



Exploration aléatoire de modèles finis et non probabilistes

Johan Oudinet
<oudinet@lri.fr>

Laboratoire de Recherche en Informatique (LRI)
<http://www.lri.fr>
Université de Paris-Sud & CNRS

17 novembre 2009



UNIVERSITÉ
PARIS-SUD 11





Plan

- 1 Couverture par exploration aléatoire
- 2 Exploration aléatoire uniforme
- 3 Résultats expérimentaux
- 4 Conclusion

1 Couverture par exploration aléatoire

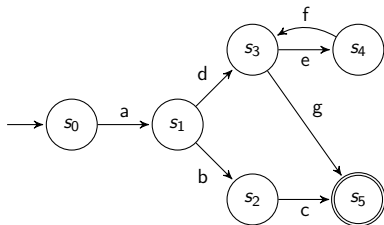
2 Exploration aléatoire uniforme

3 Résultats expérimentaux

4 Conclusion



Couverture de modèles



On peut vouloir couvrir :

- les 6 états
- les 7 transitions
- les 6 chemins de longueur ≤ 9 qui terminent en s_5

Problème

Si grands modèles et longs chemins, alors il y a un nombre exponentiel de chemins (d'une longueur bornée) à couvrir.

⇒ **couverture aléatoire des chemins**

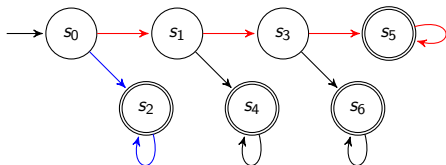


Les marches aléatoires (isotropes)

- Une marche aléatoire isotrope dans l'espace d'état du modèle (un graphe de contrôle, etc.) est :
une séquence d'états s_0, s_1, \dots, s_n telle que s_i est choisi uniformément parmi les successeurs de l'état s_{i-1}
- Très simple à implémenter et ne demande qu'une connaissance locale du graphe.
- Beaucoup d'applications en Test (protocoles), Simulation et Model-checking (travaux récents).



Inconvénient des marches aléatoires isotropes



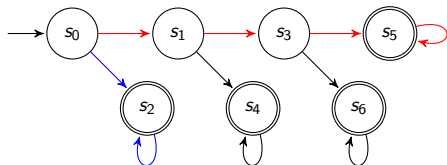
$$Pr(\{s_0, s_2, s_2, s_2\}) = 0.5$$

$$Pr(\{s_0, s_1, s_3, s_5\}) = 0.125$$

- La couverture obtenue est dépendante de la topologie du graphe



Inconvénient des marches aléatoires isotropes



$$Pr(\{s_0, s_2, s_2, s_2\}) = 0.5$$

$$Pr(\{s_0, s_1, s_3, s_5\}) = 0.125$$

- La couverture obtenue est dépendante de la topologie du graphe
- Comment équilibrer la probabilité de chaque chemin pour maximiser leur couverture?
⇒ tirage uniforme de chemins

1 Couverture par exploration aléatoire

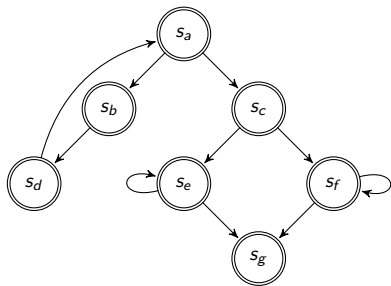
2 Exploration aléatoire uniforme

3 Résultats expérimentaux

4 Conclusion



Tirage uniforme de chemins de longueur n

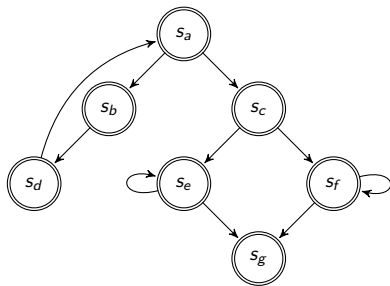


$$\begin{cases} l_s(0) = 1 \\ l_s(i) = \sum_{s \rightarrow s'} l_{s'}(i-1) \quad \forall i > 0 \end{cases}$$

$$Pr(s' | s, i) = \frac{l_{s'}(i-1)}{l_s(i)}$$



Tirage uniforme de chemins de longueur n



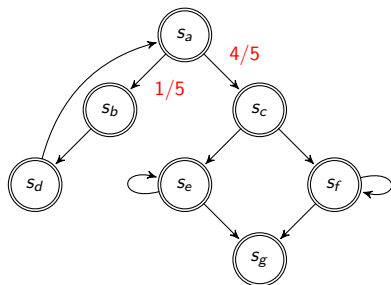
$$\begin{cases} l_s(0) = 1 \\ l_s(i) = \sum_{s \rightarrow s'} l_{s'}(i-1) \quad \forall i > 0 \end{cases}$$

$$Pr(s' | s, i) = \frac{l_{s'}(i-1)}{l_s(i)}$$

#	s_a	s_b	s_c	s_d	s_e	s_f	s_g
0	1	1	1	1	1	1	1
1	2	1	2	1	2	2	0
2	3	1	4	2	2	2	0
3	5	2	4	3	2	2	0



Tirage uniforme de chemins de longueur n



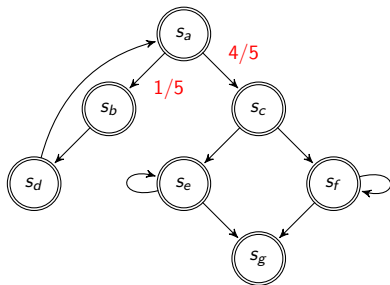
$$\begin{cases} l_s(0) = 1 \\ l_s(i) = \sum_{s \rightarrow s'} l_{s'}(i-1) \quad \forall i > 0 \end{cases}$$

$$Pr(s' | s, i) = \frac{l_{s'}(i-1)}{l_s(i)}$$

#	s_a	s_b	s_c	s_d	s_e	s_f	s_g
0	1	1	1	1	1	1	1
1	2	1	2	1	2	2	0
2	3	1	4	2	2	2	0
3	5	2	4	3	2	2	0



Tirage uniforme de chemins de longueur n



$$\begin{cases} l_s(0) = 1 \\ l_s(i) = \sum_{s \rightarrow s'} l_{s'}(i-1) \quad \forall i > 0 \end{cases}$$

$$Pr(s' | s, i) = \frac{l_{s'}(i-1)}{l_s(i)}$$

#	s_a	s_b	s_c	s_d	s_e	s_f	s_g
0	1	1	1	1	1	1	1
1	2	1	2	1	2	2	0
2	3	1	4	2	2	2	0
3	5	2	4	3	2	2	0

- Construire la table demande $\Theta(n \times d \times S)$ opérations.
- Ensuite, tirer un chemin demande $\Theta(d \times n)$ opérations arithmétiques.



Limites du tirage uniforme

Nombre de chemins d'une longueur n

Il est généralement exponentiel en n . Chaque nombre de la table de comptage occupe donc $\mathcal{O}(n)$ bits. Et chaque opération arithmétique est en $\mathcal{O}(n)$.

$\mathcal{O}(n^2 \times S)$ bits nécessaires pour stocker la table de comptage :

- si S est très grand, tirage modulaire [Gaudel, Denise, Gouraud, Lassaigne, Oudinet et Peyronnet, 2008]
- si n est grand, **variantes pour économiser de la mémoire.**
 - sans garder la table de comptage en mémoire
 - calculs avec une arithmétique flottante



Variante sans table de comptage

- La table de comptage est construite de la ligne 0 à la ligne n avec la récurrence suivante :

$$\begin{cases} l_s(0) = 1 \\ l_s(i) = \sum_{s \rightarrow s'} l_{s'}(i-1) \quad \forall i > 0 \end{cases}$$

- Le tirage d'un chemin parcourt cette table des lignes n à 0
- Si on avait la récurrence suivante :

$$l_s(i) = \sum_{t \in \mathcal{S}} F(s, t, l_t(i+1))$$

on pourrait éviter de conserver toute la table en mémoire en recalculant les lignes au fur et à mesure.

- Une génération plus lente mais plus économe en mémoire.



Solution des récurrences inverses

Solution en algèbre linéaire

Soit $A \in \mathbb{N}^{S \times S}$ la matrice d'adjacence, on peut toujours trouver un nombre n_0 et une matrice $B \in \mathbb{Q}^{S \times S}$ tels que :

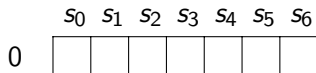
$$\forall i \geq n_0, \quad B.A^{i+1} = A^i$$

Finalement,

$$l_s(i) = \sum_{t \in S} r_{st} \times l_t(i+1) \quad \forall i > n_0$$



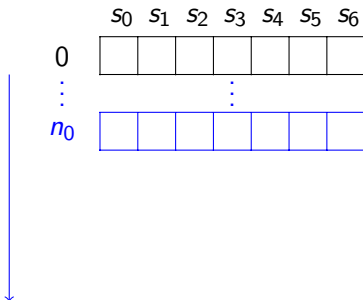
Algorithme de tirage uniforme sans table





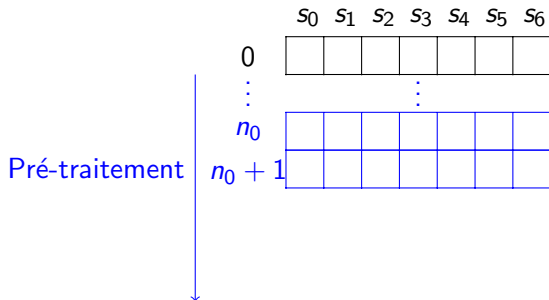
Algorithme de tirage uniforme sans table

Pré-traitement



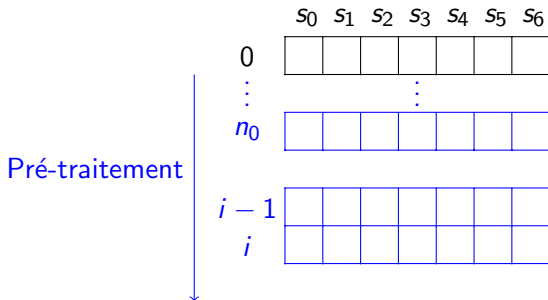


Algorithme de tirage uniforme sans table



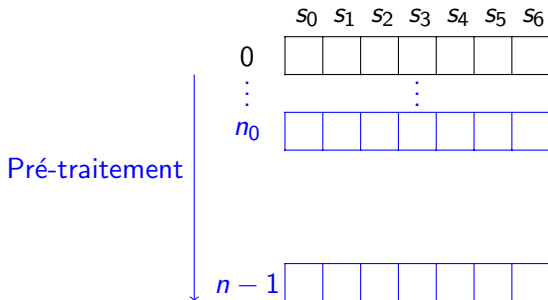


Algorithme de tirage uniforme sans table



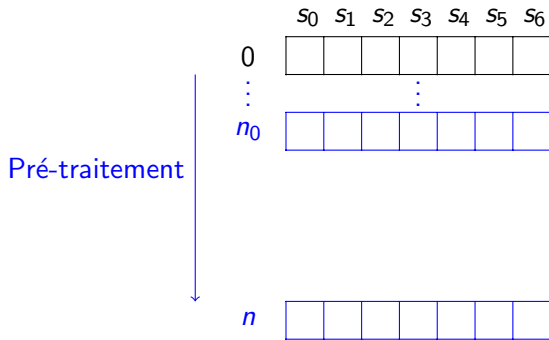


Algorithme de tirage uniforme sans table



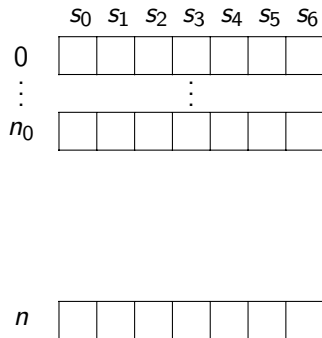


Algorithme de tirage uniforme sans table



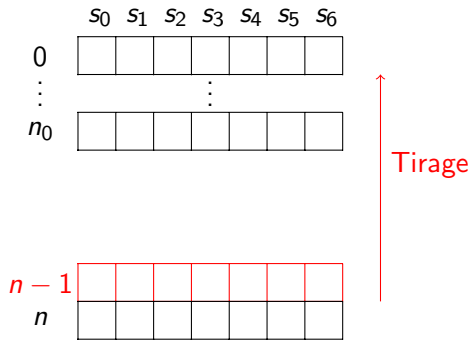


Algorithme de tirage uniforme sans table



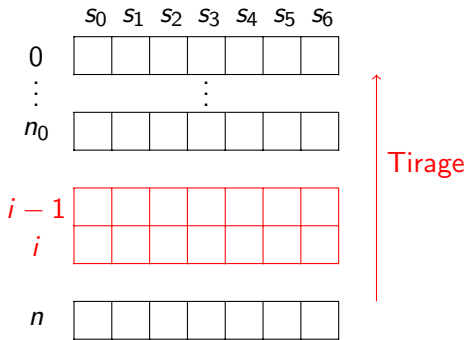


Algorithme de tirage uniforme sans table



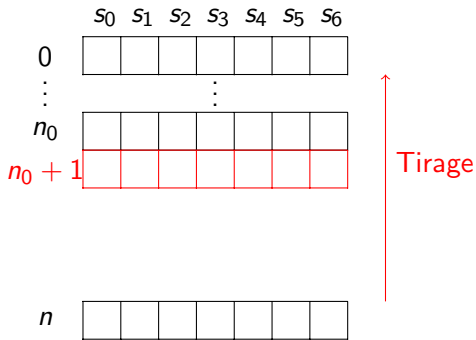


Algorithme de tirage uniforme sans table



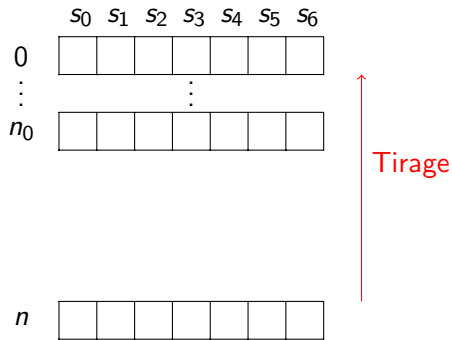


Algorithme de tirage uniforme sans table





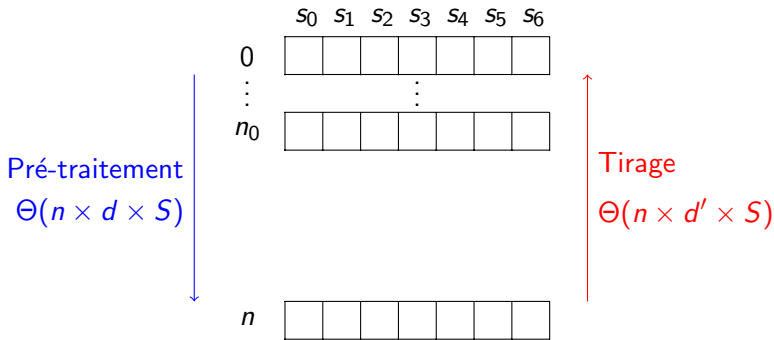
Algorithme de tirage uniforme sans table





Algorithme de tirage uniforme sans table

Mémoire : $\Theta((n_0 + 4) \times S)$ nombres





Variante avec une arithmétique flottante

- Pour gagner de la place et du temps : utiliser une arithmétique flottante à la place d'une arithmétique exacte.
- Taille de la mantisse fixe (p) et l'exposant en $\mathcal{O}(\log n)$.
- Denise et Zimmermann, 99 : borne sur l'erreur en $\mathcal{O}(n \times d \times 2^{-p})$ pour la variante avec table \Rightarrow numériquement stable.
- Essais avec la variante sans table ☹ (nombres négatifs et de norme supérieure à 1)



Complexités binaires

Soit :

- n la longueur des chemins ;
- S le nombre d'états du système.

Les complexités binaires pour générer un chemin de longueur n sont :

	Calcul exact		Calcul flottant	
	Temps	Espace	Temps	Espace
avec table	$\mathcal{O}(n^2 \times d)$	$\mathcal{O}(n^2 \times S)$	$\Theta(n \times d)$	$\Theta(n \times S)$
sans table	$\mathcal{O}(n^2 \times d' \times S)$	$\mathcal{O}(n \times S)$	$\Theta(n \times d' \times S)$	$\Theta(S)$

1 Couverture par exploration aléatoire

2 Exploration aléatoire uniforme

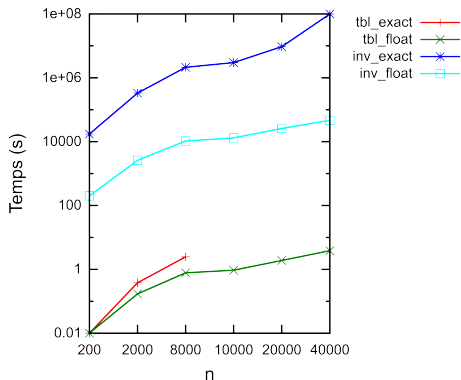
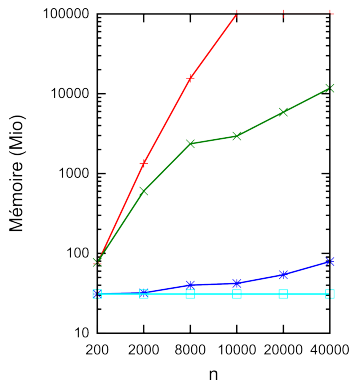
3 Résultats expérimentaux

4 Conclusion



Consommation mémoire et temps d'exécution pour générer 100 chemins

- modèle vasy_5_9 de la base VLTS : 5486 états et 9676 transitions.
- légende : avec ou sans table (tbl/inv) _ calcul exact ou flottant (exact/float)





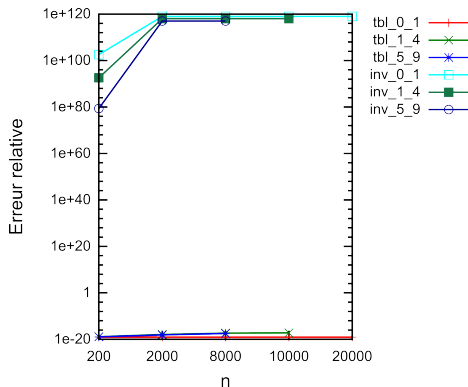
Mesure de l'erreur commise par le calcul flottant

- Erreur relative :

$$Err = \max_{\substack{s \in S \\ 0 < i \leq n}} \frac{|I_s(i) - \tilde{I}_s(i)|}{I_s(i)}$$

- avec table :
 $10^{-21} < Err < 10^{-17}$
- sans table :
 instabilité numérique

Mesure sur 3 modèles des variantes avec table (tbl) et sans table (inv) :



1 Couverture par exploration aléatoire

2 Exploration aléatoire uniforme

3 Résultats expérimentaux

4 Conclusion



Conclusion

Implémentation

Exploration aléatoire garantissant une bonne couverture des chemins du modèle, contrairement à une marche aléatoire isotrope.

Implémentation libre (licence LGPL et dépôt APP) : rukia.lri.fr

Résultats :

- En calcul exact, la variante sans table est intéressante car plus économe en mémoire.
- Longueur maximale sur `vasy_5_9` : 8000 \rightarrow 40000.
- En calcul flottant, la variante avec table permet de tirer plus rapidement des chemins plus longs (40000 avec 10Gio de RAM) et avec une très bonne approximation de l'uniformité.
- En calcul flottant, la variante sans table est sujette à l'instabilité numérique et de ce fait non utilisable.



Perspectives

- Extensions à la couverture des états et des transitions [Gaudel *et al.*, 2008 ; Gouraud *et al.*, 2001].
- Application à des fins de simulation, de test, ou de model-checking.