Randomness and Computation
Joint Workshop "New Horizons in Computing" and "Statistical Mechanical Approach to Probabilistic Information Processing" (18-21 July, 2005, Sendai, Japan)
Lecture Note in Tutorial Sessions

CFTP を用いたパーフェクトサンプリング

松井知己 [1] 来嶋秀治[2]
東京大学大学院情報理工学系研究科

1 はじめに

マルコフ連鎖を用いたサンプリングは，MCMC 法（マルコフ連鎖モンテカルロ法）等における重要な問題である．従来のサンプリング法では，初期状態から十分な回数推移させることによりサンプリングを行っているが，従来手法では (予め定めた) 有限回の推移の後サンプリングを行うため，その回数をどれほど多くしても「厳密な」サンプリングは不可能である．そのため，厳密なサンプリングからの誤差を設定して，必要な推移回数を算定する必要がある．算定を理論的に行うには，用いるマルコフ連鎖の収束の速さを議論しなければならないが，これは困難である場合が多い．実際には，推移回数の算定は経験に基づくものであることが多い．さらに，理論的，経験的どちらの手法においても，誤差を決定する必要があるが，誤差をどれだけ許容するかを決定する事は困難である場合が多い．例えば医療統計の検定手法において MCMC 法を用いる際，検定の誤り確率を踏まえた上で，サンプリング数に依存する計算誤差がある状況で，サンプル自体の誤差を決定するのは殆ど不可能と言ってよい．

与えられたマルコフ連鎖の定常分布に厳密に従うサンプリングができれば，これらの問題はすべて解消され，誤差の設定に思い悩む必要も一切無くなる．しかしながら，マルコフ連鎖は無限回の推移の後，定常分布に「厳密に」収束するのであって，有限回の推移では厳密には定常分布は得られない．Propp and Wilson によって提案された coupling from the past 法は，マルコフ連鎖の極限分布に厳密に従うサンプリングを可能とする驚くべき手法である．この手法は，無限の過去から推移しているマルコフ連鎖を仮想的に考え，現時点でサンプリングを行うという革新的なアイデアに基づく．本稿では「無限のシミュレーションを有限な（期待）時間で実現」してしまう，驚くべきアルゴリズムのアイデアを紹介する．

2 マルコフ連鎖と更新関数

本稿では，m 個の状態で構成される状態空間 $\Omega = \{s_1, s_2, \ldots, s_m\}$ と推移確率行列 P で定義されているマルコフ連鎖 $\mathcal{M}$ について議論する．すなわち，$\mathcal{M}$ が各時刻で取りうる状態は Ω 上のいずれかであり，時刻 t で状態 $x \in \Omega$ に居る時，時刻 $t+1$ の状態が $y \in \Omega$ である確率は $P(x, y)$ である．ただし，$P(x, y)$ は行列 P の x, y 成分を表す．

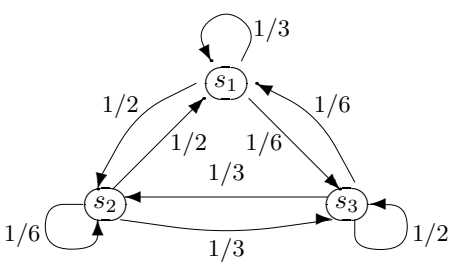

図 1: マルコフ連鎖の推移グラフ

[1]E-mail: tomomi@mist.i.u-tokyo.ac.jp
[2]E-mail: kijima@misojiro.t.u-tokyo.ac.jp

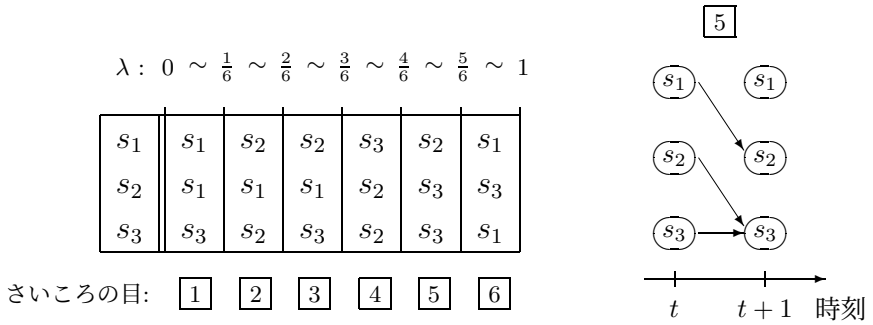

図 2: 更新関数の例

マルコフ連鎖 $\mathcal{M}$ の状態空間 Ω 上の確率分布 $\boldsymbol{\pi} = (\pi(s_1), \pi(s_2), \ldots, \pi(s_m))$ が $\boldsymbol{\pi}P = \boldsymbol{\pi}$ を満たす時，**定常分布**と呼ぶ．今後，マルコフ連鎖 $\mathcal{M}$ はエルゴード的であり，唯一の定常分布 $\boldsymbol{\pi}$ を持つと仮定する．このとき，マルコフ連鎖 $\mathcal{M}$ を無限回推移させると，その状態は定常分布 $\boldsymbol{\pi}$ に収束する．

マルコフ連鎖の推移は，一様実数乱数 $\Lambda \in [0,1)$ と，**更新関数** (update function) と呼ばれる（決定的な）関数 $\phi : \Omega \times [0,1) \to \Omega$ をもちいて表現することができる．更新関数 ϕ は，一様実数乱数 $\Lambda \in [0,1)$ と任意の $x, y \in \Omega$ に対して，$\Pr[\phi(x, \Lambda) = y] = P(x, y)$ を満たさなければならない．図 2 の左側の表は，図 1 のマルコフ連鎖に対する更新関数の例である．簡単のため，乱数の値が区間 $[(i-1)/6, i/6)$ $(i \in \{1, 2, \ldots, 6\})$ に含まれる事象をさいころの目 $\boxed{i}$ だと思うとよい．図 2 の右の図はさいころの目が $\boxed{5}$ であったときの時刻 t から $t+1$ への推移を図示したものである．もちろん，状態空間 Ω と推移確率行列 P をもつマルコフ連鎖の更新関数の決め方は**一意ではない**．更新関数の概念を再帰的に拡張すると，時刻 t_1 から t_2 $(t_1 < t_2)$ への推移も，乱数列 $\boldsymbol{\lambda} = (\lambda[t_1], \lambda[t_1+1], \ldots, \lambda[t_2-1])$ と $\Phi_{t_1}^{t_2}(x, \boldsymbol{\lambda}) \stackrel{\text{def.}}{=} \phi(\phi(\cdots\phi(x, \lambda[t_1]), \cdots, \lambda[t_2-2]), \lambda[t_2-1])$ で定義される関数 $\Phi_{t_1}^{t_2} : \Omega \times [0,1)^{t_2 - t_1} \to \Omega$ を用いて表すことができる．

マルコフ連鎖を用いてサンプリングする際，必要な推移回数の決定は重要な問題である．マルコフ連鎖の推移を（固定された）有限回数で打ち切ってしまうと厳密には定常分布では無い．これに対し Propp and Wilson [7, 8] は厳密なサンプリングを可能にする乱択アルゴリズムを提案した．このアルゴリズムは CFTP（coupling from the past：過去からのカップリング）と呼ばれる．

3 過去からのカップリング

以下に，状態空間 Ω と更新関数 ϕ をもつマルコフ連鎖 $\mathcal{M}$ に対する CFTP アルゴリズムを記す．

アルゴリズム 1 （CFTP アルゴリズム）
Step 1: シミュレーションの開始時刻を $T := -1$ とする．空列 $\boldsymbol{\lambda} := ()$ を用意する．
Step 2: 乱数 $\lambda[T]$ を生成し，数列 $\boldsymbol{\lambda}$ の先頭に挿入する．すなわち $\boldsymbol{\lambda} := (\lambda[T], \lambda[T+1], \ldots, \lambda[-1])$ とする．
Step 3: Ω のすべての状態 x について，x を初期状態とし，共通の乱数列 $\boldsymbol{\lambda}$ を用いて時刻 T から時刻 0 までマルコフ連鎖を推移させる．このとき，
(a) 時刻 0 で **coalesce** していれば $(\exists y \in \Omega, \forall x \in \Omega, y = \Phi_T^0(x, \boldsymbol{\lambda}))$，時刻 0 の状態 y を出力し，停止する．
(b) そうでなければ，シミュレーションの開始時刻を $T := T - 1$ として Step 2 に戻る．

アルゴリズム 1 が終了したときの $|T|$ の値を **coalescence 時間**と呼ぶ．

マルコフ連鎖の極限分布からのサンプリングは，(過去の方向に延びる）片側無限の乱数列をサンプリングし，対応するマルコフ連鎖の最終状態を見つけられれば良い．ここでアルゴリズムの coalesce の確認とは，無限長の乱数列の（過去の）ある時刻から現在までの有限長だけを見て同値類判定を行うことに対応する．このとき，乱数列がマイナス方向に無限であることが肝である．したがって coalesce しないからといって，**決して中断してはならない**．もし中断してしまう（ようにする）と，厳密性は失われる．

いま，アルゴリズム 1 の coalescence 時間を確率変数 T^* で表す．各反復での Step 3 のシミュレーション時間を考えれば，アルゴリズムの計算時間は $1+2+\cdots+|T^*| = \frac{|T^*|(|T^*|+1)}{2}$ に比例することがわかる．ここで，アルゴリズム 1 の Step 3 (b) で，過去へ戻るステップ幅を 2 倍 2 倍してみよう．

アルゴリズム 2 （標準 CFTP アルゴリズム）
Step 1: シミュレーションの開始時刻を $T := -1$ とする．空列 $\boldsymbol{\lambda} = ()$ を用意する．
Step 2: 乱数 $\lambda[T], \ldots, \lambda[\lceil T/2 \rceil - 1]$ を生成し，数列 $\boldsymbol{\lambda}$ の先頭に挿入する．すなわち，$\boldsymbol{\lambda} := (\lambda[T], \lambda[T+1], \ldots, \lambda[-1])$ とする．
Step 3: Ω のすべての状態について，共通の乱数列 $\boldsymbol{\lambda}$ を用いて時刻 T から時刻 0 まで推移させる．
(a) 時刻 0 で coalesce していれば $(\exists y \in \Omega, \forall x \in \Omega, y = \Phi_T^0(x, \boldsymbol{\lambda}))$，時刻 0 の状態 y を出力し，停止する．
(b) そうでなければ，シミュレーションの開始時刻を $T := 2T$ として Step 2 に戻る．

簡単に分かるように，このアルゴリズムもパーフェクトサンプリングを実現する．このアルゴリズムの計算時間は $1+2+4+\cdots+2^{\lceil \log_2 |T^*| \rceil} < 4|T^*|$ に比例するので，coalescence 時間の高々4 倍の時間のシミュレーションで時刻 0 の状態が得られる．

CFTP アルゴリズムの鍵は coalesce にあると言える．ならば「通常のマルコフ連鎖シミュレーション同様に時刻 0 からの推移を行い」，coalesce した時刻での状態を返せばよいのではないのか？と言う疑問を持つ方も居るかもしれない．このようなアルゴリズムを “前向きアルゴリズム” と呼ぼう．“前向きアルゴリズム” では，パーフェクトサンプリングが行えないことが簡単に理解できる例として図 3 の例を紹介する．この例では状態空間 $\{s_1, s_2, s_3, s_4\}$ を持ち，$P(x, y) = 1/4$ $(\forall (x, y) \in \Omega^2)$ を満たし，定常分布は $\boldsymbol{\pi} = (1/4, 1/4, 1/4, 1/4)$ である．このマルコフ連鎖に対して図 3 の左上のような更新関数を与える．この更新関数を用いて “前向きアルゴリズム” を行うと，得られる値は常に s_4 となってしまうのである．注意として，この更新関数を使っても，CFTP アルゴリズムで得られる値はもちろんパーフェクトサンプリングである．

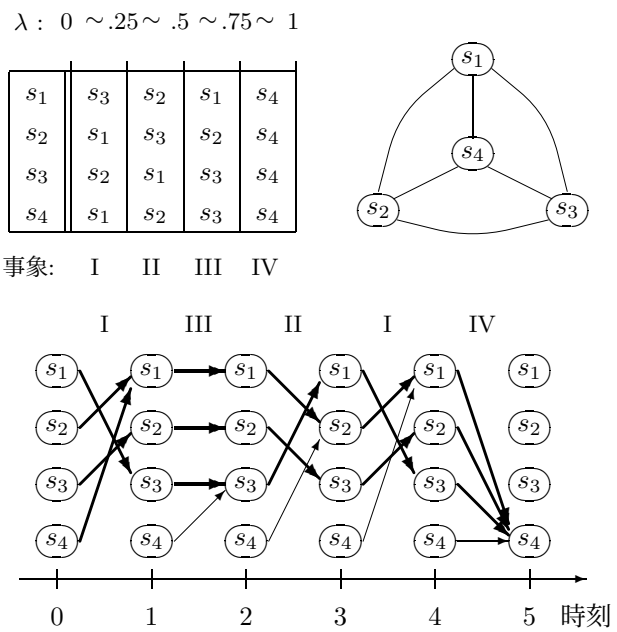

図 3: 前向きではいけない例

4 単調 CFTP

前節で紹介した CFTP アルゴリズムは，過去のある時点における全状態からのシミュレーションを実行すると言うものであった．しかし「全状態からのシミュレーションを実行」することは，実務上の多くの場合困難である．これを克服する一つの方法が単調 CFTP である．まず最初に CFTP アルゴリズムを設計するための「巧妙な更新関数」を導入しよう．

定義 1 更新関数が**単調**であるとは，次の 2 つの条件を満たすことである．
条件 1: 状態空間 Ω 上に半順序関係 “$\preceq$” が存在する．
条件 2: 任意の推移は半順序を保存する．すなわち，$x \succeq y$ を満たす任意の状態対 $(x, y) \in \Omega^2$ および任意の実数 $\lambda \in [0, 1)$ に対して，$\phi(x, \lambda) \succeq \phi(y, \lambda)$ が成り立つ．

単調な更新関数で定義されたマルコフ連鎖のことを，**単調マルコフ連鎖**と呼ぶ．いま，半順序集合 $(\Omega, \succeq)$ に対して，唯一の最大元 $x_{\max} \in \Omega$ および最小元 $x_{\min} \in \Omega$ が存在するとしよう．すなわち，任意の $x \in \Omega$ に対して $x_{\max} \succeq x \succeq x_{\min}$ が成り立つとする．このとき，次の命題が導かれる．

命題 2 更新関数 ϕ で定義されたマルコフ連鎖は半順序集合 $(\Omega, \succeq)$ に関して単調であり，$\exists x_{\max}, \exists x_{\min} \in \Omega$, $\forall x \in \Omega$, $x_{\max} \succeq x \succeq x_{\min}$ とする．この時，$T < 0$ および $\boldsymbol{\lambda} \in [0,1)^{|T|}$ に対して coalesce している（すなわち $\exists y \in \Omega$, $\forall x \in \Omega$, $y = \Phi_T^0(x, \boldsymbol{\lambda})$）ことの必要十分条件は $\Phi_T^0(x_{\max}, \boldsymbol{\lambda}) = \Phi_T^0(x_{\min}, \boldsymbol{\lambda})$ である．

以上の議論から，単調なマルコフ連鎖を設計できたならば，最大元と最小元の 2 状態からのみシミュレーションを行えば良いので，状態空間の大きさに依らず coalescence 時間に比例する時間でパーフェクトサンプリングが行える．標準的な単調 CFTP アルゴリズムは次のように定義される．

アルゴリズム 3 （単調 CFTP アルゴリズム）
アルゴリズム 2 の Step 3 を以下に変更する．
Step 3: 最大元 $x_{\max}$ と最小元 $x_{\min}$ について，共通の乱数列 $\boldsymbol{\lambda}$ を用いて時刻 T から時刻 0 までマルコフ連鎖を推移させる．このとき，
(a) coalesce していれば（$\exists y \in \Omega$, $y = \Phi_T^0(x_{\max}, \boldsymbol{\lambda}) = \Phi_T^0(x_{\min}, \boldsymbol{\lambda})$），時刻 0 の状態 y を出力し，停止する．
(b) そうでなければ，シミュレーションの開始時刻を $T := 2T$ として Step 2 に戻る．

定理 3 エルゴード的な単調マルコフ連鎖に対して，アルゴリズム 3 は確率 1 で有限停止し，アルゴリズム 3 の返す値は定常分布に厳密に従う．

イジングモデル: 以下では単調マルコフ連鎖の例として，頂点数 n のグラフ $G = (V, E)$ 上のイジングモデルを考えよう．イジングモデルでは，各頂点上のスピン $\{-1, +1\}$ が与えられた，状態 $\boldsymbol{x} \in \{-1, +1\}^V$ について議論する．温度 $T > 0$ において，状態 $\boldsymbol{x}$ はエネルギー $H(\boldsymbol{x}) \stackrel{\text{def.}}{=} -\frac{1}{T}\sum_{\{u,v\} \in E} x(u)x(v)$ をもつ．状態 $\boldsymbol{x}$ の存在確率 $\Pr(\boldsymbol{x})$ はエネルギー関数に対して，$\Pr(\boldsymbol{x}) = \frac{1}{Z(T)} \mathrm{e}^{-H(\boldsymbol{x})}$ で表わされる．上記の $Z(T)$ は分配関数（正規化定数）である．このイジングモデルを表現するマルコフ連鎖 $\mathcal{M}$ を次のように定義する．マルコフ連鎖 $\mathcal{M}$ は状態空間 $\{-1, +1\}^V$ を持ち，現在の状態 $X \in \{-1, +1\}^V$ から次の時刻の状態 X' への推移は次のように定義される．まず，G の頂点 $v \in V$ をひとつ，一様ランダムに選ぶ．頂点 v に隣接する頂点のうち，状態 X においてスピンが $+1$ の頂点の個数を $k_+(v, X)$，-1 の頂点の個数を $k_-(v, X)$ で表し，

$$p(v, X) \stackrel{\text{def.}}{=} \frac{\mathrm{e}^{\frac{2}{T}k_-(v,X)}}{\mathrm{e}^{\frac{2}{T}k_-(v,X)} + \mathrm{e}^{\frac{2}{T}k_+(v,X)}}$$

とする．このとき，確率 $p(v, X)$ で $X'(v) = -1$ とし，確率 $1 - p(v, X)$ で $X'(v) = +1$ とする．その他の頂点 $w \in V \backslash \{v\}$ は $X'(w) = X(w)$ とする．有限マルコフ連鎖 $\mathcal{M}$ はエルゴード的であり，詳細均衡方程式が成り立つ事から，定常分布は Gibbs 分布となる．

上記のマルコフ連鎖 $\mathcal{M}$ が単調であることを確認しよう．まず，状態空間 $\{-1, +1\}^V$ 上に半順序関係 "$\succeq$" を導入する．状態 $\boldsymbol{x}, \boldsymbol{y} \in \{-1, +1\}^V$ に対して $\boldsymbol{x} \succeq \boldsymbol{y} \stackrel{\text{def.}}{\Leftrightarrow} x(v) \geq y(v) \quad (\forall v \in V)$ とする．ただし，$+1 \geq -1$ である．この半順序集合 $(\{-1, +1\}^V, \succeq)$ において，あきらかに最大元および最小元 $\boldsymbol{x}_{\max}, \boldsymbol{x}_{\min} \in \{-1, +1\}^V$ が存在して，それぞれ $x_{\max}(v) = +1$ $(\forall v \in V)$, $x_{\min}(v) = -1$ $(\forall v \in V)$ を満たす．

次に，マルコフ連鎖 $\mathcal{M}$ を実現する更新関数 $\phi : \{-1, +1\}^V \times [0, n) \to \{-1, +1\}^V$ を以下のように定義する．いま，V の各頂点には 0 から $n-1$ までの番号が振ってあるとしよう．実数 $\lambda \in [0, n)$ に対し，その整数部分 $\lfloor \lambda \rfloor$ の番号を持つ頂点を $v \in V$ とする．このとき，次の時刻の状態 $\boldsymbol{z} = \phi(\boldsymbol{x}, \lambda)$ を，実数 λ の小数部分 $\lambda - \lfloor \lambda \rfloor$ を用いて，

$$\begin{aligned} z(v) &= \begin{cases} -1, & 0 \leq \lambda - \lfloor \lambda \rfloor < p(v, X), \\ +1, & p(v, X) \leq \lambda - \lfloor \lambda \rfloor < 1, \end{cases} \\ z(w) &= x(w), \quad w \in V \backslash \{v\}, \end{aligned}$$

と定義すると，ϕ はマルコフ連鎖 $\mathcal{M}$ を実現している．

更新関数 ϕ の単調性を示す．いま $\boldsymbol{x} \succeq \boldsymbol{y}$ として，実数 $\lambda \in [0, n)$ に対応する推移を考える．表記の簡単のため $\boldsymbol{x}' = \phi(\boldsymbol{x}, \lambda)$ および $\boldsymbol{y}' = \phi(\boldsymbol{y}, \lambda)$ と記し，$\lfloor\lambda\rfloor$ の番号を持つ頂点を v とする．このとき，更新関数によって v 以外の頂点のスピンは保存されるので，v 以外の任意の頂点 w に対して，$x'(w) \geq y'(w)$ が成り立つ．次に，頂点 v について $k_-(v, \boldsymbol{y}) \geq k_-(v, \boldsymbol{x})$ を考慮すると，$p(v, \boldsymbol{x}) \geq p(v, \boldsymbol{y})$ が成り立つ．このことは更新関数の定義から，

$$\begin{aligned} x'(v) = -1 \quad &\Rightarrow \quad y'(v) = -1, \\ y'(v) = +1 \quad &\Rightarrow \quad x'(v) = +1 \end{aligned}$$

が成り立つことを意味する．したがって $x'(v) \geq y'(v)$ となる．

定理 4 更新関数 ϕ は単調である．すなわち，$\boldsymbol{x} \succeq \boldsymbol{y}$ ならば「$\forall\lambda \in [0, n)$, $\phi(\boldsymbol{x}, \lambda) \succeq \phi(\boldsymbol{y}, \lambda)$」が成り立つ．

上記より，アルゴリズム 3 では $\boldsymbol{x}_{\max} = \mathbf{1}$ と $\boldsymbol{x}_{\min} = -\mathbf{1}$ の 2 状態からのみシミュレーションすれば良い．

今回紹介したもの以外の単調マルコフ連鎖として，タイリングのランダム生成に対するものがある（Propp のホームページ中にある [10] では，この単調 CFTP アルゴリズムの動く様子が見られる）．また，著者らによっていくつかの結果が得られている [4, 5] 更新関数の単調性を用いることなく CFTP アルゴリズムが効率的に実行できるものも提案されている．例えば Propp and Wilson は，連結無向グラフ上の（根付き）全張木に対して，(単調でない）CFTP を用いたパーフェクトサンプリング法を提案している [8]．全長木は，数え上げの多項式時間解法を利用したパーフェクトサンプリングが可能であるが，これに比べて Propp and Wilson の手法はおどろくほどシンプルである．

5 記憶領域の削減

単調 CFTP により，高速なパーフェクトサンプリングの設計が可能となった．しかし，いざ（単調）CFTP アルゴリズムを実装しようとすると，思わぬ問題点が存在する．それは乱数の記憶領域である．CFTP アルゴリズムでは coalesce が確認できなかった場合には，更に過去に遡ってシミュレーションを行わなければならないが，このとき，生成された乱数を記憶しておかなければならない．たとえば，単調 CFTP アルゴリズムならば，計算時間に比例する記憶領域が要求されるのである．以下ではこの点について改良された，Wilson の Read Once アルゴリズム [9] について述べる

これまでは時刻 $t < 0$ から時刻 0 の間の推移だけを考えていたが，少し一般化して，時刻 t_1 から時刻 t_2 までの推移における，(全状態からの）colaesce とは，(乱）数列 $\boldsymbol{\lambda} \in [0, 1)^{t_2 - t_1}$ に対して，$\exists y \in \Omega, \forall x \in \Omega$, $y = \Phi_{t_1}^{t_2}(x, \boldsymbol{\lambda})$ が成り立っている事と定義する．

以下，マルコフ連鎖の推移を定義している更新関数は，coalesce する確率が正（非零）であると仮定する．図 4 は t_1 から t_2 の間の推移を模式的に表したものである．図 4 の左側は，全状態について数列 $\boldsymbol{\lambda}_s$ を用いて時刻 t_1 から時刻 t_2 まで推移を行い，時刻 t_2 で coalesce している様子を表している．右側は，数列 $\boldsymbol{\lambda}_f$ を用いると時刻 t_2 では coalesce していない様子である．図 4 は単調マルコフ連鎖の最大元と最小元からの推移を表していると思って見るとイメージしやすい．ただし以下の議論では，計算量の議論の一部を除いて，更新関数の単調性の仮定は必要としない．

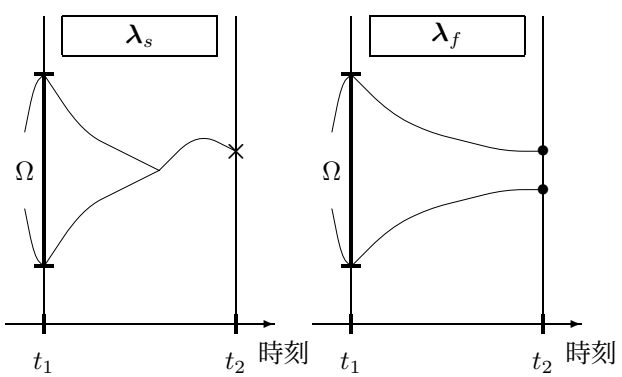

図 4: 時刻 t_1 から時刻 t_2 までの推移の模式図

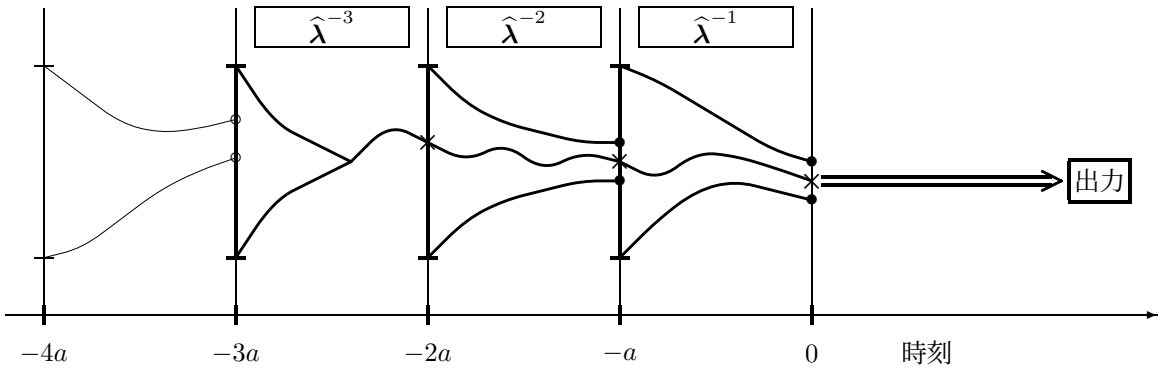

図 5: a 遡る CFTP アルゴリズムの概念図

5.1 Read Once アルゴリズム

まず，特殊な CFTP アルゴリズムを導入する．

アルゴリズム 4 （a 遡る CFTP アルゴリズム）
Step 0: 正整数 $a > 0$ を決める．
Step 1: シミュレーションの開始時刻を $T := 0$ とする．空列 $\boldsymbol{\lambda} = ()$ を用意する．反復回数 $i := 1$ とする．
Step 2: シミュレーションの開始時刻を $T := T - a$ とする．
Step 3: 乱数列 $\widehat{\boldsymbol{\lambda}}^{-i} = (\widehat{\lambda}[T], \ldots, \widehat{\lambda}[T+a-1])$ を生成し，数列 $\widehat{\boldsymbol{\lambda}}$ の先頭に挿入する．すなわち，$\widehat{\boldsymbol{\lambda}} := (\widehat{\boldsymbol{\lambda}}^{-i}; \widehat{\boldsymbol{\lambda}}^{-i+1}; \cdots; \widehat{\boldsymbol{\lambda}}^{-1}) = (\widehat{\lambda}[T], \widehat{\lambda}[T+1], \ldots, \widehat{\lambda}[-1])$ とする．
Step 4: Ω のすべての状態について，共通の乱数列 $\widehat{\boldsymbol{\lambda}}^{-i}$ を用いて時刻 T から時刻 $T+a$ までマルコフ連鎖を推移させる．このとき，
(a) もし，時刻 $T+a$ で coalesce していれば（すなわち，$\exists z \in \Omega, \forall x \in \Omega, z = \Phi_T^{T+a}(x, \widehat{\boldsymbol{\lambda}}^{-i})$），時刻 0 の状態 $y = \Phi_T^0(x, \widehat{\boldsymbol{\lambda}})$ を出力し，停止する．
(b) そうでなければ，反復回数を $i := i+1$ として Step 2 に戻る．

図 5 はアルゴリズム 4 が 3 反復（$i = 3$）で終了した様子を表している．上記のアルゴリズムがパーフェクトサンプリングを実現している事は容易に分かるであろう．アルゴリズム 4 において正整数 a の決め方は重要である．アルゴリズム 4 の計算時間について考えると，時間 a で coalesce する確率を p とすると，アルゴリズムの期待計算時間は a/p である．すなわち，coalesce するような乱数列が存在しないほど a が小さいと，アルゴリズムは停止しない．また，a を coalescence 時間の期待値 $\mathrm{E}[T_*]$ 程度にすると p は定数となるので，例えば単調 CFTP ならば期待計算時間は $\mathrm{O}(\mathrm{E}[T_*])$ となり，標準単調 CFTP と同程度の計算時間となる．以上の準備をふまえて，Read Once アルゴリズムを紹介しよう．

アルゴリズム 5 （Read Once アルゴリズム）
Step 0: 正整数 $a > 0$ を与える．
Step 1: 乱数列 $\boldsymbol{\lambda}^1 = (\lambda[0], \lambda[1], \ldots, \lambda[a-1])$ を生成する[3]．Ω のすべての状態について，共通の乱数列 $\boldsymbol{\lambda}^1$ を用いて，時刻 0 から時刻 a までマルコフ連鎖を推移させる．このとき，
(a) もし時刻 0 と時刻 a の間で coalesce していれば（すなわち，$\exists z \in \Omega, \forall x \in \Omega, z = \Phi_0^a(x, \boldsymbol{\lambda}^1)$），変数 y に時刻 a の状態を代入する．すなわち $y := \Phi_0^a(x, \boldsymbol{\lambda}^1)$ とする．反復回数を $i := 1$ とする．
(b) そうでなければ，Step 1 へ戻る．
Step 2: 乱数列 $\boldsymbol{\lambda}^{i+1} = (\lambda[ia], \lambda[ia+1], \ldots, \lambda[(i+1)a-1])$ を生成する[4]．Ω のすべての状態について，共通の乱数列 $\boldsymbol{\lambda}^{i+1}$ を用いて，時刻 ia から時刻 $(i+1)a$ まで推移させる．特に $y' = \Phi_{ia}^{(i+1)a}(y, \boldsymbol{\lambda}^{i+1})$ とする．

[3]便宜上 $\boldsymbol{\lambda}^1$ と記述するが，この数列を記憶する必要は無い．
[4]脚注 3 と同様，数列 $\boldsymbol{\lambda}^{i+1}$ を記憶する必要は無い．

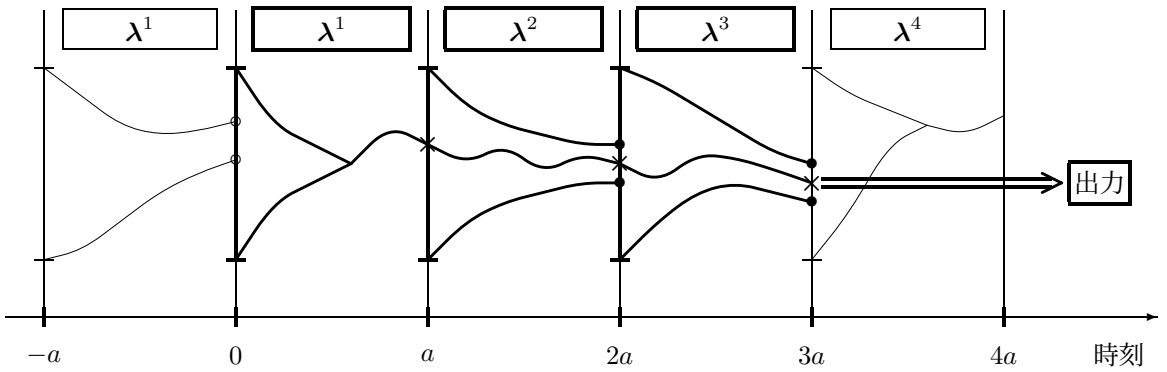

図 6: Read Once アルゴリズムの概念図

(a) 時刻 ia と時刻 $(i+1)a$ の間で coalesce していれば（$\exists z \in \Omega,\, \forall x \in \Omega,\, z = \Phi_{ia}^{(i+1)a}(x, \boldsymbol{\lambda}^{i+1})$ ），時刻 ia の状態 y を出力し，停止する．
(b) そうでなければ，$y := y'$ と更新する．反復回数を $i := i+1$ とし，Step 2 に戻る．

図 6 は Read Once アルゴリズム（アルゴリズム 5）が 3 反復（$i = 3$）で終了した様子を表している．

定理 5 エルゴード的なマルコフ連鎖に対して，ある a が存在して，アルゴリズム 5 は確率 1 で有限停止し，アルゴリズム 5 の返す値は定常分布に厳密に従う．

Read Once アルゴリズム（アルゴリズム 5）のアイデアは，a 遡る CFTP アルゴリズム（アルゴリズム 4）の模倣である．アルゴリズムからもわかるとおり，Read Once アルゴリズムでは乱数を記憶する必要が無い．Read Once アルゴリズムの期待計算時間も，a 遡るアルゴリズムと同様であることがわかる．ただ，気をつけなければならないのは，Read Once アルゴリズムの Step 1 で，coalesce した状態を得るための前処理が行われていることから，正確には，Read Once アルゴリズムで 1 つのサンプルを得るのに必要な期待計算時間は a 遡る CFTP アルゴリズムの 2 倍程度であることがわかる．

さらに，Read Once の良いところは，アルゴリズムの出力する状態 y と，アルゴリズム 5 の終了判定のために行った推移の coalesce した状態（Step 2 (a) で得られた状態 z）は互いに独立，となる点である．したがって，サンプルが複数個必要な時に，アルゴリズム 5 を連続して実行することができる．具体的にはサンプルが N 個必要な場合にはアルゴリズム 5 の Step 2 (a) を次のように置き換えれば良い．

Step 2: (a)′ もし時刻 $(i+1)a$ で coalesce していれば y を出力する．もし N 個のサンプルが出力されていたら停止する．そうでなければ，$y := \Phi_{ia}^{(i+1)a}(x, \boldsymbol{\lambda}^{i+1})$ とし，$i := 1$ と更新して Step 2 に戻る．

このように置き換えられた Read Once アルゴリズムで N 個のサンプルを生成するのに必要な期待計算時間は $(N+1)a/p$ となるので，a 遡る CFTP アルゴリズムとほぼ同じ計算時間で，乱数を記憶することなくサンプルが得られることがわかる．

Read Once アルゴリズムの計算時間は a の決め方に大きく依存する．特に a が小さいと，アルゴリズムは全く止まらなくなってしまう．しかし，a をどのように決めたらよいものだろうか．最後に，パラメータ a を全く与えないで Read Once アルゴリズムを実現するという，驚くべき Twin Run アルゴリズムを紹介しよう．

5.2 Twin Run アルゴリズム

Twin Run アルゴリズムのアイデアは，サンプリングとは別の乱数列を生成してステップバックする時間を決めようと言うものである．まず準備として，アルゴリズム中で呼び出す 2 つの手続きを導入する．

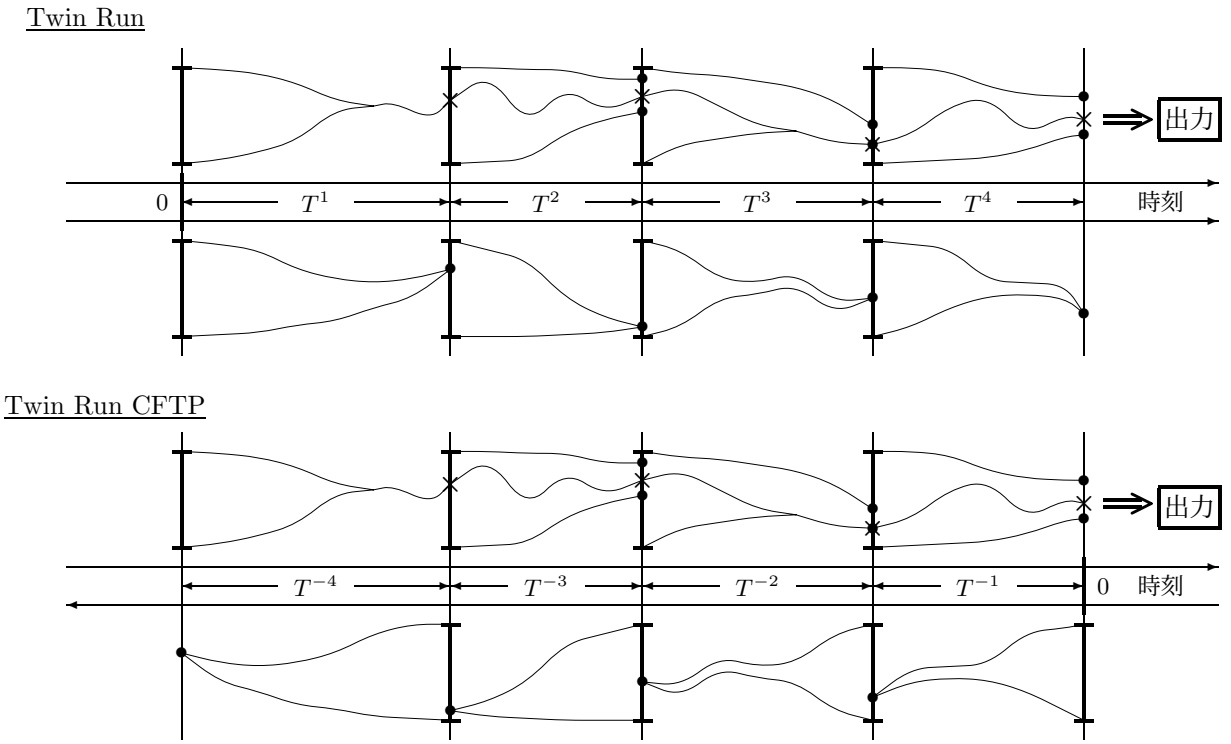

図 7: Twin Run アルゴリズムの概念図

プロセス A (*-successful)
Step 1: シミュレーションの終了時刻 T を $T := 1$ に設定する．
Step 2: 2 つの乱数 $\lambda_1[T-1]$, $\lambda_2[T-1]$ を生成する．便宜上，$\boldsymbol{\lambda}_1 = (\lambda_1[0], \lambda_1[1], \ldots, \lambda_1[T-1])$, $\boldsymbol{\lambda}_2 = (\lambda_2[0], \lambda_2[1], \ldots, \lambda_2[T-1])$ と表す[5]．数列 $\boldsymbol{\lambda}_1$ と $\boldsymbol{\lambda}_2$ を用いて，それぞれ推移を実行する．このとき，
(a) 数列 $\boldsymbol{\lambda}_1$ および $\boldsymbol{\lambda}_2$ について，ともに時刻 T で coalesce している[6]場合．先に coalesce していた乱数列の添え字を $j \in \{1, 2\}$ とする[7]．同時の場合は 1/2 の確率で j を 1 または 2 とする．数列 $\boldsymbol{\lambda}_j$ に対する時刻 T の状態 $z_j = \Phi_0^T(x, \boldsymbol{\lambda}_j)$ を返して終了する．
(b) それ以外の時，$T := T + 1$ とし Step 2 に戻る．

プロセス B (*-failing)
Step 0: 入力された状態を $y \in \Omega$ とする．
Step 1: シミュレーションの終了時刻 T を $T := 1$ に設定する．
Step 2: 2 つの乱数 $\lambda_1[T-1]$, $\lambda_2[T-1]$ を生成する．便宜上，$\boldsymbol{\lambda}_1 = (\lambda_1[0], \lambda_1[1], \ldots, \lambda_1[T-1])$, $\boldsymbol{\lambda}_2 = (\lambda_2[0], \lambda_2[1], \ldots, \lambda_2[T-1])$ と表す．数列 $\boldsymbol{\lambda}_1$ と $\boldsymbol{\lambda}_2$ を用いて，それぞれ推移を実行する．特に $y_1 := \Phi_0^T(y, \boldsymbol{\lambda}_1)$, $y_2 := \Phi_0^T(y, \boldsymbol{\lambda}_2)$ とする．このとき，
(a) 数列 $\boldsymbol{\lambda}_1$ と $\boldsymbol{\lambda}_2$ の少なくとも一方が時刻 T で coalesce している場合．coalesce していない乱数列の添え字を $j \in \{1, 2\}$ とする[8]．同時の場合は 1/2 の確率で j を 1 または 2 とする．状態 $y_j \in \Omega$ を返して終了する．
(b) それ以外の時，$T := T + 1$ とし Step 2 に戻る．

以上の準備の下，Twin Run アルゴリズムを記述する．

[5]例えば単調マルコフ連鎖ならば，この乱数列を保存せずにアルゴリズムを実行することができる．
[6]$\exists z_1, z_2 \in \Omega$, $\forall x \in \Omega$, $z_1 = \Phi_0^T(x, \boldsymbol{\lambda}_1)$, $z_2 = \Phi_0^T(x, \boldsymbol{\lambda}_2)$.
[7]たとえば，$\exists z \in \Omega$, $\forall x \in \Omega$, $z = \Phi_0^{T-1}(x, \boldsymbol{\lambda}_1)$ なら $j = 1$.
[8]たとえば，$\exists x_1, x_2 \in \Omega$, $\Phi_0^T(x_1, \boldsymbol{\lambda}_2) \neq \Phi_0^T(x_2, \boldsymbol{\lambda}_2)$ なら $j = 2$.

アルゴリズム 6 (Twin Run アルゴリズム)
Step 1: プロセス A を実行し，出力を x とする．1/2 の確率で (a),(b) のいずれかを実行する．
(a) $x \in \Omega$ を出力しアルゴリズムを終了する．
(b) Step 2 に進む．
Step 2: 入力 $x \in \Omega$ を与えて手続き B を実行し，変数 $x \in \Omega$ を出力された値に更新する．1/2 の確率で次の (a), (b) のいずれかを実行する．
(a) $x \in \Omega$ を出力しアルゴリズムを終了する．
(b) Step 2 にもどる．

Twin Run アルゴリズム（アルゴリズム 6）は，次の CFTP アルゴリズムを模倣する．

アルゴリズム 7 (Twin Run CFTP アルゴリズム)
Step 1: シミュレーションの開始時刻を $T := 0$ とする．空列 $\boldsymbol{\lambda} = ()$ を用意し，反復回数を $i := 1$ にする．
Step 2: 前向き coupling を実行し，その coalescence 時間を $T^{-i} > 0$ とする．シミュレーションの開始時刻を $T := T - T^{-i}$ とする．
Step 3: 乱数列 $\boldsymbol{\lambda}^{-i} = (\lambda[T], \ldots, \lambda[T + T^{-i} - 1])$ を生成し，数列 $\boldsymbol{\lambda}$ の先頭に挿入する．すなわち，$\boldsymbol{\lambda} := (\boldsymbol{\lambda}^{-i}; \boldsymbol{\lambda}^{-i+1}; \cdots; \boldsymbol{\lambda}^{-1}) = (\lambda[T], \lambda[T+1], \ldots, \lambda[-1])$ とする．
Step 4: Ω のすべての状態について，共通の乱数列 $\boldsymbol{\lambda}^{-i}$ を用いて時刻 T から時刻 $T + T^{-i}$ までマルコフ連鎖を推移させる．このとき，
(a) もし，時刻 $T + T^{-i}$ で coalesce していれば[9]（すなわち，$\exists z \in \Omega, \forall x \in \Omega, z = \Phi_T^{T+T^{-i}}(x, \boldsymbol{\lambda}^{-i})$），時刻 0 の状態 $y = \Phi_T^0(x, \boldsymbol{\lambda})$ を出力し，停止する．
(b) そうでなければ，反復回数を $i := i + 1$ として Step 2 に戻る．

アルゴリズム 6 と 7 が本質的に等しい事は容易に分かる．以下ではアルゴリズム 7 の基本的なアイデアを簡単に記す．図 7 の上半分は，Twin Run アルゴリズムが 4 反復（$i = 4$）で終了した様子を，下半分はアルゴリズム 7 が 4 反復（$i = 4$）で終了した様子を表している．図 7 の Twin Run CFTP（下半分）の上段は，アルゴリズム 4 で出力する状態を決定する推移を示しており，これは本質的に Twin Run（上半分）の上段と同じものである．図 4 最下段の時間軸が右から左へ向かっているのは，値 T^{-i} を決定するのに，(別の乱数列を用いて) 前向き coupling を行っている事を表している．実はアルゴリズム 7 は, a 遡るアルゴリズム（アルゴリズム 4）では固定されていた遡る時間（a）を，反復毎に Step 2 で決定するように変更したものである.（是非，アルゴリズム 4 と 7 を直接比較していただきたい.）アルゴリズム 7 は本質的に 2 本の乱数列を用いているが，出力を決定するのに用いているのはそのうち 1 本（上段）だけであり，2 本目の乱数列（下段）は遡る幅を決定する時計の役割にしか用いていない．

確率変数 T^{-i} の分布関数は前向き coupling の coalescence 時間 T^* の分布関数に等しく，これは，1 本目の乱数列 (上段) を使った CFTP の coalescence 時間 T_* の分布関数にも等しい．したがってアルゴリズム 4 の各反復において，(Step 4(a) で) アルゴリズムが終了する確率はいつも 1/2 である．すなわちアルゴリズム 7 は，coalescence 時間 T_* を知ることなく，アルゴリズム 4 を毎反復 1/2 の確率で終了させる驚くべき仕掛けとなっている．このことから明らかなように，アルゴリズム 7 の反復回数の期待値はたったの 2 である．

アルゴリズム 6 と 7 が本質的に等しいことから，以下の定理が成り立つ．

定理 6 エルゴード的な有限マルコフ連鎖に対して，Twin Run アルゴリズム（アルゴリズム 6）が確率 1 で有限停止し，アルゴリズムの返す値は定常分布に厳密に従う．

Twin Run アルゴリズムの驚愕すべき点は，パラメータ a の設定を不要とする巧妙なアイデアに加えて，手続き A, B を持ち込むことで，記憶領域を殆ど必要としないという成果を同時に達成している点であろう．

[9]正確には，coalescence 時間がちょうど T^{-i} の場合は確率 1/2 で **(b)** に進む．

6　まとめ

本稿では，マルコフ連鎖を用いたパーフェクトサンプリング法として，CFTP 及び単調 CFTP，Read Once アルゴリズムを紹介した．他の方法としては，Fill による interruptible アルゴリズム [2] がある．連続なマルコフ連鎖に対するパーフェクトサンプリング法としては，Murdoch and Green が 2 段階カップリングのアイデアを用いたアルゴリズムを提案している [6].

CFTP については，Propp and Wilson の論文 [7, 8]，Dimakos によるガイド [1] が詳しい．入門には学生用のテキストブック [3]（とても薄い！）が非常に分かりやすく，お薦めである．また著者らは最近日本語解説記事をまとめている [11]．CFTP の拡張や様々なモデルへの適用については，現在も研究が続いている．近年の専門書では，CFTP に１節が割かれている事も多い．著者らは，CFTP の開発と研究及び利用が，今後更に広がってゆくと予想している．

参考文献

[1] X. K. Dimakos : “A guide to exact simulation,” International Statistical Review, **69** (2001), 27–48.

[2] J. Fill : “An interruptible algorithm for perfect sampling via Markov chains,” The Annals of Applied Probability, **8** (1998), 131–162.

[3] O. Häggström : *Finite Markov Chains and Algorithmic Application*, London Mathematical Society, Student Texts **52**, Cambridge University Press, 2002.

[4] S. Kijima and T. Matsui : “Polynomial time perfect sampling algorithm for two-rowed contingency tables,” Mathematics and Computer Science III, M. Drmota, P. Flajolet, D. Gardy and B. Gittenberger (Ed.), Birkhäuser, 175–186, 2004.

[5] S. Kijima and T. Matsui : “Rapidly mixing chain and perfect sampler for logarithmic separable concave distributions on simplex,” Proceedings of the 2005 International Conference on the Analysis of Algorithms, Discrete Mathematics and Computer Science, Proceedings Series Volume AD, 369–380, 2005.

[6] D. J. Murdoch and P. J. Green : “Exact sampling from a continuous state space,” Scandinavian Journal of Statistics, **25** (1998), 483–502.

[7] J. Propp and D. Wilson : “Exact sampling with coupled Markov chains and applications to statistical mechanics,” Random Structures and Algorithms, **9** (1996), 232–252.

[8] J. Propp and D. Wilson : “How to get a perfectly random sample from a generic Markov chain and generate a random spanning tree of a directed graph,” Journal of Algorithms, **27** (1998), 170–217.

[9] D. Wilson : “How to couple from the past using a read-once source of randomness,” Random Structures and Algorithms, **16** (2000), 85–113.

[10] http://www.math.wisc.edu/˜ propp/tiling/www/applets/

[11] 来嶋秀治, 松井知己 :「完璧にサンプリングしよう！」オペレーションズ・リサーチ, **50** (2005), 第一回 169–174 (no. 3), 第二回 264–269 (no. 4), 第三回 329–334 (no. 5).