Chapitre 10 : Algorithmes probabilistes

INF4705 - Analyse et conception d'algorithmes

Gilles Pesant   Simon Brockbank

École Polytechnique Montréal
gilles.pesant@polymtl.ca, simon.brockbank@polymtl.ca

Hiver 2017

Plan

Introduction

Motivation

Les algorithmes probabilistes mettent le hasard à profit en faisant des choix (pseudo-)aléatoires, ce qui a généralement des répercussions sur le temps de calcul.

Conséquences

- Deux exécutions d'un algorithme probabiliste sur le même exemplaire peuvent prendre un temps différent et même donner des résultats différents.
- On permettra entre autres, avec une faible probabilité
  - de retourner une mauvaise réponse
  - de ne jamais terminer.

Définition d'un algorithme

Algorithme déterministe

Un algorithme est une suite finie d'instructions précises. Un algorithme qui résout un certain problème donne un résultat correct pour chaque exemplaire.

Définition d'un algorithme

Algorithme probabiliste

Un algorithme est une suite d'instructions . Un algorithme qui résout un certain problème donne souvent un résultat pour chaque exemplaire.

Définition d'un algorithme

Algorithme probabiliste

Un algorithme est une suite d'instructions . Un algorithme qui résout un certain problème donne souvent un résultat pour chaque exemplaire.

- On peut répéter la résolution d'un exemplaire jusqu'à ce que l'exécution termine ou qu'on ait suffisamment confiance que notre solution est bonne.

Types d'algorithmes probabilistes

algorithme	**termine ?**	**correct ?**
numérique probabiliste	$\surd$	
Monte Carlo	$\surd$	
Las Vegas		$\surd$
(Sherwood)	$\surd$	$\surd$

Exemple

Q - En quelle année Montréal a-t-elle accueilli les Olympiques ?

Réponse numérique probabiliste :

entre 1971 et 1976, entre 1973 et 1978, entre 1975 et 1980, entre 1970 et 1975, entre 1972 et 1977 . . .
19 fois sur 20.

Réponse Monte Carlo :

1976, 1976, 1976, 1976, 1977, 1976, 1976, 1976, 1976, 1642, 1976.

Réponse Las Vegas :

1976, 1976, 1976, Désolé !, 1976, (. . .), 1976, `bus error`, 1976, 1976.

Plan

Algorithmes numériques probabilistes

Donnent une solution approximative (parfois sous la forme d'un intervalle de confiance) dont la précision augmente avec le temps de calcul alloué.

Généralement plus rapides qu'un algorithme exact mais parfois aussi une meilleure alternative si :

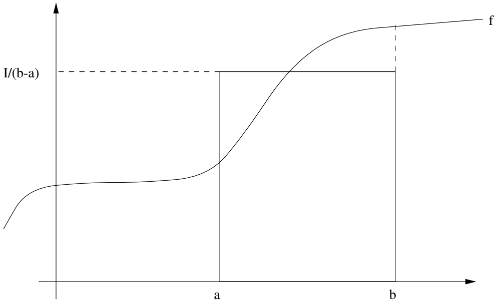
- les données expérimentales sont imprécises
- la représentation interne des nombres est approximative (p.ex. irrationnels)

Exemple : intégration numérique

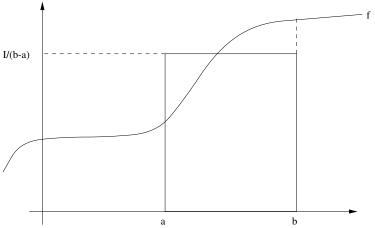
Le plus connu des algorithmes numériques probabilistes.

Soit $f : \mathbb{R} \to \mathbb{R}^{\geq 0}$ une fonction continue difficile à intégrer de manière symbolique et $a, b \in \mathbb{R}$ tels que $a \leq b$.
L'aire sous la courbe $y = f(x)$ entre $x = a$ et $x = b$ est donnée par

$$I = \int_a^b f(x)dx$$



Exemple : intégration numérique



Puisqu'un rectangle de même aire et même base a pour hauteur $I/(b-a)$, la "hauteur" moyenne de la courbe entre a et b sera également $I/(b-a)$.

L'algorithme numérique probabiliste estime cette hauteur moyenne par échantillonnage aléatoire de f dans $[a,b]$.

Exemple : intégration numérique

Algorithme d'intégration numérique probabiliste

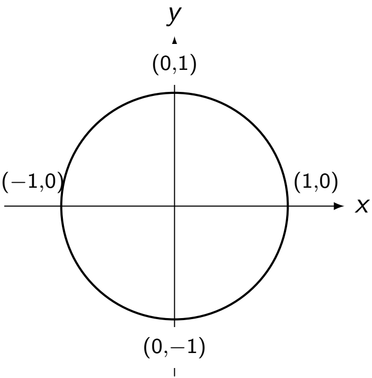
inp(f, a, b, n)
  $s \leftarrow 0$;
  pour $i \leftarrow 1$ **à** n **faire**
    $x \leftarrow$ choisi dans $[a, b]$ uniformément au hasard ;
    $s \leftarrow s + f(x)$;
  retourner $(s/n) \times (b - a)$;

L'erreur sur la valeur estimée par cet algorithme est inversement proportionnelle à $\sqrt{n}$ (la preuve de cela dépasse le cadre du cours).

Meilleure alternative qu'une approche déterministe lorsque le domaine de f est de dimension élevée (≥ 4) .

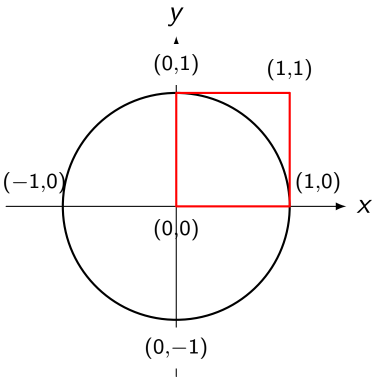
Exemple : Approximation du nombre π

Considérons le cercle de rayon unitaire centré à l'origine.



Exemple : Approximation du nombre π

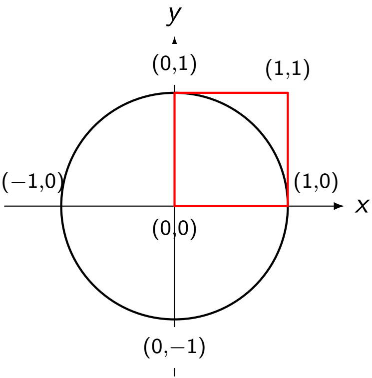
Considérons le cercle de rayon unitaire centré à l'origine.
Choisissons uniformément au hasard un point (x, y) dans $[0, 1] \times [0, 1]$.



Exemple : Approximation du nombre π

Considérons le cercle de rayon unitaire centré à l'origine.
Choisissons uniformément au hasard un point (x, y) dans $[0, 1] \times [0, 1]$.

Si $x^2 + y^2 \leq 1$ alors (x, y) est dans le cercle. Quelle est la probabilité d'un tel événement ?

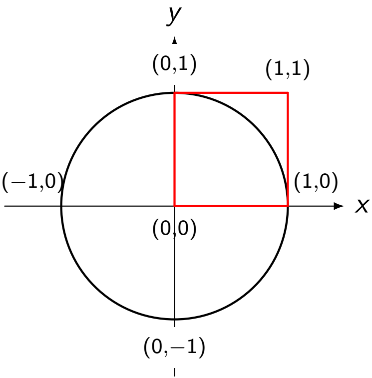


Exemple : Approximation du nombre π

Considérons le cercle de rayon unitaire centré à l'origine.
Choisissons uniformément au hasard un point (x, y) dans $[0, 1] \times [0, 1]$.

Si $x^2 + y^2 \leq 1$ alors (x, y) est dans le cercle. Quelle est la probabilité d'un tel événement ?

$$\frac{\pi 1^2/4}{1^2} = \frac{\pi}{4}$$



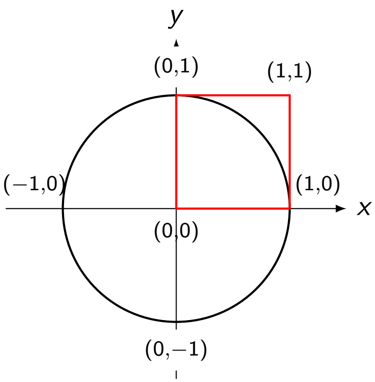
Exemple : Approximation du nombre π

Considérons le cercle de rayon unitaire centré à l'origine.
Choisissons uniformément au hasard un point (x, y) dans $[0, 1] \times [0, 1]$.

Si $x^2 + y^2 \leq 1$ alors (x, y) est dans le cercle. Quelle est la probabilité d'un tel événement ?

$$\frac{\pi 1^2/4}{1^2} = \frac{\pi}{4}$$

Donc $\pi \approx \frac{4 \times \text{nb de succès}}{\text{nb d'essais}}$



Plan

Algorithmes Sherwood

Donnent toujours une (bonne) réponse mais le temps de calcul varie indépendamment de l'exemplaire, à cause des choix aléatoires qui sont faits.

Servent à égaliser le temps de calcul espéré pour chacun des exemplaires, lorsqu'il y a une grosse différence entre le comportement moyen et en pire cas d'un algorithme déterministe.

Algorithmes Sherwood

Effet Robin des Bois

On redistribue l'inefficacité de quelques-uns à tout le monde.

Le comportement en pire cas existe toujours mais il n'est plus systématiquement associé à quelques exemplaires particuliers.

On rend ainsi un tel algorithme moins vulnérable aux mauvaises distributions d'exemplaires.

Algorithmes Sherwood

Ex. “quicksort”

$\mathcal{O}(n \lg n)$ en moyenne mais $\Omega(n^2)$ en pire cas (lorsque déjà trié si on choisit le premier élément comme pivot)

Si on a surtout des exemplaires presque triés, ça coûtera cher... que faire ?

Algorithme Sherwood : choisissons le pivot uniformément au hasard $\Rightarrow \mathcal{O}(n \lg n)$ espéré peu importe l'exemplaire

Algorithmes Las Vegas

La réponse donnée est toujours correcte mais, avec une probabilité faible, ils ne donnent pas de réponse du tout.

Souvent plus rapides que l'alternative déterministe,
même s'il faut les relancer jusqu'à ce qu'on obtienne une réponse.

Relance aléatoire

Si la probabilité de succès (terminaison) est p, le nombre espéré de relances (répétitions) de l'algorithme Las Vegas afin d'obtenir une réponse est $1/p$:

épreuve de Bernouilli ; var aléatoire $X =$ nb répétitions pour succès
$P(X = k) = (1-p)^{k-1}p$ (distribution géométrique)
$E[X] = \sum_{k=1}^{\infty} k(1-p)^{k-1}p = 1/p$

Algorithmes Las Vegas

Temps d'exécution espéré

Soient T_s et T_e les temps d'exécution espérés en cas de succès et d'échec, respectivement, pour un seul appel.

Le temps d'exécution espéré pour l'ensemble des répétitions sera

$$T_{tot} = (\frac{1}{p} - 1)T_e + T_s$$

Si $T_s = T_e$, on a simplement

$$T_{tot} = T_s/p$$

Ex. n-reines

Mesurons l'effort de calcul par le nombre de noeuds explorés dans l'arbre de recherche.

8-reines

approche	p	NbNœuds_s	NbNœuds_e	NbNœuds_{tot}
déterministe ("backtrack")	1.0000	114.00	–	114.00
Las Vegas pur	0.1293	9.00	6.97	55.93
Las Vegas hybride 2-6	0.8750	22.53	39.67	28.20

39-reines

approche	p	NbNœuds_{tot}	T_{tot}
déterministe	1.00	10^{10}	41 hrs
Las Vegas hybride 29-10	$\sim$0.21	500	8.5 ms

Plan

Algorithmes Monte Carlo

- mauvaise réponse à l'occasion
- probabilité élevée de retourner la bonne réponse indépendamment de l'exemplaire considéré

Algorithmes Monte Carlo

p-correct

Un algorithme Monte Carlo est p-correct si la probabilité qu'il retourne la bonne réponse est au moins p, peu importe l'exemplaire :

$$\forall \text{ exemplaire} \quad P(\text{ succès} \mid \text{exemplaire }) \geq p$$

Ex. Tirer à pile ou face la réponse à un problème de décision :

$$1/2\text{-correct}$$

Algorithmes Monte Carlo

biais

Un algorithme Monte Carlo dont au moins une des réponses possibles est nécessairement correcte lorsque obtenue est dit biaisé.

Si au contraire la probabilité d'erreur est non nulle pour toutes les réponses possibles, on le dit non biaisé :

$$\forall \text{ réponse} \qquad P(\text{ succès} \mid \text{réponse }) < 1$$

Ex. Tirer à pile ou face la réponse à un problème de décision :

non biaisé

Algorithmes Monte Carlo

Amplification de l'avantage stochastique

On augmente la probabilité de retourner la réponse correcte en répétant des appels indépendants à un algorithme Monte Carlo.

Algorithmes Monte Carlo

Soit `Abiaisé` un algorithme biaisé pour la réponse “oui”.

`Abiaisé-amplifié`(e, k)

pour $j \leftarrow 1$ **à** k **faire**
  si `Abiaisé`(e)=“oui” **alors retourner** “oui” ;
retourner “non” ;

Amplification pour un algorithme biaisé

k répétitions d'un algorithme p-correct biaisé nous permettent d'obtenir un algorithme q-correct où

$$q = 1 - (1-p)^k$$

Pour garantir une probabilité d'erreur d'au plus ϵ,
on devra faire $k = \lceil \log_{1/(1-p)} 1/\epsilon \rceil$ répétitions.

Exemple

Même quand p est faible !

$$p = 1/10; \quad k = 10 \text{ répétitions}$$

$$q = 1 - (1 - 1/10)^{10} \approx 0.65$$

Algorithmes Monte Carlo

Amplification pour un algorithme non biaisé

On peut aussi l'appliquer aux algorithmes non biaisés si $p > 1/2$.
À défaut de certitude, on retournera la réponse majoritaire.

`Anonbiaisé-amplifié`$(e, k$ {impair}$)$

  $c \leftarrow 0$;
  pour $j \leftarrow 1$ **à** k **faire**
    si Anonbiaisé(e)="oui" **alors** c++ ;
    sinon c - - ;
  si $c > 0$ **alors retourner** "oui" ;
  sinon retourner "non" ;

Exemple

Soit un algorithme 3/4-correct non biaisé ; $k = 3$
Énumérons les 8 cas possibles avec leur probabilité
(B = bonne réponse ; M = mauvaise réponse) :

1er	2e	3e	proba.	réponse	
B	B	B	27/64	B	$\surd$
B	B	M	9/64	B	$\surd$
B	M	B	9/64	B	$\surd$
B	M	M	3/64	M	
M	B	B	9/64	B	$\surd$
M	B	M	3/64	M	
M	M	B	3/64	M	
M	M	M	1/64	M	

$$27/64 + 9/64 + 9/64 + 9/64 = 27/32 > 84\% > 75\%$$

Algorithmes Monte Carlo

Amplification pour un algorithme non biaisé

k répétitions d'un algorithme p-correct, $p > 1/2$, non biaisé nous permettent d'obtenir un algorithme q-correct où

$$q = \Sigma_{i=0}^{\lfloor k/2 \rfloor} \binom{k}{i} p^{k-i}(1-p)^i$$

Algorithmes Monte Carlo

Rendement de l'amplification

Pour amplifier un algorithme Monte Carlo 0.55-correct jusqu'à 0.95-correct :

- s'il est non biaisé, il nous faut 269 répétitions
- s'il est biaisé, il nous faut 4 répétitions

Exemple : Vérification de produit matriciel

Soient trois matrices A, B et C dans $\mathbb{Z}^{n \times n}$.

On souhaite vérifier que $AB = C$.

algorithme déterministe

$D \leftarrow AB$;
pour $i \leftarrow 1$ **à** n **faire**
  pour $j \leftarrow 1$ **à** n **faire**
    si $D_{ij} \neq C_{ij}$ **alors retourner** "non" ;
retourner "oui" ;

Exemple : Vérification de produit matriciel

Soient les matrices $X \in \{0,1\}^{1\times n}$, $\quad C^X, D^X \in \mathbb{Z}^{1\times n}$.

$X(AB)$ calcule la somme point à point de certaines lignes (indiquées par X) du produit AB.
Mais on peut aussi la calculer ainsi : $(XA)B$

algorithme probabiliste Monte Carlo

`Freivalds`(A, B, C, n)
  pour $j \leftarrow 1$ **à** n **faire**
    $X_j \leftarrow U[0..1]$;
  $D^X \leftarrow (XA)B$;
  $C^X \leftarrow XC$;
  pour $j \leftarrow 1$ **à** n **faire**
    si $D^X_j \neq C^X_j$ **alors retourner** "non" ;
  retourner "oui" ;

Exemple : Vérification de produit matriciel

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \; B = \begin{pmatrix} 3 & 1 & 4 \\ 1 & 5 & 9 \\ 2 & 6 & 5 \end{pmatrix} \; C = \begin{pmatrix} 11 & 29 & 37 \\ 29 & 65 & 91 \\ 47 & 99 & 45 \end{pmatrix}$$

$$X = (\quad , \quad , \quad)$$

$$\begin{aligned} XA &= (\quad , \quad , \quad) \\ (XA)B &= (\quad , \quad , \quad) \\ XC &= (\quad , \quad , \quad) \end{aligned}$$

Exemple : Vérification de produit matriciel

biaisé ?

p-correct ?

Exemple : Vérification de produit matriciel

amplification ?

`Freivalds-amplifié`(A, B, C, n, k)

pour $i \leftarrow 1$ **à** k **faire**

  si `Freivalds`$(A, B, C, n) =$ "non" **alors retourner** "non" ;

retourner "oui" ;