計算代数の direct sampler への応用
An Application of Computer Algebra to Direct Samplers

高山信毅 *
Nobuki Takayama
神戸大学
Department of Mathematics, Kobe University †
間野修平
Shuhei Mano
統計数理研究所
The Institute of Statistical Mathematics

Abstract

Diaconis-Sturmfels (1998) は行列 A のきめる affine toric ideal の生成元が A できまるある分布の Markov chain Monte Carlo (MCMC) simulation のための Markov 基底を与えることを示した. さらにこの生成元は Gröbner 基底の計算で求めることが可能である.

A 超幾何系はこの affine toric ideal と一階の方程式系できまる方程式系である. 上記の分布の分配関数 (または正規化定数) はこの方程式系の解となる. 間野はこの分配関数の満たす漸化式 (contiguity relation) を使えば効率的な direct sampler を構成できることを示した. この漸化式を求めるにはさまざまな計算代数の手法が適用可能である.

この小文は講演内容に従い, 間野の direct sampler を概説し, また最近得られたいくつかの結果の要約を述べる. [1].

Abstract

Diaconis-Sturmfels (1998) show that any set of generators of the affine toric ideal for a matrix A gives a Markov basis for the Markov chain Monte Carlo (MCMC) simulation for a distribution associated to A. Moreover, a set of generators can be obtained by Gröbner basis computation.

The A-hypergeometric system is a system of differential equations consisting of affine toric ideal in the ∂_i space and a set of first order differential equations. The partition function (or the normalizing constant) of the distribution associated to A is a solution of this system of differential equations. Mano show that recurrence relations (or contiguity relations) of the partition function gives an efficient direct sampler. We can apply several methods in computer algebra to derive recurrence relations for the direct sampler.

In this note, we shortly introduce Mano's direct sampler and present a sketch of our recent results on the direct sampler.

*平代達哉君 (神戸大) の修論の一部結果も紹介する

†takayama@math.kobe-u.ac.jp

[1] 関連の研究も含めて参考文献表は hgm OpenXM (検索) で取得できる. この小文では文献は本文に掲載し文末の文献表は用意しなかった. 講演のスライドは

1. http://www.math.kobe-u.ac.jp/HOME/taka/2018/rims-2018.pdf,
2. http://www.math.kobe-u.ac.jp/HOME/taka/2018/rims-2018-en.pdf(英語版).

Sampler とは Sampler とは “与えられた分布に従う乱数 (ベクトル)” を生成するアルゴリズムのことである.

例 (二項分布)

$$u_1 + u_2 = \beta, u_i \in \mathbf{N}_0$$

を満たす数のベクトル $u = (u_1, u_2)$ を分布が

$$\frac{\beta!}{u_1!u_2!} p_1^{u_1} p_2^{u_2} \tag{1}$$

となるようにランダムに生成したい (ここで $p_i \geq 0$, $p_1 + p_2 = 1$). つまり random variable[2] U が値 u を取る確率 $P(U = u)$ について次の式が成り立つようにしたい.

$$P(U = u) = (1) \text{ 式}$$

たとえば $\beta = 2$, $p_i = 1/2$ なら

$$P(U = (0,2)) = \frac{1}{4}, P(U = (1,1)) = \frac{1}{2}, P(U = (2,0)) = \frac{1}{4}$$

統計システム R にはこれを遂行するアルゴリズムが実装されていて, たとえば下記のようにサンプル (乱数) を得ることができる.

```
rbinom(20,size=2,prob=1/2);
[1] 1 2 1 2 1 1 0 2 0 1 1 2 2 0 2 2 1 2 1 1
```

出力は $(1,1), (2,0), (1,1), (2,0), (1,1), (1,1), (0,2), \ldots$ に対応. ちなみにヒストグラムを書くには

```
hist(rbinom(2000,size=2,prob=1/2))
```

と入力すればよい.

Direct sampler でこの乱数を作るには?

Input: β, p_1, p_2
Output: (u_1, u_2)

1. $(c_1, c_2) = (0, 0)$ (init count vector)
2. $e_1 = \frac{p_1}{p_1+p_2}$, $e_2 = \frac{p_2}{p_1+p_2}$.
3. $[0,1]$ を $e_1 : e_2$ の区間 E_1, E_2 に分割.
4. $[0,1]$ に値をもつ一様乱数 t を一つ生成.
5. **if** $t \in E_1$, **then** $c_1 ++$, $\beta --$ **else if** $t \in E_2$, **then** $c_2 ++$, $\beta --$.
6. **if** $\beta > 0$, **then** goto 4 **else** return $u = (c_1, c_2)$.

例 $\beta = 3, p_i = 1/2$ でこの direct sampler をプログラムで実装するなり, 教室で鉛筆倒しなどで, 実験してみれば, その分布は以下のようになるはずである.

$(3,0)$ で $\frac{1}{8}$, $(2,1)$ で $\frac{3}{8}$, $(1,2)$ で $\frac{3}{8}$, $(0,3)$ で $\frac{1}{8}$.

定理 1 (よく知られている)
(u_1, u_2) を得る確率は (1) 式.

Proof. Step 5 で選ばれる index の列を $i_1, i_2, \ldots, i_\beta$ とする. i_j は 1 か 2 である. $\sharp\{k \mid i_k = 1\} = c_1$, $\sharp\{k \mid i_k = 2\} = c_2$, であった. よってこの index 列を得る確率は $p_1^{c_1} p_2^{c_2}$. 同じ c_1, c_2 の時 $1, 2$ の並べ方は $\binom{\beta}{c_1}$ とおり. □

ちなみに R の `rbinom` コマンドはより効率的な乱数生成アルゴリズムを遂行している.

[2] シミュレーションの立場では乱数と思って良い

A 分布とは?　A 分布はこの二項分布を一般化したものであり, 分割表の上の超幾何分布, Young 図形達のある分布等, 多くの重要な分布を含む. この A 分布を定義しよう.

A: $d \times n$ 行列. 整数成分[3]. $p \in \mathbf{R}^n_{\geq 0}$, $\beta \in \mathbf{N}^d_0$ を固定.

$$Z_A(\beta; p) = \sum_{Au=\beta, u \in \mathbf{N}^n_0} \frac{p^u}{u!} \tag{2}$$

とおいて, $u \in \mathbf{N}^n_0$ に対して A 分布を次の式できめる.

$$P(U = u) = \frac{p^u}{u! Z_A(\beta; p)} \tag{3}$$

ここで $u! = u_1! \cdots u_n!$.
例: $A = (1, 1)$ なら二項分布. この時 $\beta_1! Z_A(\beta; p) = (p_1 + p_2)^\beta$.

間野は A 分布に対して direct sampler アルゴリズムを次の論文で与えた “S.Mano, The A-hypergeometric System Associated with the Rational Normal Curve and Exchangeable Structures, Electronic Journal of Statistics 11 (2017), 4452–4487” [4].

direct sampler アルゴリズム (間野, 2018)

Input: β, p
Output: c

1. $c := (0, 0, \ldots, 0)$ (init count vector)
2. $e_i := \frac{p_i Z(\beta - a_i; p)}{\beta_1 Z(\beta; p)}$, $i = 1, \ldots, n$[a].
3. $[0, 1]$ を $e_1 : e_2 : \ldots : e_n$ に分割.
4. $[0, 1]$ に値を持つ一様乱数 t を一つ生成.
5. t が e_j の領域に入ったら c_j を 1 増やす. $\beta := \beta - a_j$.
6. $\beta \neq 0$ なら goto 2.

終了したとき c は $Ac = \beta^b$ を満たす. なお a_i は行列 A の i 列.

[a] $\beta - a_i \notin \mathbf{N}^d_0$ なら $e_i = 0$. $|\beta| = \beta_1 + \ldots + \beta_d$
[b] 計算途中の β でなく input の β

さてこの direct sampler アルゴリズムでは Step 2 で異なるパラメータを持つ分配関数 Z の比を計算する必要がある. $Z_A(\beta; p)$ を直接定義から計算するのは β_1 や A が大きいと計算時間の点で困難が増す[5]. あとで述べるように A 超幾何系の性質を用いるとこの異なるパラメータを持つ分配関数の間の漸化式を求めることが原理的には可能であることがわかる. 間野は $A = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 0 & 1 & 2 & \cdots & n \end{pmatrix}$ 等の時にこの漸化式を具体的に求めて Young 図形上のある分布についての効率的な direct sampler アルゴリズムを与えた.

[3] さらに条件として, 1 行目の成分は全部 1. rank $= d$ も仮定しておく.
[4] https://projecteuclid.org/euclid.ejs/1510887943
[5] このような例については
http://www.math.kobe-u.ac.jp/HOME/taka/2015/2015-07-osaka-slides.pdf の p.5,6 に述べられている.

ここで注目したいのは漸化式を計算代数で求める問題は計算代数, 数式処理の分野で 30 年以上研究が続いていることである. 大雑把にくくると次のようになる.

1. 1980 年代後半の D.Zeilberger の研究以降計算代数の問題に. 参考: "$A = B$"[6]. その後,
2. 微分差分作用素のグレブナー基底で $(I + (S-1)D_n) \cap D_{n-1}$ を求める. ここで D_n は多項式係数の微分差分作用素環. S は差分作用素.
3. Creative Telescoping 法で $(I + (S-1)R_n) \cap R_{n-1}$ を求める. ここで R_n は有理式係数の微分差分作用素環.

間野 direct sampler の性質

定理 2

1. *A*-超幾何系の *Pfaffian* はグレブナー基底で計算可能. これが漸化式 *(contiguity)* も与えていて遷移確率 e_i が漸化式で計算できる[7].
2. N 個の乱数を計算するための計算量は $O(r^2\beta_1 N)$ プラス *Gröbner* 基底計算の計算量. ここで r は A の *normalized volume*. また有理数の四則の計算量は $O(1)$ とする.

補足:

1. MCMC[8] の計算量は $O(n(N * T + (\text{burn-in の回数})))$ [9].
2. A 超幾何系では Pfaffian = contiguity となる. Pfaffian の導出は計算量の多い計算であるが, 特別な A 超幾何系に対しては理論的考察で Pfaffian が導出されている. たとえば後藤, 松本の $E(k,n)$ の contiguity relation は Gröbner 基底によるものより効率的な分割表の sampler を提供している[10].

MCMC に対する direct sampler の利点は, burn-in, thinning などのパラメータ調整が不要 (たとえば, 問題に対する経験不要) であることである. 十分な数のサンプルを生成すればシミュレーション結果は十分信頼のおけるものである. 一方, この direct sampler の欠点は定理が示しているように MCMC に比べ速度が遅いことである. 定理からわかるように β_1 や A の normalized volume r が大きくなると, たとえ contiguity がわかっているとしても遅くなる. たとえば 3×3 分割表では $r = \binom{4}{2} = 6$ であるが 5×5 分割表では $r = \binom{8}{4} = 70$ であり $O(r^2)$ の効果で急激に遅くなる. さらに我々の実装実験では `double` 型の利用では急速な誤差の増大のために漸化式による値が真の値を大きくはずれるので bignum の使用が不可欠であった. しかしながら direct sampler はこのままで並列化可能なので並列計算によりこの欠点を緩和できる場合もあるものと思われる. また `double` 型を利用出来る場合かつ r が小さい場合はかなり高速化できる.

例 Risa/Asir のパッケージ `gtt_ds.rr` の timing data を参考までに記す. 実行は Intel Xeon CPU(2.70GHz), 256G memory の仕様の計算機でおこなった.

幾何/推測	A	B	C	合計
A	2	2	0	4
B	8	9	2	19
C	0	0	3	3
計	10	11	5	26

$$, \quad p = \begin{pmatrix} 1 & 9/10 & 11/10 \\ 1 & 13/10 & 99/100 \\ 1 & 1 & 1 \end{pmatrix}$$

$Au = \beta$: 行和, 列和が上の表で与えられたもので固定. このとき 100 個の乱数の生成時間は 81.5s + 48.1s(GC).
ソフトウエア `gtt_ds.rr`, `tk_ds_ahg.rr` の実行例.

---

[6] `https://www.math.upenn.edu/~wilf/AeqB.html`

[7] Risa/Asir に一般の場合の実装がパッケージ `tk_ds_ahg.rr` として提供されている.

[8] MCMC による方法については Diaconis-Sturmfels (1998) の論文
`https://projecteuclid.org/euclid.aos/1030563990`, その後の展開は教科書 "グレブナー道場" を参照

[9] n は Markov 基底のベクトルとしての長さ. T は thinning の間隔

[10] Risa/Asir での実装は `gtt_ds.rr`

幾何/推測	5	4	3	2	1	合計
5	2	1	1	0	0	4
4	8	3	3	0	0	14
3	0	2	1	1	1	5
2	0	0	0	1	1	2
1	0	0	0	0	1	1
計	10	6	5	2	3	26

図 1: 成績データ (グレブナー道場 p.221 より転載)

```
[1822] load("gtt_ds.rr");
[2720] gtt_ds.direct_sampler([[4,14,3],[10,6,5]],
                  [[1,9/10,11/10],[1,13/10,99/100],[1,1,1]]);
[ 0 1 3 ]
[ 8 5 1 ]
[ 2 0 1 ]
[2721] gtt_ds.direct_sampler([[4,14,3],[10,6,5]],[[1,9/10,11/10],[1,13/10,99/100],[1,1,1]]);
[ 3 1 0 ]
[ 6 4 4 ]
[ 1 1 1 ]
[2722] gtt_ds.direct_sampler([[4,14,3],[10,6,5]],[[1,9/10,11/10],[1,13/10,99/100],[1,1,1]]);
[ 2 1 1 ]
[ 6 4 4 ]
[ 2 1 0 ]
```

MCMC と direct sampler による p-値の計算の比較 [11)]

Direct sampler の利点である “パラメータの調整は不要である” という点をデモするため, ここでは direct sampler による 5×5 分割表に対する p 値の計算例をひとつ紹介する. 上で述べたようにこの場合 contuiguity を用いた direct sampler は計算速度が遅くなるため, ここでは分配関数 Z を Taylor 展開による近似計算で求める方法で direct sampler と MCMC を比較する.

図 1 は “グレブナー道場” に掲載されている, 幾何工学/推測工学の成績データに対する χ^2 検定統計量による p-値の計算である. 仮説を定める p は[12)]

$$\begin{pmatrix} 1.01 & 1.05 & 0.95 & 1.06 & 0.93 \\ 0.97 & 1.05 & 0.97 & 1.07 & 0.97 \\ 0.94 & 1.03 & 0.98 & 0.99 & 1.07 \\ 0.97 & 1.01 & 0.93 & 0.99 & 1.01 \\ 1.01 & 1.01 & 0.99 & 1.03 & 1.06 \end{pmatrix}$$

当てはめ値はこの p に対する期待値を用いている. 平代の direct sampler は間野の漸化式による方法ではなく $Z_A(\beta; p)$ の Taylor 展開を用いた近似計算によるものである[13)]. 上記 p に対しては 2 次の Taylor 展開での direct sampler (mano taylor 2) は実行時間はかかるものの, 図 2 のグラフで示されているように p 値の計算について良好な結果を得ている.

[11)] 平代達哉の修士論文からの転載

[12)] この p はテスト用に適当に決めた値であり, 意味はない.

[13)] p の成分がすべて 1 の時は壺のモデルによる sampler にほかならない.

MCMC		CPU 時間 (μs)
	burn-in:0 thinnng:無し	362,809
	burn-in:10000 thinnng:無し	378,440
	burn-in:0 thinnng:100	17,063,450
	burn-in:10000 thinnng:100	17,064,158
Direct sampler		
	Taylor 0 次	27,174,019
	Taylor 一次	289,105,633
	Taylor 二次	14,849,937,181

サンプル数は 99 万. CPU 時間[14] は 1,000,000 μs で 1 秒.

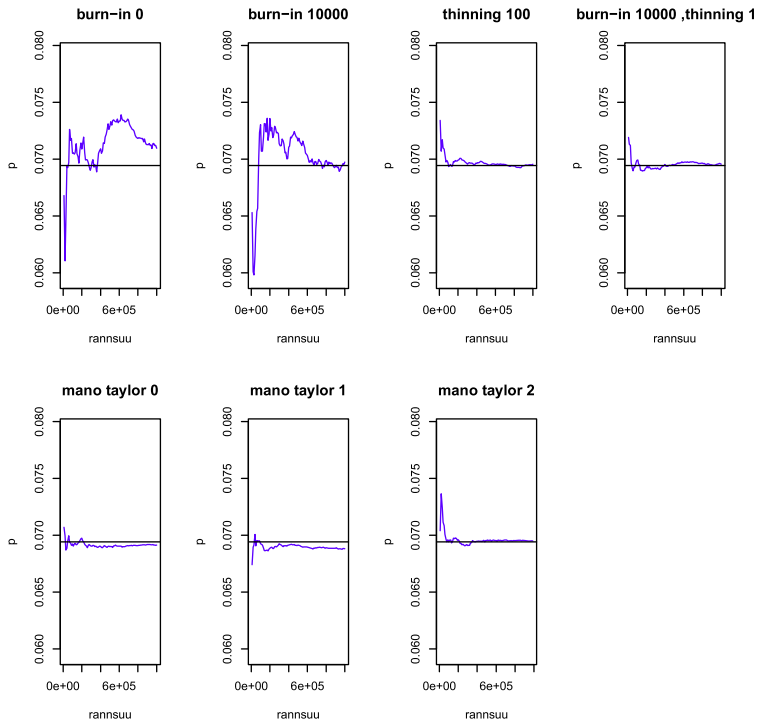

図 2: MCMC と direct sampler による p-値の計算 (平代達哉, 修論から転載). 右上の図は thinning 100 であるが “00” が欠けているので注意.

計算代数が有効に使えた例

“日比, グレブナー基底” (朝倉書店) ではグラフから定義される A できまる toric ideal が議論されている. ここでは図 3 のグラフからきまる[15]

[14] `clock()` で計測. Xeon E5-4650 CPU, 2.7 GHz; 256 GB of memory.

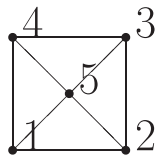

図 3: Graph for A

$$A^T = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix} \tag{4}$$

について, direct sampler を構成するための漸化式を p のすべての成分が 1 の時[16] に計算してみる. つまり, $Z_A(b;\mathbf{1})$ の b についての漸化式を求めることが問題である.

このような問題については数十年にわたり研究されてきた creative telescoping が有効であると思われる. ここでは `HolonomicFunctions.m` (Christopher Kouchan) [17] パッケージでこの漸化式を計算してみた. 一晩の計算で次のような答えを得た (b_i は β_i).

$$\begin{aligned}
& ((1+b_1)(1+2b_1)(1+b_1+b_2+b_3+b_4-b_5)(1+b_1-b_2+b_3-b_4+b_5))S_1 \\
+\ & (1+b_1+b_3)(1+2b_1+2b_3)(b_1-b_2+b_3-b_4-b_5), \\
& (1+b_2)(1+2b_2)(-1+b_1-b_2+b_3-b_4-b_5)(1+b_1+b_2+b_3+b_4-b_5)S_2 \\
+\ & (1+b_2+b_4)(1+2b_2+2b_4)(b_1-b_2+b_3-b_4+b_5), \\
& (1+b_3)(1+2b_3)(1+b_1+b_2+b_3+b_4-b_5)(1+b_1-b_2+b_3-b_4+b_5)S_3 \\
+\ & (1+b_1+b_3)(1+2b_1+2b_3)(b_1-b_2+b_3-b_4-b_5), \\
& (1+b_4)(1+2b_4)(-1+b_1-b_2+b_3-b_4-b_5)(1+b_1+b_2+b_3+b_4-b_5)S_4 \\
+\ & (1+b_2+b_4)(1+2b_2+2b_4)(b_1-b_2+b_3-b_4+b_5), \\
& (-1+b_1-b_2+b_3-b_4-b_5)(1+b_1-b_2+b_3-b_4+b_5)S_5 \\
-\ & (-b_1-b_2-b_3-b_4+b_5)
\end{aligned}$$

ここで $S_i f(b_i) = f(b_i+1)$ (b_i についての差分作用素).

この出力を得るための Mathematica への入力は以下のとおりである.

```
ann4 = Annihilator[(1/Factorial[u1])*(1/Factorial[u2])*(1/
     Factorial[u3])*(1/
     Factorial[-b1 - b2 - b3 + u1 + u2 + b4 + b5])*(1/
     Factorial[2*b3 - u2 - u3])*(1/Factorial[2*b2 - u1 - u2])*(1/
     Factorial[b1 - b2 - b3 + u2 + u3 - b4 + b5])*(1/
     Factorial[b1 + b2 + b3 - u1 - u2 - u3 + b4 - b5])*1, {S[b1],
   S[b2], S[b3], S[b4], S[b5], S[u1], S[u2], S[u3]}]
FindCreativeTelescoping[ann4, {S[u1] - 1, S[u2] - 1,
  S[u3] - 1}, {S[b1], S[b2], S[b3], S[b4], S[b5]}]
```

なるべく小さい分母多項式を見つける Heuristics (by C.Kouchan) が実装されていて効率的である.

まとめとアナロジー

[15] 図 3 の頂点 i, j が繋がってる時 i 列目と j 列目に 1 を書く.

[16] 独立性の検定に使う値

[17] `https://risc.jku.at/m/christoph-koutschan/`

統計で興味のある A ⇒ Direct sampler を作りたい ⇒ 漸化式を求める計算代数の問題

I_A のグレブナー基底	⇒	MCMC が作れる[18]
A-超幾何 の漸化式[19]	⇒	間野の direct sampler が作れる[20].

1. 素手で (理論的考察で) 漸化式を作れれば, random vector を生成するより効率的なアルゴリズムが作れる.
2. 計算代数の手法で $Z_A(\beta; p)$ の β についての漸化式が作れればより効率的な direct sampler が作れる.

参考

thinning や burn-in の調整は微妙な問題である. たとえばこの解説でのデータ例のように使う分割表の個数を固定して比較する場合や統計量の計算がそれなりの時間を必要とする場合は thinning した方が良いが, 推定量の分散を小さくするのが目的であれば thinning をしないですべての分割表を使ったほうがよいという議論もある[21].

たとえば, 次の分割表を考える.

	B_1	B_2	
A_1	53	19	72
A_2	56	31	87
	109	50	159

χ^2 統計量による検定において $u_{11} \leq 43$ か $56 \leq u_{11}$ のときオッズ比 1(属性 A と属性 B は独立) の帰無仮説は 5%水準で棄却される. 観察は $u_{11} = 53$ だから, 帰無仮説は棄却されない. 観察されたオッズ比が 1.544173... なので, 属性 A と属性 B は独立ではなく, オッズ比 3/2 という対立仮説をとってみる. 対立仮説の下で帰無仮説が正しく棄却される確率を検出力という, 厳密に計算すると検出力は $p_0 = 0.172863...$ である. MCMC による検出力の評価をデモした. 対立仮説の下での MCMC により 1,000 個の分割表を抽出し, そのうち $u_{11} \leq 43$ か $56 \leq u_{11}$ をとるものの割合を検出力の推定量とした. この試行を 10 回行い, i 回目の試行における検出力の推定値を p_i とする. 精度の基準として, RMSE (root mean square error)

$$\sqrt{\frac{1}{10}\sum_{i=1}^{10}(p_i - p_0)^2}$$

をとる.

burn-in	thinning	RMSE
1,000	-	0.055001
1,000,000	-	0.059027
1,000	1,000	0.013220

burn-in 1,000, 1,000,000 の結果がほぼ変わらないので, burn-in 1,000 で定常に達しているようである. しかし, burn-in 1,000,000 で十分に定常に達していても, thinning を行なわなければ誤差が大きい. これは 1,000 個の分割表の間の相関に起因する.

計算時間[22] は thinning の有無にほぼよらない. burn-in 1,000, thinning 1,000 では burn-in の後に 1,000,000 個の分割表を生成しているが, 1,000,000 個すべてを使った検出力の推定量の RMSE は 0.002384

[18] JST CREST 日比チーム編, グレブナー道場, 2011, 共立出版, を参照

[19] contiguity とも呼ぶ

[20] S.Mano, Partitions, Hypergeometric Systems, and Dirichlet Processes in Statistics, JSS Research Series in Statistics (2018), Springer

[21] W.A.Link, M.J.Eaton, On thinning of chains of MCMC, Methods in Ecology and Evolution 3 (2012), 112-115

[22] ランダムな分割表の生成の時間, その他の処理に時間がかかる場合は thinning が有効と思われる.

であり，thinning する場合の 0.013220 と較べて，相関がない場合に期待される $1/\sqrt{1,000} = 0.031623...$ 倍ほどではないが，1/5 よりも小さくなっている．thinning せず，すべての分割表を使う方が良いことが分かる．