コイン投げギャンブル必勝法
-符号理論による擬似乱数の非統計的検定-

原本博史, 松本 眞, 西村 拓士

2004 年 6 月 21 日 広島大学代数学セミナー

1　コイン投げギャンブル

胴元：コインを投げ続ける
賭け手：ある時点で 1 円払い賭ける

$$\text{その後の 8 回すべて表} \implies \text{賭け手は } 256(=2^8) \text{ 円貰う}$$
$$\text{それ以外の場合} \implies \text{返金は無し}$$

$$\begin{aligned}&(\text{儲かる金額の期待値})\\&= (\text{勝つ確率}) \times (\text{勝ったときの儲け金}) - 1\end{aligned}$$

儲かる金額の期待値は 0 円。

2　戦略

賭け手は過去 o 回のコイン投げの結果のうち、**裏の出た回数** $(=: T)$ を記憶できる (※表・裏の**パターン**は記憶できない)。
$\Longrightarrow T$ の値によって、未来の f 回のうち、何回裏が出るかを予測する。
$\Longrightarrow$ 賭けるか賭けないかを決定する。

3　疑似乱数とは

疑似乱数とは
計算機の中で、あたかもさいころを振って得られるかのようなでたらめな数列を、高速に再現性があるように生成する方法の総称
使用目的

- 確率的現象と関わる、あらゆる現象のシミュレーション（物理、金融、ゲーム)
- 情報の暗号化

ここでは32ビット整数(2進32桁)の疑似乱数を発生。

4　使った擬似乱数

`random()`: C 言語の現推奨擬似乱数 (1990–)
`ran_array()`:Knuth の新推奨 (1997–)
最下位 1 ビットをコイン投げシミュレーションに使う。

$$\text{偶数 (最下位 1 ビットが 0)} \leftrightarrow \text{表}$$
$$\text{奇数 (最下位 1 ビットが 1)} \leftrightarrow \text{裏}$$

5 `random()`

ラグ付きフィボナッチ生成法

$$\mathbf{x}_{i+31} = \mathbf{x}_{i+28} + \mathbf{x}_i \in \mathbb{Z}/2^{32} \quad (i = 1, 2, \ldots)$$

最下位 1 ビットは $\mathbb{F}_2$ 上の線形漸化式

$$x_{i+31} = x_{i+28} + x_i \in \mathbb{Z}/2 = \mathbb{F}_2 \quad (i = 1, 2, \ldots)$$

と見なせる。
$\mathbf{x}_1, \ldots, \mathbf{x}_{31} \in \mathbb{Z}/2^{32}$ には適当な初期値を入れる。
この漸化式で次々と 32 ビット整数を生成して、疑似乱数として使う。

6　random() によるシミュレーション

使う擬似乱数:random()

♠戦略
過去31回のコイン投げのうち、裏の出た回数をTとして

- $T \leq 14$のときは(次の８回は表と予想し)賭ける
- それ以外の場合は賭けない

- 上の戦略を$1,000,000$回とり、賭けた回数と勝った回数を記録
  (「31回のコイン投げ」を100万回観察することになる)
- $1,000,000$回を1セットとし、5セット実験する

回数	賭け数	勝数	勝率	儲け金	正規化偏差
1セット目	358921	2783	0.007754	353527	36.953
2セット目	359130	2587	0.007204	303142	31.677
3セット目	359180	2631	0.007325	314356	32.847
4セット目	359767	2714	0.007544	335017	34.977
5セット目	359836	2670	0.007420	323684	33.791

コインが真にランダムのときの勝率=$1/256 = 0.00390625$
正規化偏差の計算方法 $\cdots$ $M =$ (賭け数) とおくと

$$\text{勝数の正規化偏差} = \left(\text{勝数} - \frac{M}{256}\right) \Bigg/ \sqrt{\frac{1}{256} \cdot \frac{255}{256} \cdot M}$$

(参考：P(正規化偏差 $\geq 30) \approx 4 \times 10^{-198}$)

7　Lüscher の捨て改良

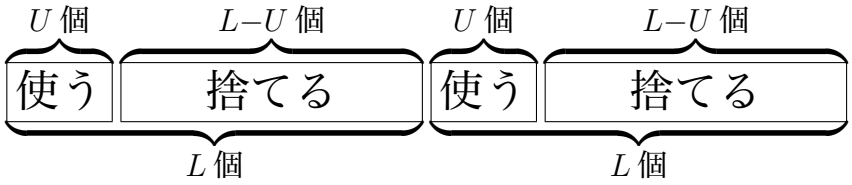
Lüscher(1994) による擬似乱数の改良法

1) L 個擬似乱数を生成 (L:Luxury Level)

2) 最初の U 個を使う

3) 残りの $L-U$ 個捨てる

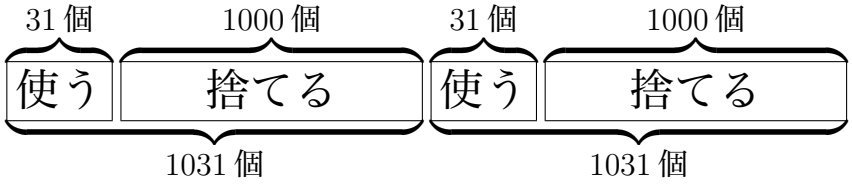
を繰り返す。



以下、「$(L-U)/L$ 捨て」と表記する。
物理シミュレーションではいくつかの標準的な手法の一つ。良い改良法と思われている。

8　改良 random() によるシミュレーション

使う擬似乱数:random(),1000/1031 捨て



♡戦略
過去31回の結果のうち、裏の出た数Tについて

- $T \leq 12$ または $T \geq 19$のときは賭ける
- それ以外の場合は賭けない

回数	賭け数	勝数	勝率	儲け金	正規化偏差
1セット目	283726	1215	0.004282	27314	3.2112
2セット目	283667	1253	0.004417	37101	4.3623
3セット目	282520	1246	0.004410	36456	4.2951
4セット目	277116	1133	0.004089	12932	1.5384
5セット目	278597	1239	0.004447	38587	4.5781

コインが真にランダムのときの勝率=$1/256 = 0.00390625$

(参考：P(正規化偏差 $\geq 4) \approx 0.00003$)

9 ran_array()

ran_array(): Knuth の新推奨
("The art of computer programming (3rd ed)" (1997–))
ラグ付きフィボナッチ生成法+Lüscher の捨て改良

$$\mathbf{x}_{i+100} = -\mathbf{x}_{i+63} + \mathbf{x}_i \in \mathbb{Z}/2^{30} \quad (i = 1, 2, \ldots)$$

最下位 1 ビットは $\mathbb{F}_2$ 上の線形漸化式

$$x_{i+100} = x_{i+63} + x_i \in \mathbb{F}_2 \quad (i = 1, 2, \ldots)$$

と見なせる。

9.1 ran_array() によるシミュレーション

使う擬似乱数:ran_array()

◇戦略
過去100回の結果のうち、裏の出た数Tについて

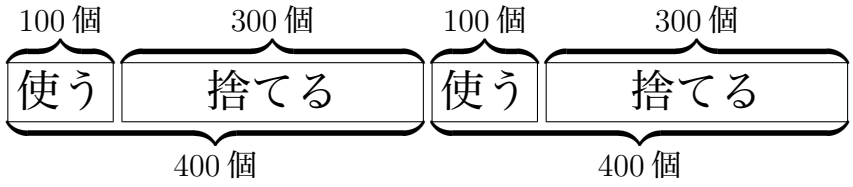
- $T \leq 44$ または $T \geq 56$ のときは賭ける
- それ以外の場合は賭けない

回数	賭け数	勝数	勝率	儲け金	正規化偏差
1セット目	271318	1175	0.004331	29482	3.5444
2セット目	271973	1234	0.004537	43931	5.2752
3セット目	271169	1202	0.004433	36543	4.3945
4セット目	271465	1255	0.004623	49815	5.9873
5セット目	270734	1263	0.004565	45682	5.4979

コインが真にランダムのときの勝率=$1/256 = 0.00390625$

9.2 改良 ran_array() によるシミュレーション

使う擬似乱数:ran_array(),300/400 捨て



♣戦略
過去 100 回の結果のうち、裏の出た数 T について

- $T \geq 51$ のときは賭ける
- それ以外の場合は賭けない

- 上の戦略を 50 億回とり、勝った回数を記録 (「100 回のコイン投げ」を 50 億回観察することになる)
- 50 億回を 1 セットとし、5 セット実験する

回数	賭け数	勝数	勝率	儲け金	正規化偏差
1セット目	2301344613	9001272	0.00391131	2981019	3.89138
2セット目	2301079453	9001071	0.00391167	3194723	4.17059
3セット目	2300966443	8994821	0.00390915	1707733	2.22943
4セット目	2301357635	8999022	0.00391031	2391997	3.12247
5セット目	2301061639	8997310	0.00391007	2249721	2.93694

コインが真にランダムのときの勝率=$1/256 = 0.00390625$
(参考:50億回実験するのに、ノートパソコンで約10分かかる)

- 各Tに関して、そこで賭けた場合の具体的な勝率を計算できる。
- 戦略を決めたとき、その戦略で儲かる儲け金の期待値を計算できる。

擬似乱数の生成法	戦略	儲け金の期待値
`random`	$T \leq 14$	0.944369
`random`+1000/1031捨て	$T \leq 12$ or $T \geq 19$	0.100746
`ran_array`	$T \leq 44$ or $T \geq 56$	0.134252
`ran_array`+300/400捨て	$T \geq 51$	0.001039

1円賭けたときに儲かる金額の期待値

10　擬似乱数の確率モデル

S : 状態空間 (初期値の空間)
$g : S \to S$: 状態遷移関数 (漸化式)
$b : S \to \{0, 1\}$: 出力関数

$\{0, 1\}$ に値をとる、長さ M の擬似乱数を発生させる擬似乱数発生器 G は、関数

$$O_G : S \longrightarrow \{0, 1\}^M \ ; \ s \mapsto (b(s), b(g(s)), \ldots, b(g^{M-1}(s)))$$

と見なせる。

仮定 1 各シミュレーションごとに、初期状態は状態空間 S から一様かつ独立に選ばれるものとする。

$\Longrightarrow$ この仮定のもとでは、擬似乱数発生器の出力は $\{0,1\}^M$ に値を取る確率変数になる。

以下、1 回のシミュレーションとは、 $o(\text{observed})$ 回の出力をみて、次の $f(\text{future})$ 回の出力を予想し、その結果を見ることとする。このとき

$$M = o + f$$

となる。

$P_G(s/o;t/f) :=$

擬似乱数発生器Gを用いて、**仮定**1の下で

o回中s回裏の出たという条件下で

f回中t回裏の出る確率

理想的な擬似乱数発生器では

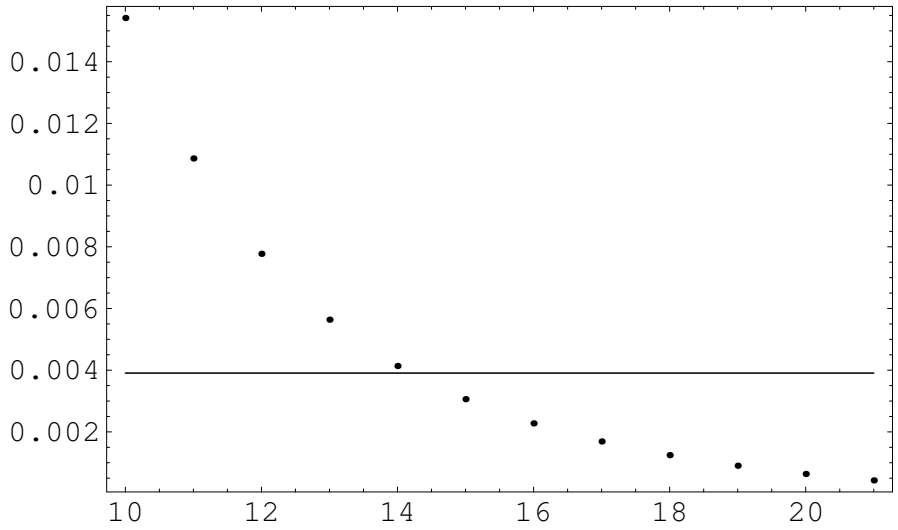
$$P_G(s/o;t/f) = \left(\begin{array}{c} f \\ t \end{array} \right) /2^f$$

となる。

- 一般に$P_G(s/o;t/f)$を計算するのは困難。
- 先の疑似乱数については計算可能(符号理論のMacWilliams恒等式を使う)

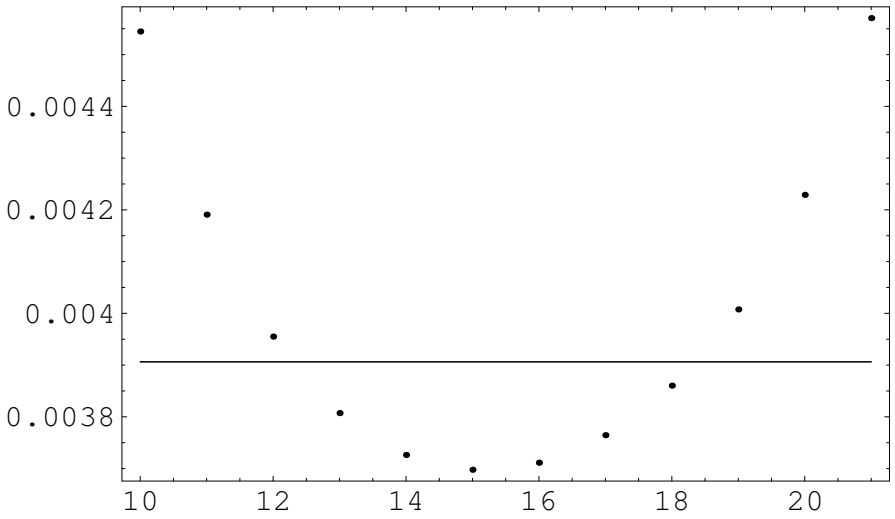
11　確率計算の結果

$G = \texttt{random}()$ の場合の $P_G(s/31; 0/8)$ のグラフ
$(10 \leq s \leq 21)$



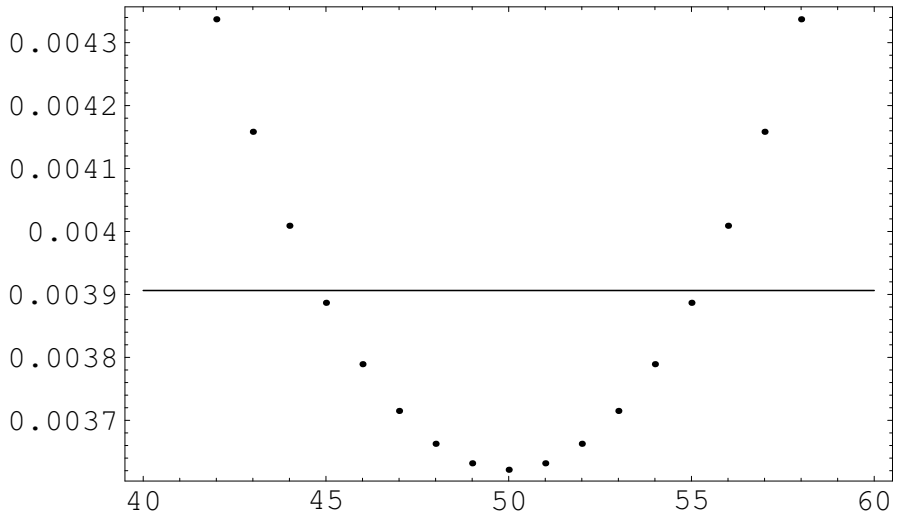
横線:$1/256$=0.00390625
$\Rightarrow T \leq 14$ のときは賭ける。
1円を賭けたときの儲け金の期待値=0.944369
「戦略をきめる」とは $\{0, \ldots, 31\}$ の部分集合 B を決めること (この場合 $B = \{s \mid 0 \leq s \leq 14\}$)

$G = \texttt{random}(), 1000/1031$捨ての場合の$P_G(s/31; 0/8)$のグラフ$(10 \leq s \leq 21)$



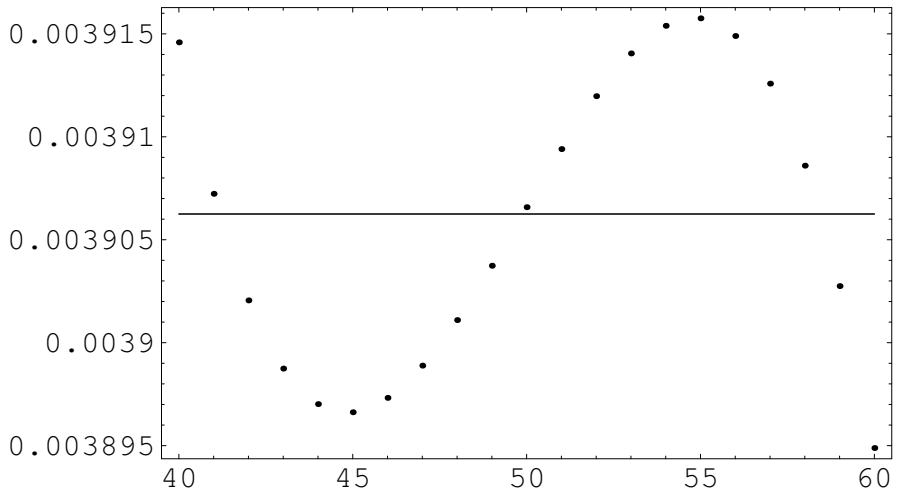
横線:1/256=0.00390625
$\Rightarrow T \leq 14$または$T \geq 19$のときは賭ける。
1円を賭けたときの儲け金の期待値=0.100746

$G = \texttt{ran\_array}()$ の場合の $P_G(s/100; 0/8)$ のグラフ
$(40 \leq s \leq 60)$



横線:1/256=0.00390625
$\Rightarrow T \leq 44$ または $T \geq 56$ のときは賭ける。
1円を賭けたときの儲け金の期待値=0.134252

$G = \texttt{ran_array}(), 300/400$捨ての場合の $P_G(s/100; 0/8)$ のグラフ $(40 \leq s \leq 60)$



横線:1/256=0.00390625
$\Rightarrow T \geq 51$ のときは賭ける。
1円を賭けたときの儲け金の期待値=0.001039

12　確率計算法

$\mathbf{x} = (x_1, \ldots, x_o, x_{o+1}, \ldots, x_{o+f}) \in \{0,1\}^M (M = o + f)$ に対して分離Hamming重みを以下のように定義する。

$$\begin{aligned} \mathrm{wt}_o(\mathbf{x}) &:= x_1, \ldots, x_o\text{の中の}1\text{の個数} \\ \mathrm{wt}_f(\mathbf{x}) &:= x_{o+1}, \ldots, x_{o+f}\text{の中の}1\text{の個数} \end{aligned}$$

各$0 \leq s \leq o, 0 \leq t \leq f$に対して

$$N_{s,t}(G) := \#\{r \in S \mid \mathrm{wt}_o(O_G(r)) = s, \mathrm{wt}_f(O_G(r)) = t\}$$

と置くと、先の条件付き確率は

$$P_G(s/o; t/f) = \frac{N_{s,t}(G)}{\sum_{j=0}^{f} N_{s,j}(G)}$$

と表せる。

$WP_G(B)$: 戦略 $B \subset \{0, \ldots, o\}$ に従って賭けをして勝つ確率
とすると

$$\sum_{s \in B} N_{s,0}(G) \Big/ \sum_{s \in B, 0 \leq j \leq f} N_{s,j}(G)$$

$E_G(B)$: 戦略 B の下、1 回の賭けで儲かる金額の期待値
とすると

$$2^f \cdot WP_G(B) - 1$$

13　分離重み数え上げ (符号理論)

- $\{0,1\}$ を $\mathbb{F}_2$ と同一視
- S は $\mathbb{F}_2$ 線形空間
- $O_G : S \to \mathbb{F}_2^M$ は線形写像

と仮定する
例:`random()` について、その最下位 1 ビットは線形漸化式

$$x_{i+31} = x_{i+28} + x_i \in \mathbb{F}_2$$

で生成されていて、写像

$$O_G : (x_1, \ldots, x_{31}) \mapsto (x_1, \ldots, x_{39})$$

は $G_G : S = \mathbb{F}_2^{31} \to \mathbb{F}_2^{31+8}$ なる線形写像。

部分集合$C \subset \{0,1\}^M$、$0 \leq s \leq o, 0 \leq t \leq f$に対して

$$A_{s,t}(C) := \#\{\mathbf{x} \in C \mid \mathrm{wt}_o(\mathbf{x}) = s \text{ and } \mathrm{wt}_f(\mathbf{x}) = t\}$$

をCの分離重み数え上げと言う。

以下$C := O_G(S)$とおく。
$N_{s,t}(G) = \#(\mathrm{Ker}(O_G)) \cdot A_{s,t}(O_G(S))$だから、

$$P_G(s/o; t/f) = \frac{A_{s,t}(C)}{\sum_{j=0}^{f} A_{s,j}(C)}$$

である。

$A_{s,t}(C)$ の計算は、一般には $\dim C$ に対して NP 完全である (Vardy 1997)。
しかし、$\dim C^{\perp}$ が小さい場合、MacWilliams 恒等式と呼ばれる恒等式を用いて $A_{s,t}(C)$ を計算することが出来る。

14 MacWilliams恒等式

$C \subset \mathbb{F}_2^M$:線形部分空間

定義 1 Cの分離重み数え上げ多項式W_Cとは

$$W_C(x, y, X, Y) := \sum_{0 \leq i \leq o, 0 \leq j \leq f} A_{i,j}(C) x^{o-i} y^i X^{f-j} Y^j$$

のこと$(M = o + f)$。

$C^\perp$の分離重み数え上げが求まれば、Cの分離重み数え上げが求まる。
$\mathbb{F}_2^M$には内積

$$\langle (x_1, \ldots, x_M), (y_1, \ldots, y_M) \rangle := \sum_{i=1}^{M} x_i y_i (\in \mathbb{F}_2)$$

が定まっている。このときCの直交空間を

$$C^\perp := \{\mathbf{y} \in \mathbb{F}_2^M | \langle \mathbf{x}, \mathbf{y} \rangle = 0 \text{ for all } \mathbf{x} \in C\}$$

で定める。

定理 1 (分離 MacWilliams 恒等式)

$$W_C(x, y, X, Y) = \frac{1}{\#C^{\perp}} W_{C^{\perp}}(x + y, x - y, X + Y, X - Y)$$

$\dim C^{\perp}$ が小さければ右辺は (総当たりで) 計算可能。
$\Rightarrow$ 線形部分空間 C での数え上げが可能。

実験では

- $C^{\perp}$ の分離重み数え上げの計算にはC言語
- MacWilliams恒等式の計算にはMathematica

を使った。
今回のシミュレーションでは$\dim C^{\perp} = 8$となるパラメータを選んだ。

$$O_G : S = \mathbb{F}_2^o \to \mathbb{F}_2^{o+f},\ o = \dim C = o,\ f = 8,\ O_G\text{は単射}$$

2^{100} の元を持つ線形部分空間の分離重みの数え上げが可能 (総当たりでは計算不可能)。

15　捨て改良の場合の計算法

Lüscher の捨て改良 $\cdots$ $\mathbb{F}_2$ 線形ではないが、区分的に線形

定義 2 線形な擬似乱数発生器 G とは

- S を状態空間 ($\mathbb{F}_2$線形)
- $g : S \to S$ を状態遷移関数 ($\mathbb{F}_2$線形)
- $b : S \to \{0, 1\}$ を出力関数 ($\mathbb{F}_2$線形)

で初期値 $s \in S$ に対して状態を $(s, g(s), g^2(s), \ldots)$ と遷移させ、$(b(s), b(g(s)), b(g^2(s)), \ldots)$ を出力する。

$G + (L - U/L$ 捨て$)$ による擬似乱数発生器を G' とすると

- $S' = S \times \mathbb{Z}/U$ を状態空間
- $g' : (s, i) \mapsto \begin{cases} (g(s), i+1) & \text{if } i < U-1, \\ (g^{L-U+1}(s), 0) & \text{if } i = U-1, \end{cases}$ を状態遷移関数
- $b' : (s, i) \mapsto b(s)$ を出力関数

G' の状態遷移は

$$(s,0) \mapsto (g(s),1) \mapsto (g^2(s),2) \mapsto \cdots$$
$$\mapsto \underbrace{(g^{U-1}(s),U-1) \mapsto (g^L(s),0)}_{L-U \text{ 個捨て}} \mapsto (g^{L+1}(s),1) \mapsto \cdots$$

G' の出力は

$$(b(s), b(g(s)), b(g^2(s)), \ldots, b(g^{U-1}(s)), b(g^L(s)), b(g^{L+1}(s)), \ldots)$$

擬似乱数発生器G'は

$$O_{G'} : S \times \mathbb{Z}/U \to \mathbb{F}_2^M$$

について、各$i \in \mathbb{Z}/U$を止めるごとに

$$O_{G,i}(s) := O_{G'}(s, i)$$

が$\mathbb{F}_2$線形写像になる。
$\Longrightarrow (O_{G,i}(S))^{\perp}$が大きくなければ$A_{s,t}(O_{G,i}(S))$は計算可能で

$$N_{s,t}(G') := \sum_{i=0}^{U-1} \#(\ker(O_{G,i}))A_{s,t}(O_{G,i}(S))$$

$\Longrightarrow P_{G'}(s/o; t/f)$が計算可能

16　種々の生成法の比較

各種擬似乱数生成法を

- 最適戦略下での儲け金の期待値
- 10,000,000 個の擬似乱数の生成時間

で比較する。
ここで最適戦略とは

$$B = \{s \mid 0 \leq s \leq o, P_G(s/o, 0/f) > 2^{-f}\}$$

のこと。

比較した擬似乱数の生成法

- `ran_array()`+1000/1100捨て
- 13項線形漸化式
  x_{i+100} = ある12個のx_jたちの一次結合 $(i \leq j \leq i+99)$
- 15項線形漸化式
  x_{i+100} = ある14個のx_jたちの一次結合 $(i \leq j \leq i+99)$
- Mersenne Twister 89
- Mersenne Twister 127

Mersenne Twister とは

$$\mathbf{x}_{i+n} := \mathbf{x}_{i+m} \oplus \mathbf{x}_{i+1}B \oplus \mathbf{x}_i A \in \mathbb{F}_2^{32} \quad (i = 1, 2, \ldots)$$

で生成される擬似乱数。ここで A, B はある 32×32 行列。
また状態空間は MT89 : 89 ビット, MT127 : 127 ビット。

擬似乱数の生成法	最小重み (重複度)	儲け金の期待値	生成時間 (sec.)
`ran_array()`+1000/1100 捨て	12(16)	1.422×10^{-8}	1.320
13 項線形漸化式	13(8)	1.764×10^{-7}	0.742
15 項線形漸化式	15(8)	1.921×10^{-8}	0.844
MT89	35(1)	1.280×10^{-12}	0.801
MT127	34(6)	3.029×10^{-15}	0.795

(参考：MT19937 を使って 10,000,000 個の疑似乱数の生成するのに必要な時間は 0.72(sec.))