Parallélisation d'une combinaison des méthodes de Monte Carlo et quasi-Monte Carlo et application aux réseaux de files d'attente

B. Tuffin et L.M. Le Ny

IRISA

Campus universitaire de Beaulieu

35042 Rennes Cedex, France

Résumé

Nous proposons un algorithme parallèle qui regroupe les caractéristiques des deux méthodes de simulation Monte Carlo et quasi-Monte Carlo. Une analyse détaillée de cet algorithme, complétée par quelques exemples, montre que l'efficacité de l'estimateur est une fonction linéaire du nombre de processeurs. Comme exemple d'utilisation concrète, nous avons évalué des mesures de performance d'un réseau de files d'attente multi-classes à forme produit en régime stationnaire.

Mots-Clés : Monte Carlo, Quasi-Monte Carlo, Simulation parallèle, Réseaux de files d'attente, Constante de normalisation.

Abstract

We propose a parallel algorithm which uses both Monte Carlo and quasi-Monte Carlo methods. A detailed analysis of this algorithm, followed by examples, shows that the estimator's efficiency is a linear function of the processor number. As a concrete application example, we evaluate performance measures of a multi-class queueing network in steady state.

Keywords Monte Carlo, Quasi-Monte Carlo, Parallel simulation, Queueing networks, Normalization constant.

1 Introduction

Nous nous intéressons à la parallélisation d'un algorithme qui exploite les avantages des deux méthodes de simulation Monte Carlo et quasi-Monte Carlo.

Les simulations de type Monte Carlo permettent d'estimer statistiquement un paramètre en utilisant une suite aléatoire de nombres [7, 15]; l'erreur commise peut être évaluée à l'aide du théorème de la limite centrale. Par exemple, pour le calcul approché d'une intégrale multiple en dimension s, la vitesse de convergence est indépendante de s et est en moyenne en $O(N^{-1/2})$ où N est la taille de l'échantillon utilisé.

Cet ordre de grandeur étant seulement une moyenne, il existe nécessairement des suites pour lesquelles la convergence est plus rapide. L'usage de ces suites appelées *suites à discrépance faible* intervient dans les méthodes dites de quasi-Monte Carlo [5, 11]. Malheureusement les bornes connues de l'erreur sont dans ce cas inexploitables en pratique. Nous sommes donc conduits à combiner les deux techniques pour tirer profit de leurs avantages respectifs (borne de l'erreur pour Monte Carlo, même si elle n'est que probabiliste, et convergence rapide pour quasi-Monte Carlo), en utilisant les suites à discrépance faible pour réduire la variance dans les méthodes de Monte Carlo. C'est la parallélisation de cette technique que nous développons ici, et nous constaterons qu'elle permet d'obtenir des résultats encore plus performants.

Les différentes étapes de ce travail sont les suivantes : nous effectuons d'abord quelques rappels sur les méthodes de Monte Carlo et de quasi-Monte Carlo, en particulier sur la notion de discrépance. Après avoir donné un exemple simple de suite à discrépance faible, nous présentons la méthode combinée que nous utilisons. Nous rappelons brièvement ses propriétés, puis nous nous intéressons à sa parallélisation. Pour ce faire, nous avons été amené à implanter un générateur pseudo-aléatoire adéquat sur l'ordinateur utilisé (Paragon d'Intel). Nous pouvons ainsi proposer une méthode pour laquelle le gain obtenu est linéaire. Enfin, nous donnons un exemple d'application à un domaine très important pour l'évaluation des performances des systèmes de télécommunication et de production :

les réseaux de files d'attente à forme produit. Les probabilités stationnaires sont alors connues à une constante multiplicative près, et cette constante de normalisation, incalculable pour les gros réseaux, peut être représentée par une intégrale que nous estimerons par la méthode décrite précédemment.

2 Combinaison de méthodes de Monte Carlo et quasi-Monte Carlo

2.1 Monte Carlo et quasi-Monte Carlo

Soit $(X_i)_{1 \leq i \leq I}$ une suite finie de I vecteurs aléatoires indépendants uniformément distribués sur $[0,1)^s$. On sait, par la loi des grands nombres, qu'un estimateur sans biais de

$$\mathcal{I} = \int_{[0,1]^s} f(x)dx$$

est

$$\bar{f}_I = \frac{1}{I} \sum_{i=1}^{I} f(X_i).$$

La variance de l'estimateur $\bar{f}_I$ est alors σ^2/I ; σ^2 étant la variance de $f(X)$ avec X uniformément distribué sur $[0,1)^s$. Le théorème de la limite centrale nous donne une idée de la distribution de cet estimateur en fonction de I . On sait que

$$\sqrt{I}\frac{\bar{f}_I - \mathcal{I}}{\sigma}$$

a approximativement pour distribution la loi normale centrée réduite. Ceci nous permet donc d'obtenir un intervalle de confiance pour $\mathcal{I}$:

$$\mathcal{I} \in \left[\frac{1}{I} \sum_{i=1}^{I} f(X_i) - \frac{c_\alpha \sigma}{\sqrt{I}}, \frac{1}{I} \sum_{i=1}^{I} f(X_i) + \frac{c_\alpha \sigma}{\sqrt{I}} \right]$$

au risque α, où $c_\alpha = \Phi^{-1}(1 - \frac{\alpha}{2})$ et Φ est la fonction de répartition de la loi normale centrée réduite. La vitesse de convergence d'une telle méthode est donc, en moyenne, en $O(I^{-1/2})$, indépendamment de la dimension s du problème. L'efficacité de l'estimateur $\bar{f}_I$ est alors $I/(t\sigma^2)$ où t est le temps de calcul pour obtenir $\bar{f}_I$.

Dans les méthodes de quasi-Monte Carlo, les suites considérées sont déterministes et se répartissent très bien sur tout le domaine d'intégration. On souhaite que la suite soit uniformément répartie, c'est à dire que, si B un sous-intervalle de $[0,1)^s$ et $A_N(B,\mathcal{P})$ le nombre d'éléments de la suite $\mathcal{P} = (\xi_n)_{n\in\mathbb{N}}$ parmi les N premiers appartenant à B, i.e.

$$A_N(B,\mathcal{P}) = \sum_{n=1}^{N} 1_B(\xi_n),$$

on ait alors

$$\lim_{N\to\infty} A_N(B,\mathcal{P}) = \int_{[0,1]^s} 1_B(x)dx.$$

Pour mesurer la qualité de cette répartition, définissons la *discrépance* des N premiers termes de $\mathcal{P}$ par

$$D_N^*(\mathcal{P}) = \sup_{x\in[0,1]^s} \left| \frac{A_N([0,x),\mathcal{P})}{N} - \prod_{i=1}^{s} x_i \right|.$$

avec $x = (x_1, \cdots, x_s)$. La suite $\mathcal{P} = (\xi_n)_{n \in \mathbb{N}}$ est alors uniformément distribuée si et seulement si $\lim_{N \to +\infty} D_N^*(\mathcal{P}) = 0$ [11]. Les bornes de l'erreur dans les méthodes de quasi-Monte Carlo pour l'approximation de $\mathcal{I}$ peuvent être obtenues en fonction de la discrépance. Soit P une partition de $[0,1]^s$ en sous-intervalles et $\Delta(f, J)$ la somme alternée des valeurs de f aux sommets du sous-intervalle J [5, 11]. On définit la variation au sens de Vitali par

$$V_{Vit}(f) = \sup_P \sum_{J \in P} |\Delta(f, J)| .$$

À partir de $V_{Vit}(f)$ on définit $V(f)$, la variation de f au sens de Hardy et Krause par

$$V(f) = \sum_{k=1}^{s} \sum_{1 \le i_1 < \cdots < i_k \le s} V_{Vit}^{(k)}(f; i_1, \cdots, i_k)$$

où $V_{Vit}^{(k)}(f; i_1, \cdots, i_k)$ est la variation au sens de Vitali de la restriction de f à l'espace de dimension k $\{(u_1, \cdots, u_s) \in [0,1]^s : u_j = 1 \text{ pour } j \neq i_1, \cdots, i_k\}$. On a la borne suivante de l'erreur [5, 11] :

$$\left| \frac{1}{N} \sum_{n=1}^{N} f(\xi_n) - \int_{[0,1)^s} f(u) du \right| \le V(f) D_N^*(\mathcal{P}).$$

Une suite $\mathcal{P} = (\xi_n)_{n \in \mathbb{N}}$ est dite à *discrépance faible* si on a $D_N^*(\mathcal{P}) = O(N^{-1} (\ln N)^s)$. Il a été prouvé par Roth [2] que, pour une suite infinie, on ne peut avoir une meilleure convergence que $O(N^{-1} (\ln N)^{\alpha(s)})$ où $\alpha(s)$ est telle que $\alpha(1) = 1$ et $(s-1)/2 \le \alpha(s) \le s$.

Il existe de nombreuses suites à discrépance faible. Parmi celles-ci on notera les suites de Halton, SQRT, de Sobol', de Niederreiter, de Faure [19].

A titre d'exemple, décrivons brièvement les suites SQRT et de Sobol' qui sont les plus performantes pour la méthode combinée décrite plus loin.

Définissons d'abord la suite SQRT, dont la traduction algorithmique est immédiate. Soit $z_1, \cdots, z_s$ s nombres irrationnels linéairement indépendants sur $\mathbb{Q}$. Alors $\mathcal{P} = (\xi_n)_{n \in \mathbb{N}}$, la suite sur $[0,1)^s$ définie par

$$\xi_n = \{n\boldsymbol{z}\} = (\{nz_1\}, \cdots, \{nz_s\}),$$

où $\{nz\} = nz \bmod 1$, est à discrépance faible. La suite SQRT est la suite $(\{n\boldsymbol{\alpha}\})_{n \in \mathbb{N}}$ où $\boldsymbol{\alpha} = (\sqrt{p_1}, \cdots, \sqrt{p_s})$, p_i étant le i^{eme} plus petit nombre premier.

Les suites de Sobol' [12] utilisent la nature binaire des ordinateurs pour diminuer le temps de calcul (l'addition devenant le ou exclusif). Alors, $\forall n \geq 1$, $\xi^{(n)} = \frac{a_1}{2} V^{(1)} + \cdots + \frac{a_l}{2^l} V^{(l)} \bmod 1$ où les $V^{(i)} \in [0,1)^s$ sont des vecteurs de direction préalablement calculés [12] et les $a_i \in \{0,1\}$ pour $1 \leq i \leq l$ sont les $l = l(n)$ bits du code de Gray de n. Notons que dans le travail initial de Sobol', les a_i sont les digits du développement de n en base 2, mais le code de Gray permet une plus grande rapidité de génération. Notons de plus que l'algorithme de génération que nous utiliserons est basé sur celui explicitement donné dans [12].

Les bornes de l'erreur étant très peu utilisables en pratique [19], car les majorations possibles de la variation et de la discrépance donnent peu d'information (pour N fixé, elles donnent une borne excessive de l'erreur réelle), on utilise une méthode combinée pour pouvoir estimer l'erreur d'approximation tout en gardant la vitesse de convergence inhérente aux suites à discrépance faible.

2.2 La méthode combinée

Cette méthode utilise les suites à discrépance faible dans les méthodes de Monte Carlo dans un objectif de réduction de la variance. L'idée a été initiée dans [4] et dans [18] avant d'être approfondie dans [19] et [20].

Soit $(X_i)_{1\leq i\leq I}$ I variables aléatoires indépendantes de même loi uniforme sur $[0,1)^s$ et $\mathcal{P} = (\xi_n)_{n\in\mathbb{N}}$ une suite à discrépance faible sur $[0,1)^s$ telle que l'une de celles mentionnées dans le paragraphe précédent. Considérons les I variables aléatoires indépendantes

$$Y_i = \frac{1}{N}\sum_{n=1}^{N} f(\{X_i + \xi_n\}), \tag{1}$$

où $1 \leq i \leq I$ et $\{x\} = (\{x_1\}, \ldots, \{x_s\})$ est le vecteur des parties fractionnaires de chaque coordonnée du vecteur $x \in \mathbb{R}^s$.

Il est prouvé dans [19] que, sous certaines conditions, la variance de la variable aléatoire Y_i est en $O(N^{-2}(\ln N)^{2s})$. On obtient donc une réduction conséquente de la variance par comparaison avec celle de la somme de N variables aléatoires indépendantes et identiquement distribuées $f(X)$. Citons certains résultats obtenus dans [19, 20] :

Théorème 1 *Soit $\mathcal{P} = (\xi_n)_{n\in\mathbb{N}}$ une suite à discrépance faible sur $[0,1)^s$. Si f est une fonction à variation finie et X une v.a. uniformément répartie sur $[0,1)^s$, on a*

$$\sigma^2\left(\frac{1}{N}\sum_{n=1}^{N} f(\{X+\xi_n\})\right) = O(N^{-2}(\ln N)^{2s}). \tag{2}$$

De plus, on peut montrer que cette vitesse de convergence est aussi valable pour d'autres fonctions que celles à variation finie. En effet, considérons l'ensemble $\mathcal{F}$ des fonctions continues, muni de la mesure de Wiener qui est concentrée sur des fonctions à variation infinie [10]. Rappelons que la mesure de Wiener μ_W est une mesure gaussienne d'espérance nulle et de noyau de covariance

$$R(x,y) = \int_{\mathcal{F}} f(x)f(y)d\mu_W(f) = \prod_{i=1}^{s} \min(x_i, y_i).$$

On peut alors obtenir le théorème suivant :

Théorème 2 *Si $\mathcal{P} = (\xi_n)_{n\in\mathbb{N}}$ est suite à discrépance faible sur $[0,1)^s$, la variance moyenne $\sigma^2_{N,avg}$ de la variable aléatoire $\displaystyle\frac{1}{N}\sum_{n=1}^{N} f(\{\xi_n + X\})$, moyenne prise sur l'ensemble $\mathcal{F}$ des fonctions f continues sur $[0,1]^s$ muni de la mesure de Wiener μ_W, est en $O(N^{-2}(\ln N)^{2s})$.*

La variance de l'estimateur utilisé $\sum_{i=1}^{I} Y_i/I$ (pour NI appels de f) est donc dans ce cas en $O(I^{-1}N^{-2}(\ln N)^{2s})$. Il est donc intéressant de prendre N

le plus grand possible pour réduire au maximum la variance. Cependant, I doit être suffisamment grand pour que l'approximation de la loi normale dans le théorème de la limite centrale permette d'obtenir un intervalle de confiance intéressant.

3 Simulation parallèle

3.1 Description de la méthode choisie

Nous avons effectué les simulations sur une machine parallèle Intel de type Paragon, composée de 56 processeurs $i860$, dont 3 noeuds de service et 2 autres réservés aux entrées-sorties parallèles. Chaque noeud utilisé dans nos calculs dispose de 16 Méga-octets de mémoire.

Soit Y la suite finie de variables aléatoires comprenant I termes $(Y_i)_{1 \leq i \leq I}$ et P le nombre de processeurs utilisés. Choisissons I multiple de P et convenons d'appeler itération un calcul de Y_i comme défini dans le paragraphe précédent. Nous pouvons alors effectuer I/P itérations sur chaque processeur, puis collecter les résultats. La vecteur aléatoire X_i étant simulé à l'aide d'un générateur congruentiel non fourni sur la machine Paragon, nous avons dû en intégrer un à notre bibliothèque de suites à discrépance faible (pour une telle simulation parallèle, il n'a par contre pas été nécessaire de changer le code des générateurs de suites à discrépance faible initialement écrit pour un ordinateur séquentiel).

Le générateur pseudo-aléatoire que nous utilisons (il n'en existait pas sur l'Intel Paragon) est intéressant pour sa simplicité et nous a été inspiré par G. Rubino comme perfectionnement de celui proposé dans [6]. D'autres générateurs ayant de meilleures propriétés statistiques peuvent être déterminés [11, 7] mais celui que nous avons utilisé nous a donné entière satisfaction.

Dans des situations différentes où la dimension est infinie, cette technique de parallélisation n'est pas la mieux adaptée, comme par exemple pour l'étude des cycles régénératifs pour les chaines de Markov, où le temps de génération de $f(X)$ peut être très variable. Ainsi, effectuer I/P itérations sur chaque processeur n'est pas optimal, certains processeurs pouvant avoir fini bien avant les autres. Une étude statistique de ce cas est réalisée dans [8] et des méthodes sont proposées dans [6, 8]. Cependant, dans nos problèmes la dimension mathématique est fixe et le temps de génération de $f(X)$ reste toujours approximativement le même. Nous n'avons donc pas ce genre de complication.

3.2 Le générateur pseudo-aléatoire utilisé

Considérons un générateur pseudo-aléatoire congruentiel de période M

$$U_{n+1} = aU_n + b \bmod M,$$

où U_n est un entier compris entre 0 et $M-1$ et $X_n = U_n/M$ est le n^{eme} élément de la suite pseudo-aléatoire. La méthode que nous utilisons consiste à séparer cette suite en P sous-suites "indépendantes" $(X_{nP+k})_{n\in\mathbb{N}}$ $k = 0, \cdots, P-1$, une étant allouée à chaque processeur. La k^{eme} sous-suite pseudo-aléatoire $(X_n^{(k)})_{n\in\mathbb{N}}$ est alors donnée par

$$X_n^{(k)} = U_n^{(k)}/M$$

avec $U_0^{(k)} = U_k$ et $U_{n+1}^{(k)} = a^P U_n^{(k)} + b' \bmod M$, où $b' = b(1+a+a^2+\cdots+a^{P-1})$. Nous prendrons pour valeur $a = 1664525$, $b = 1013904223$ et $M = 2^{32}$ [9, 12].

3.3 L'algorithme parallèle

Nous donnons ci-dessous les principales étapes de l'algorithme choisi, Les notations utilisées sont celles définies dans 2.2 et 3.2.

1. `Définition de la fonction f.`
2. `Initialisation des calculs et du générateur pseudo-aléatoire sur chaque processeur`

   `Pour` k `de` 0 `à` $P-1$`,`

       `Affectation de` $U_0^{(k)}$ `au processeur` k`.`
3. `Pour` k `de` 0 `à` $P-1$`,`

       `pour` i `de` 1 `à` I/P`,`

```
            Création du vecteur (X_i^(k))
            Pour n de 1 à N,
                  Création du vecteur ξ_n
                  Calcul de f({X_i^(k) + ξ_n})
            fin pour n
            Calcul de Y_i^(k) = 1/N Σ_{n=1}^N f({X_i^(k) + ξ_n}) et de (Y_i^(k))^2
      fin pour i
   fin pour k
4. Regroupement des résultats obtenus sur chaque processeur
   Estimation finale: 1/(NI) Σ_{k=1}^P Σ_{i=1}^{I/P} Σ_{n=1}^N f({X_i^(k) + ξ_n})
   et intervalle de confiance à partir du moment d'ordre 2.
```

La parallélisation se déroule donc à l'étape 3 où l'on répartit les itérations à réaliser sur les différents processeurs (boucle indexée par k).

3.4 Gain

Dans cette partie, nous illustrons le gain apporté par la simulation parallèle. Rappelons que que ce gain n'est en général pas mieux que linéaire par rapport au nombre de processeurs utilisés.

La fonction test que nous utilisons (mais une autre donnerait des résultats en tout point similaires), en dimension $s = 5$, est

$$f(x_1, \cdots, x_5) = \prod_{i=1}^{5} \frac{\pi}{2} \sin(\pi x_i).$$

La simulation est réalisée pour $N = 1000$ éléments de la suite à discrépance faible SQRT et $I = 30000$ itérations indépendantes.

La figure 1 donne l'évolution du gain obtenu sur l'efficacité $I/(\sigma^2 t)$ de l'estimateur de Monte Carlo (voir section 2.1) en utilisant P processeurs, pour P variant de 1 à 20. Comme on peut le constater, la courbe obtenue est linéaire. Il est à noter que l'efficacité n'est ici pas à confondre avec celle correspondant au quotient gain sur P habituellement utilisée en parallélisme.

4 Application aux réseaux de files d'attente à forme produit

Les systèmes de communication ou informatiques peuvent être efficacement représentés par des modèles stochastiques. Parmi ceux-ci, les réseaux de Jackson fermés multi-classes à forme produit ont une importance considérable étant donné leurs propriétés. La forme générale des probabilités stationnaires pour de tels réseaux est connue, à une constante de normalisation près qui est en général inconnue. Ainsi un problème important dans ce domaine consiste à déterminer des algorithmes efficaces pour l'évaluation de ces modèles et, en particulier, le calcul de ces constantes de normalisation qui sont des sommes sur tous les états possibles du réseau. Les méthodes de calcul (MVA [14], convolution [3] ...) sont rapidement inefficaces quand le nombre d'états augmente. Dans ce cas, on utilise des méthodes d'approximation. Celles donnant de meilleurs résultats pour le moment sont celles de Monte Carlo [15, 16, 21, 17] et plus récemment celles utilisant la méthode combinée [20].

Les réseaux que nous étudions dans cet article sont constitués de M stations, J classes de clients et une population de N_j clients de classe j. Nous supposons qu'il existe deux types de discipline de service dans les stations : des stations te type premier arrivé premier servi (PAPS) à un serveur, numérotées de 1 à L, et des stations à une infinité de serveurs (IS), numérotées de $L+1$ à M. Toutes les classes ont la même distribution de service exponentielle pour chaque station PAPS de moyenne $1/\mu_m$ pour $m = 1, \cdots, L$. Par contre, les classes peuvent avoir des services exponentiels différents aux stations IS de moyenne $1/\mu_{jm}$ pour la classe j à la station m.

Les routages sont supposés indépendants de l'état, et sont donc donnés par une matrice de transition. Pour chaque classe j, soit λ_{jm} le taux de visite relatif d'un client de classe j à la station m (obtenu à partir de la matrice des routages). Soit $\rho_{jm} = \lambda_{jm}/\mu_m$ pour $m = 1, \cdots, L$ et $\rho_{jm} = \lambda_{jm}/\mu_{jm}$, pour $m = L+1, \cdots, M$, l'intensité du trafic pour la classe j à la station m. On note $\rho_{j0} = \sum_{m=L+1}^{M} \rho_{jm}$, $1 \leq j \leq J$ l'intensité totale du trafic de classe j pour l'ensemble des stations IS.

Un état du réseau est un vecteur de dimension JM

$$n = (n_{jm})_{1\leq j\leq J,\ 1\leq m\leq M}$$

où n_{jm} est le nombre de clients de classe j à la station m.

L'ensemble de tous les états possibles du réseau est alors [1]

$$\Omega = \left\{ \boldsymbol{n} \left| \sum_{m=1}^{M} n_{jm} = N_j \text{ pour } j = 1, \cdots, J \right. \right\}.$$

Si $n_m = \sum_{j=1}^{J} n_{jm}$, les probabilités stationnaires pour un tel réseau sont données par

$$\pi(\boldsymbol{n}) = \frac{1}{g}\delta(\boldsymbol{n}) \text{ avec } \delta(\boldsymbol{n}) = \prod_{m=1}^{M} n_m! \prod_{j=1}^{J} \frac{\rho_{jm}^{n_{jm}}}{n_{jm}!},$$

où

$$g = \sum_{\boldsymbol{n}\in\Omega} \delta(\boldsymbol{n}) \tag{3}$$

est la constante de normalisation.

Il est prouvé dans [13] que la constante de normalisation g peut s'écrire

$$g = \frac{1}{\prod_{j=1}^{J} N_j\ !} \int_{\boldsymbol{Q}^+} {}^{-\mathbf{1}'\boldsymbol{u}} \prod_{j=1}^{J} (\rho_{j0} + \boldsymbol{\rho}_j'\boldsymbol{u})^{N_j} d\boldsymbol{u}, \tag{4}$$

où

$$\boldsymbol{u} \quad = \quad (u_1, \cdots, u_L)'$$

$$\begin{aligned}
\mathbf{1} &= (1, \cdots, 1)' \\
\boldsymbol{\rho}_j &= (\rho_{j1}, \cdots, \rho_{jL})' \\
\boldsymbol{Q}^+ &= \{\boldsymbol{u} \in \mathbb{R}^L \ : \ u_l \geq 0, \ l = 1, \cdots, L\}.
\end{aligned}$$

Si on définit

$$\Omega_{(j)} = \left\{ \boldsymbol{n} \,\middle|\, \sum_{m=1}^{M} n_{km} = N'_k \text{ pour } k = 1, \cdots, J \right\}$$

avec $N'_k = N_k$ si $k \neq j$ et $N'_j = N_j - 1$, alors $g_j = \sum_{\boldsymbol{n} \in \Omega_{(j)}} \delta(\boldsymbol{n})$ est la constante de normalisation du même réseau avec un client de classe j en moins.

Étant données ces constantes de normalisation, des mesures de performance des réseaux peuvent être aisément déduites. Par exemple,

$$TH_{jm} = \lambda_{jm} \frac{g_j}{g} \tag{5}$$

représente le débit moyen des clients de classe j à la station m.

Dans cette application de la méthode combinée pour estimer les constantes de normalisation de gros réseaux, nous n'utilisons la loi uniforme sur $[0,1)^s$ pour l'intégration, mais une loi plus générale. On peut remarquer que, par une simple transformation, il est souvent possible de générer une suite ayant une distribution particulière sur un sous-espace de $\mathbb{R}^s$ à partir d'une suite uniformément répartie sur $[0,1)^s$, à l'aide de s fonctions pseudo-inverses unidimensionnelles. Par exemple, si $F^{-1}(u) = -\ln(1-u)/\gamma$ est la fonction inverse de la fonction de répartition d'une variable aléatoire exponentielle de paramètre γ (i.e. de loi $Exp(\gamma)$) et U suit une loi uniforme sur $[0,1)$, alors $F^{-1}(U)$ suit une loi $Exp(\gamma)$ [20].

Ainsi, soit $F_l^{-1}(u) = -\ln(1-u)/\gamma_l$ avec $\gamma_l > 0$. En utilisant pour estimer (4) une mesure d'échantillonnage préférentiel de loi exponentielle (voir [7] pour une description de l'échantillonnage préférentiel) $(l = 1, \cdots, L)$, un estimateur de g est [20]

$$\bar{Y}_I = \frac{1}{I} \sum_{i=1}^{I} Y_i$$

avec

$$Y_i = \frac{1}{N} \sum_{k=1}^{N} \frac{1}{\prod_{j=1}^{J} N_j!} \frac{e^{-\mathbf{1}'\boldsymbol{V}_{ik}} \prod_{j=1}^{J} (\rho_{j0} + \boldsymbol{\rho}_j' \boldsymbol{V}_{ik})^{N_j}}{p(\boldsymbol{V}_{ik})}$$

et

$$\boldsymbol{V}_{ik} = (F_1^{-1}(\{U_{1i} + \xi_{1k}\}), \cdots, F_L^{-1}(\{U_{Li} + \xi_{Lk}\}))$$

pour $(U_{li})_{1 \leq l \leq L, 1 \leq i \leq I}$ variables aléatoires indépendantes et de même loi uniforme sur $[0,1)$, $(\xi_k)_{k \in \mathbb{N}}$, avec $\xi_k = (\xi_{1k}, \cdots, \xi_{Lk})$, une suite à discrépance faible sur $[0,1)^L$ et en posant $p(v_1, \cdots, v_L) = \prod_{l=1}^{L} \gamma_l e^{-\gamma_l v_l}$ produit des densités de lois exponentielles $Exp(\gamma_l)$. L'algorithme de calcul de $\bar{Y}_I$ est alors celui décrit en section 3.3, mais avec $P = 1$ processeur.

Ross et Wang ont prouvé (voir [15]) que, asymptotiquement, la probabilité d'échantillonnage p est optimale pour réduire la variance pour des valeurs de γ_l, $1 \leq l \leq L$, dépendantes de certaines propriétés du réseau (voir [17]).

A titre d'exemple, et pour permettre en regroupant les classes de vérifier les résultats obtenus par l'algorithme MVA, considérons un système constitué de $M = 9$ stations dont $L = 7$ PAPS. Nous prenons $J = 15$ classes de clients telles que $N_1 = \cdots = N_J = 2$ et $\mu_{jm} = 0.01$ $(L + 1 \leq m \leq M)$, $s_m = 1$ et $\mu_m = 1$ $(1 \leq m \leq L)$. Les routages sont identiques pour toutes les classes : Les clients vont de la station m à la station $m + 1$ $(m = 1, \cdots, M - 1)$ et de la station M à la station 1 avec une probabilité 1.

Si on effectue une simulation comme décrite précédemment sur $P = 10$ processeurs pour $I = 100$ et $N = 10000$ points de la suite de Sobol', on obtient les résultats exposés dans le tableau 1. La même simulation effectuée sur un unique processeur donne des résultats équivalents en un temps 10 fois plus important, conformément à ce que nous pouvions attendre. La version séquentielle de cette méthode étant déjà nettement la plus performante connue [20], l'algorithme parallèle que nous avons développé ici est donc le plus performant existant pour le moment.

5 Conclusion

Nous avons présenté et analysé un algorithme parallèle qui combine les deux méthodes de simulation Monte Carlo et quasi-Monte Carlo. Le principal intérêt de cet algorithme réside dans le fait que sa vitesse est une fonction

Variable	estimation	Intervalle de confiance
g	9.940751e+64	(9.940711e+64, 9.940791e+64)
g_j	9.552608e+62	(9.552560e+62, 9.552655e+62)
TH_{jm}	9.60954270e-03	(9.60952765e-03, 9.60955795e-03)

TAB. 1 – *Résultats obtenus pour une simulation utilisant 10 processeurs. Temps de calcul : 53 secondes.*

linéaire du nombre de processeurs, tout en conservant les atouts des méthodes : borne et rapidité. Deux exemples confirmant ce fait sont proposés : le calcul d'une intégrale multiple et des constantes de normalisation d'un réseau de files d'attente multi-classes.

Références

[1] F. Baskett, M. Chandy, R. Muntz, and J. Palacios. Open, closed and mixed networks of queues with different classes of customers. *Journal of the Association for Computing Machinery*, 22:248–260, 1975.

[2] J. Beck and W. Chen. *Irregularities of Distribution.* Cambridge University Press, 1987.

[3] J.P. Buzen. Computational algorithms for closed queueing networks with exponential servers. *Communications of ACM*, 16:527–531, 1973.

[4] R. Cranley and T.N.L. Patterson. Randomization of number theoretic methods for multiple integration. *SIAM J. Numer. Anal.*, 13(6):904–914, December 1976.

[5] M. Drmota and R.F. Tichy. *Sequences, Discrepancies and Applications*, volume 1651 of *Lecture Notes in Mathematics*. Springer Verlag, Heidelberg, 1997.

[6] M. El Khadiri, R. Marie, and G. Rubino. Parallel estimation of 2-terminal network reliability by a crude Monte Carlo technique. In M. Baray and B. Özgüç, editors, *Computer and Information Sciences VI*. Elsevier Science, 1991.

[7] G.S. Fishman. *Monte Carlo: Concepts, Algorithms and Applications*. Springer-Verlag, 1997.

[8] P. Heidelberger. Discrete Event Simulations and Parallel Processing: Statistical Properties. *SIAM J. Stat. Comput.*, 9(6):1114–1132, November 1988.

[9] D. E. Knuth. *The art of computer progamming*, volume 2. Addison-Wesley, 1981.

[10] W. J. Morokoff and R. E. Caflisch. Quasi-Random Sequences and their Discrepancies. *SIAM Journal on Scientific Computing*, pages 1571–1599, December 1994.

[11] H. Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods.* CBMS-SIAM 63, Philadelphia, 1992.

[12] W.H. Press, S.A. Teutolsky, W.T. Vetterling, and B.P. Flannery. *Numerical recipes in C: the art of scientific computing.* Cambridge; New York; Oakeigh: Cambridge University Press, 1992.

[13] K.G. Ramakrishnan and D. Mitra. An Overview of PANACEA, a Software Package for Analyzing Markovian Queueing Networks. *Bell System Technical Journal*, 61:2849–2872, 1982.

[14] M. Reiser and H. Kobayashi. Queueing networks with multiple closed chains: Theory and computational algorithms. *IBM J. of Research and Development*, 19:283–294, 1975.

[15] K.W. Ross, D. Tsang, and J. Wang. Monte Carlo summation and integration applied to multichain queueing networks. *Journal of the Association for Computing Machinery*, 41(6):1110–1135, November 1994.

[16] K.W. Ross and J. Wang. Asymptotically Optimal Importance Sampling for Product-Form Queueing Networks. *ACM Transactions on Modeling and Computer Simulation*, 3:244–268, 1993.

[17] K.W. Ross and J. Wang. Implementation of Monte Carlo Integration for the Analysis of product-form queueing networks. *Performance Evaluation*, 29(4):273–292, 1997.

[18] J. E. H. Shaw. A quasirandom approach to integration in Bayesian statistics. *Ann. Statist.*, 16:895–914, 1988.

[19] B. Tuffin. *Simulation accélérée par les méthodes de Monte Carlo et quasi-Monte Carlo : théorie et applications.* PhD thesis, Université de Rennes 1, Octobre 1997.

[20] B. Tuffin. Variance Reductions applied to Product-Form Multi-Class Queuing Network. *ACM Transactions on Modeling and Computer Simulation*, 7(4):478–500, 1997.

[21] J. Wang and K.W. Ross. Asymptotic Analysis for Closed Multiclass Queueing Networks in Critical Usage. *Queueing Systems: Theory and Applications*, 16:167–191, 1994.

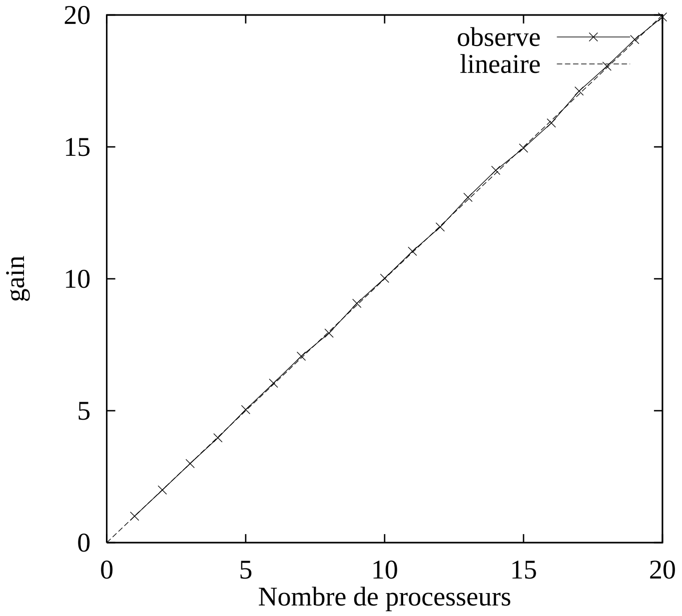

FIG. 1 – *Gain obtenu sur l'efficacité de l'estimateur par rapport au nombre de processeurs alloués.*