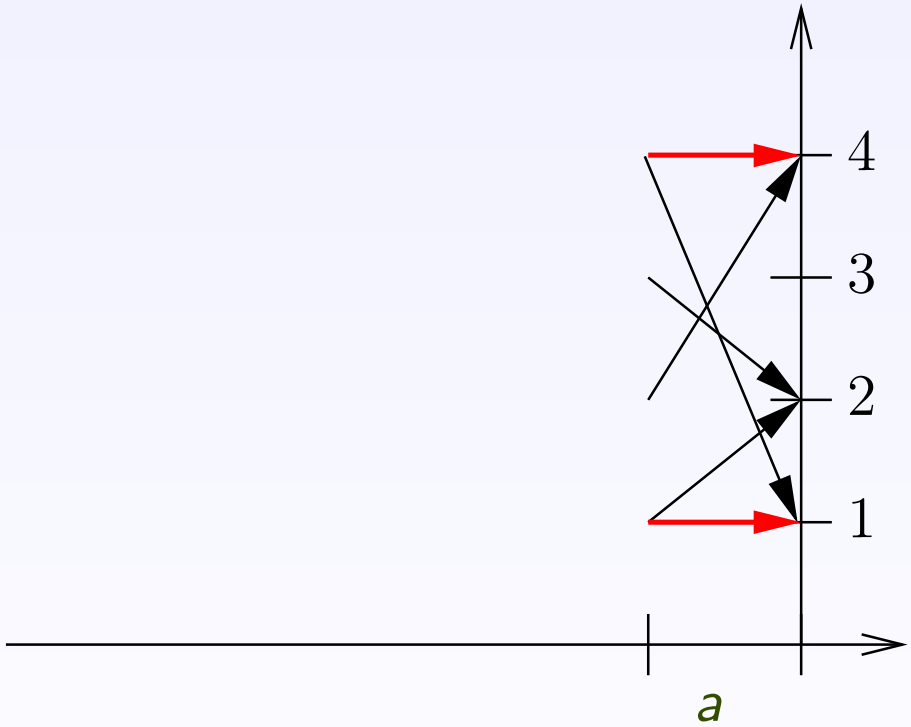
Si $[m_n, M_n]$ appartient à $\mathcal{S} = \{[x, x] \ : \ x \in \mathcal{X}\}$ pour un n fini :

$$\tau_e \quad \stackrel{\text{def}}{=} \quad \min \left\{ n \ : \ [\bot, \top] \boxdot u_{-n+1\to 0} \in \mathcal{S} \right\},$$

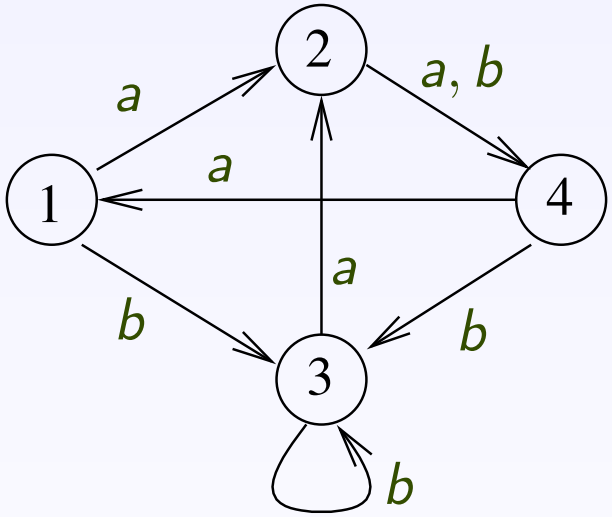
alors τ_e est le temps de couplage depuis le passé de la chaîne enveloppe. L'état défini par $[\bot, \top] \boxdot u_{-\tau_e+1\to 0}$ est distribué selon π.

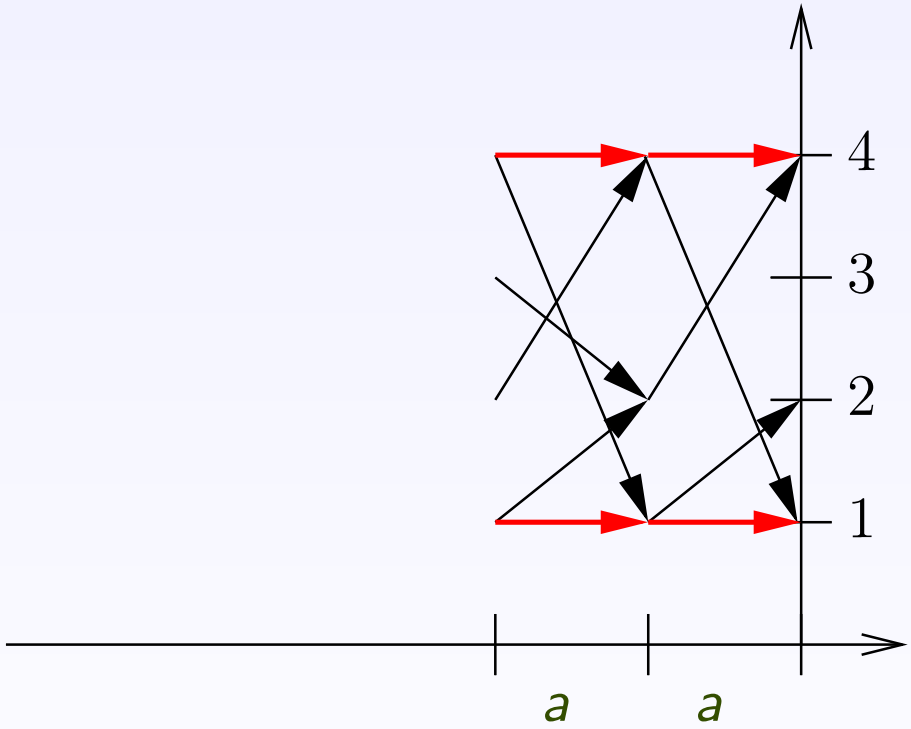
Exemple



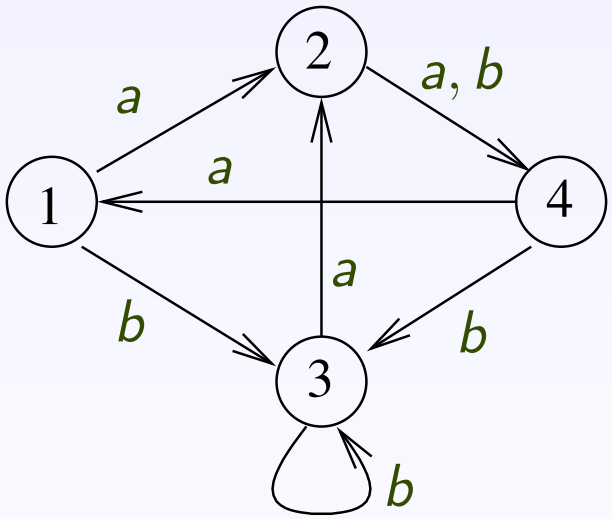


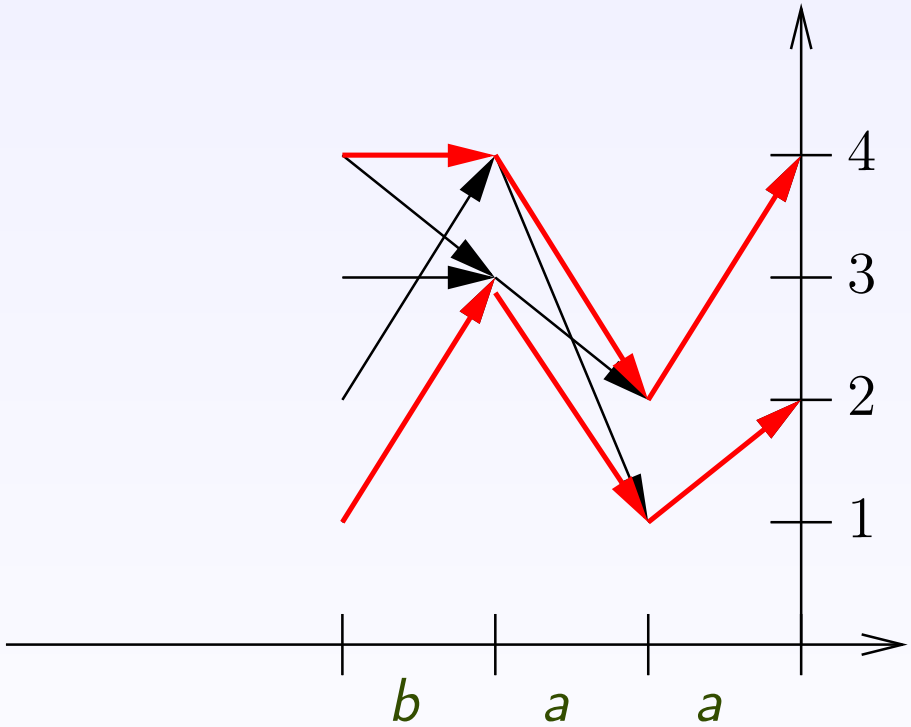
Exemple



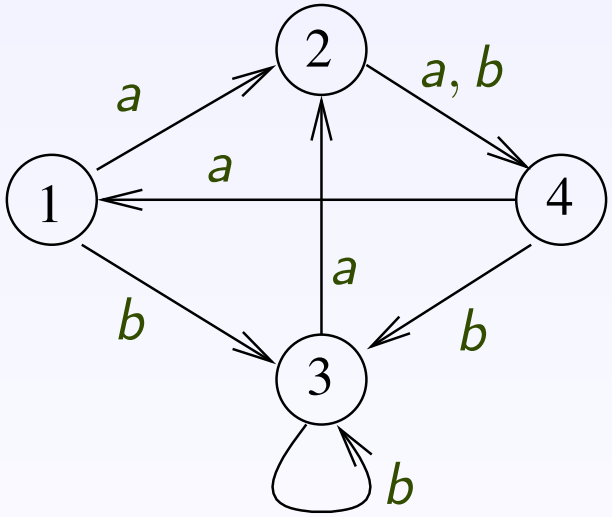


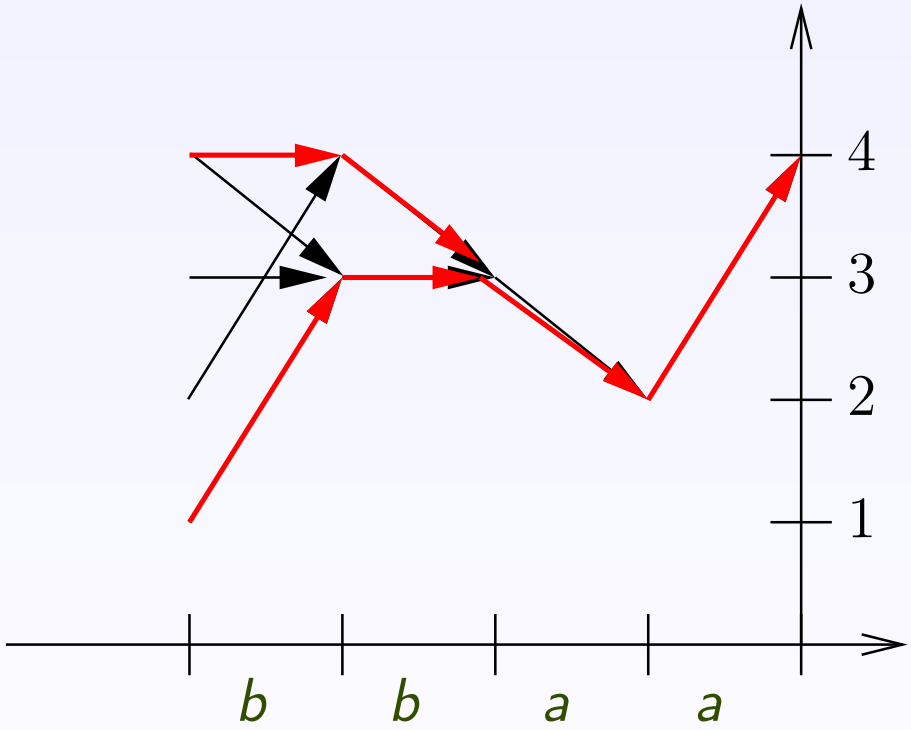
Exemple



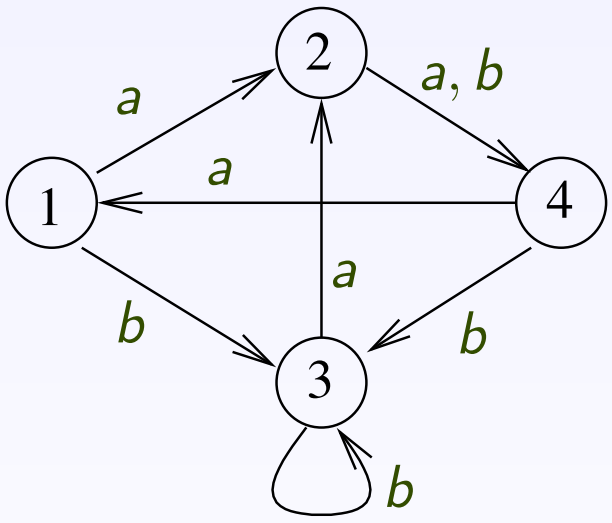


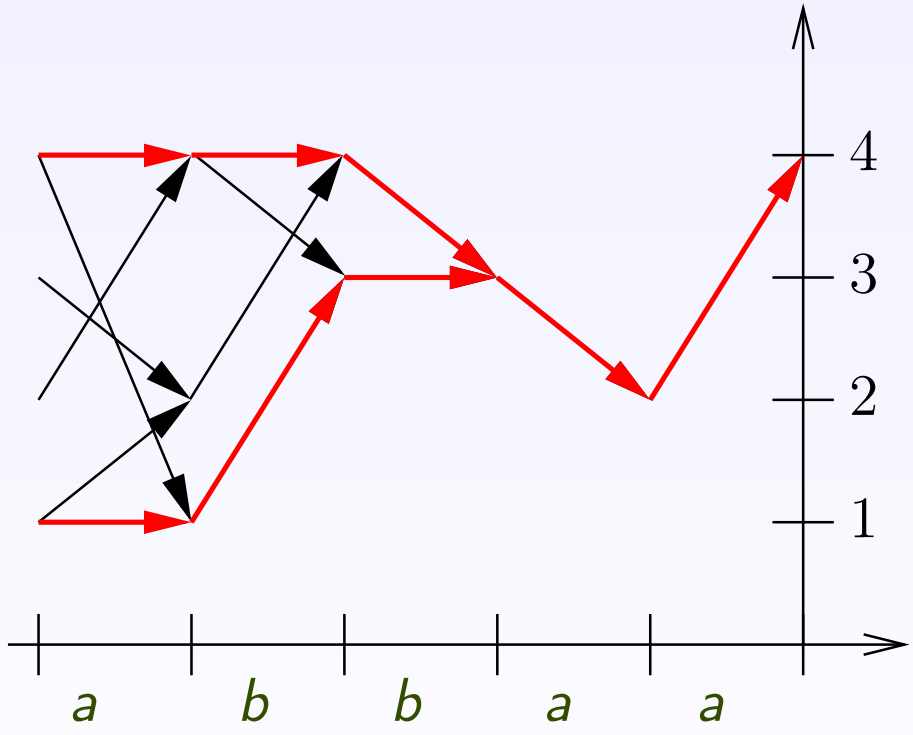
Exemple





Exemple





Envelope Perfect Sampling Algorithm (EPSA)

Data: suite i.i.d. $(u_{-n})_{n\in\mathbb{N}} \in A^{\mathbb{N}}$; l'opération enveloppe $\boxdot$
Result: Un état de $\mathcal{X}$ distribué selon π.
begin
  $n = 1$; $m := \bot$; $M := \top$;
  repeat
    for $i = n - 1$ **downto** 0 **do**
      $[m, M] := [m, M] \boxdot u_{-i}$;
    $n := 2n$;
  until $m = M$;
  $x^* := m$;
  return x^*;
end

Complexité : $O(\mathcal{C}_e \times \tau_e)$ (à comparer avec $O(\mathcal{C} \times |\mathcal{X}| \times \tau^b)$).

Commentaires

1. L'opération $\boxdot$ peut être remplacée par une sur-approximation $\odot$ telle que pour tout intervalle $[m, M]$, $[m, M] \boxdot a \subseteq [m, M] \odot a$, sans modifier la validité de l'algorithme.

2. La définition de l'opération enveloppe dépend de la description SED de la chaîne de Markov.

3. si un événement $a \in A$ est monotone, alors pour tout $m \preceq M$, $[m, M] \boxdot a = [m \cdot a, M \cdot a]$.

   Si tous les événements sont monotones (ou anti-monotones), alors EPSA coïncide avec la simulation parfaite classique - cas monotone.

Problèmes

- Il est possible que les enveloppes ne couplent pas même si les vraies trajectoires couplent.
- Quand les enveloppes couplent, leur temps de couplage peut être plus long que celui des trajectoires.
- La complexité du calcul des enveloppes peut être trop grande. Complexité de EPSA : $O(\mathcal{C}_e \times \tau_e)$.
  $\mathcal{C}_e$ ne peut pas être dans $\Omega(N)$ - sinon on ne gagne rien !

Simulation de files d'attentes non monotones

Dans le cas des files d'attentes, les action sont "homogènes en espace" par partie (i.e. $x \cdot a = x + v_R$ pour x dans la région R) et on a souvent $\mathcal{C} \sim \mathcal{C}_e$. [B., Gaujal, Vincent, 2008].

La différence entre la méthode classique et celle des enveloppes se joue donc sur la comparaison entre $|\mathcal{X}| \times \tau$ et τ_e.

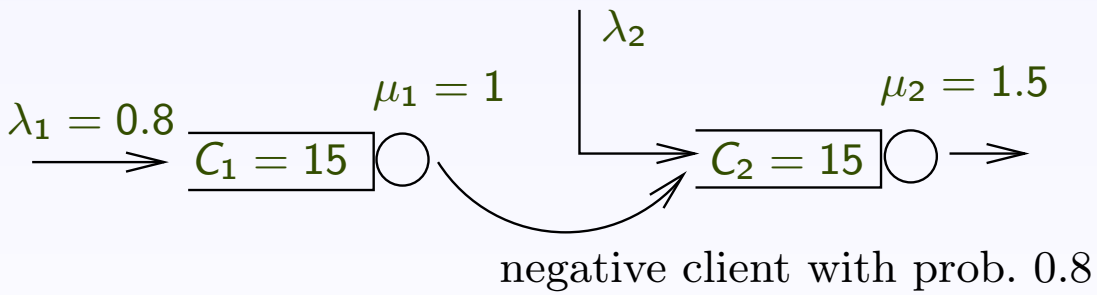


Fig.: Un réseaux de files avec clients négatifs.

Exemple

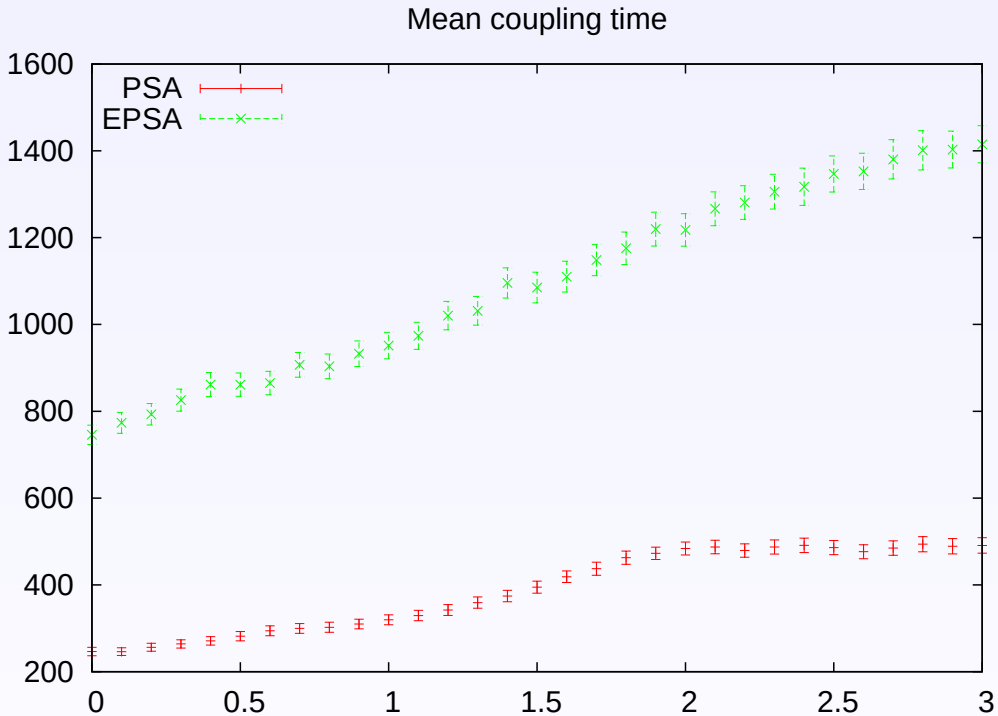


Fig.: Temps moyen de couplage de PSA et de EPSA en fonction de λ_2.

Files d'attente avec arrivée par lot (batch)

Les clients arrivent et partent par lot. un nouveau lot est accepté si il y a suffisamment de place dans la file. Sinon l'intégralité du lot est rejeté.

- Les événements 'lot' sont non-monotones.
  (Exemple : l'arrivée par lot de taille 2 dans une file de capacité C : $C-2 < C-1$ mais $C > C-1$.)
- Proposition : EPSA couple si et seulement si les lots de taille 1 peuvent survenir avec une probabilité positive dans chaque file.
- Les enveloppes peuvent être calculées en temps constant :

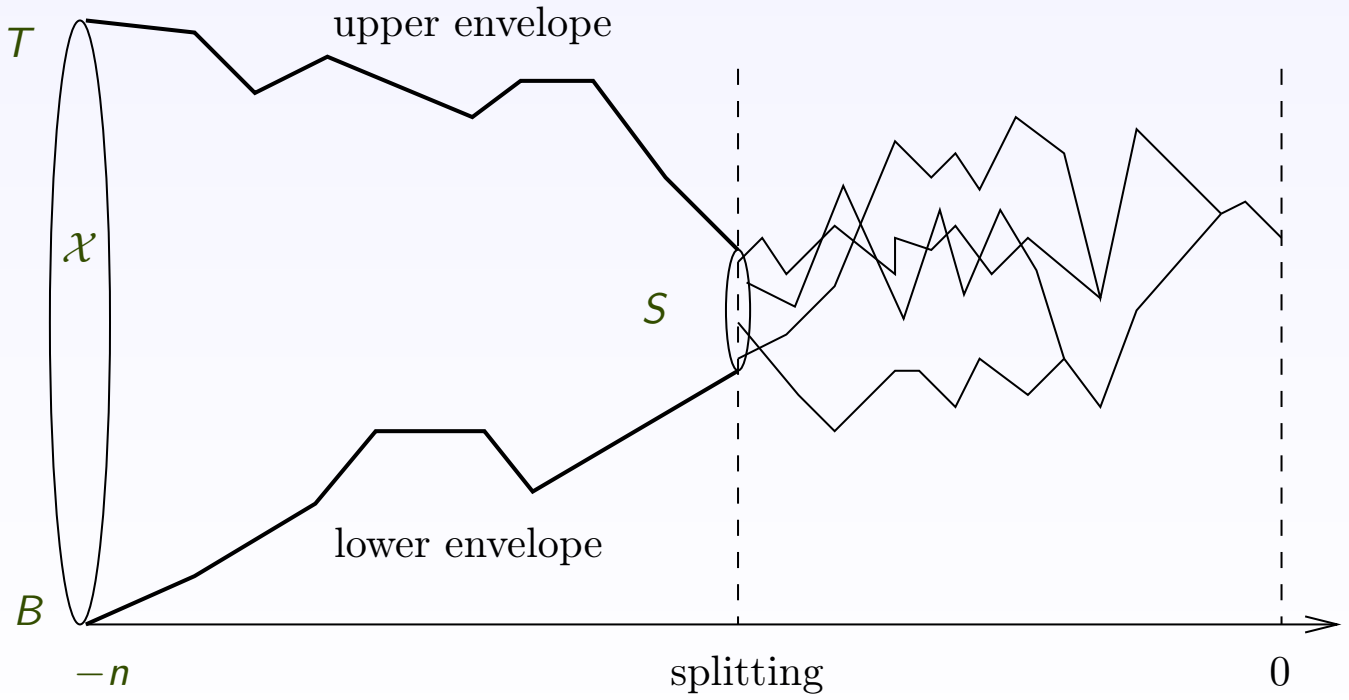
$$m' = \begin{cases} m+k, & M \leq C-k \\ (m+k) \wedge ((C+k-1) \vee m), & M > C-k. \end{cases}$$

$$M' = \begin{cases} (M+k) \wedge C, & m \leq C-k \\ M, & m > C-k \end{cases}$$

Au délà des enveloppes

Dans le cas où les enveloppes couplent beaucoup plus lentement que les trajectoires elles-mêmes (ou si elles ne couplent pas) :

- bornes
- splitting - on peut combiner les avantages des deux méthodes en faisant un éclatement.



Exemple

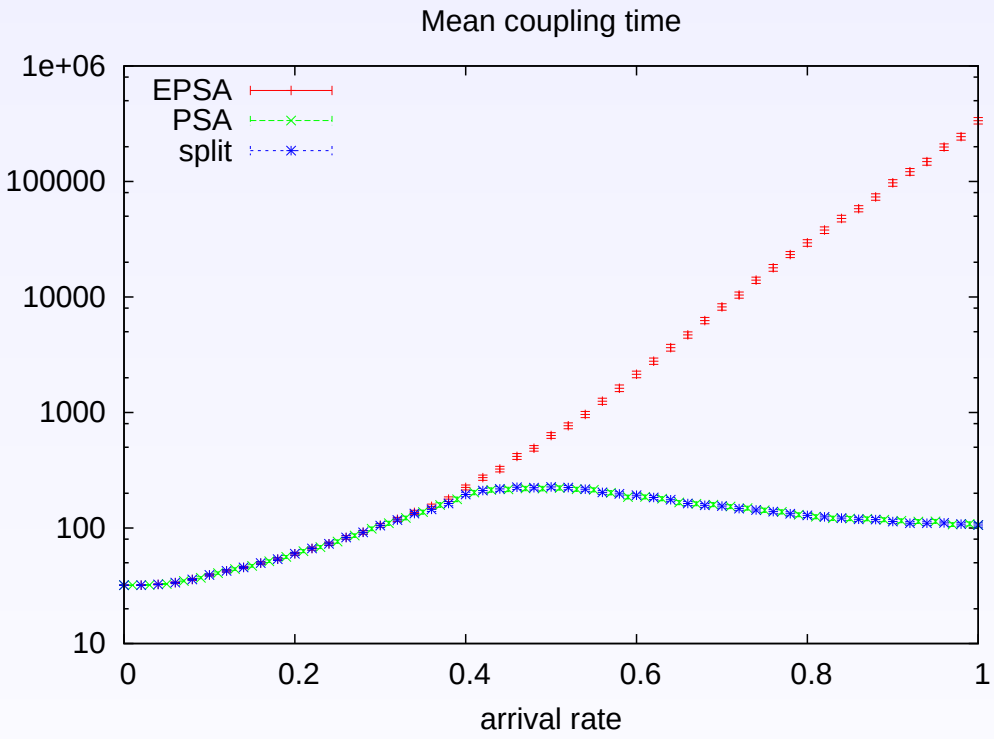
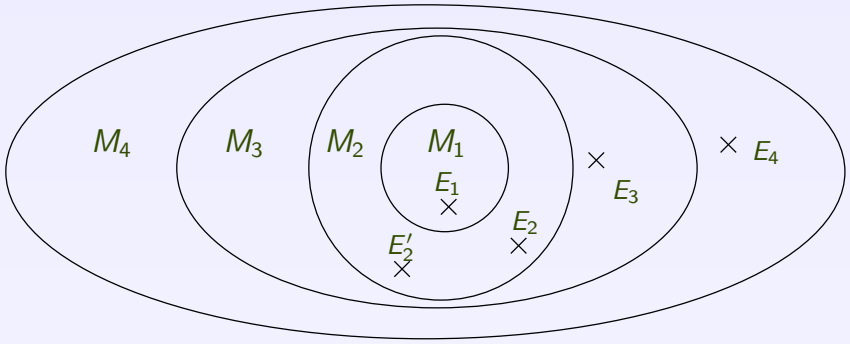


Fig.: Temps moyen de couplage de PSA, EPSA et EPSA avec éclatement pour une file $(+2, +3, -1)$.

Classes



Classes :

- M_1 - monotone MC.
- M_2 - non-monotone MC, où la simulation parfaite des enveloppes peut être utilisée efficacement.
- M_3 - enveloppes qui couplent, mais en prenant plus de temps.
- M_4 - enveloppes qui ne couplent pas (bornes, splitting).

Exemples :

- E_1 - un réseau de files d'attentes finies avec un routage monotone.
- E_2 - un réseau comme E_1 avec des clients négatifs.
  E_2' - un réseau comme E_1 avec des noeuds fork/join
- E_3 - un réseau avec des clients arrivant individuellement et par lot
- E_4 - un réseau de files d'attentes avec des arrivées par lot de taille supérieure à 2.