選挙で起こりうる

コンドルセ勝者が決まらない確率

はじめに

ここで紹介する話題“ コンドルセ勝者が決まらない確率 ”は、愛知教育大学 初等教育教員養成課程 数学選修 鳥井美希さんの平成 22 年度卒業論文と愛知教育大学 中等教育教員養成課程 数学専攻 中村貴亮君の平成 22 年卒業論文から抜粋したものである。この話題における確率の値やパソコン上での計算を面白く思い、編集したものである。

まず、卒業論文から序文“ はじめに ”を紹介しておこう。

『今回私は、選挙で起こりうる確率について研究することにした。私は、もともと生活と結びついている確率に興味を持っていた。私たちの気がつかないところでも、起こっている確率はたくさんある。なぜ、私が選挙の確率に注目したかというと「科学雑誌 newton」に松原望さんの書いた投票のパラドックスの記事があり、この内容に興味を持ったからだ。この内容には、「当選者が実は最下位者である」「当選者決定不能の確率」などのことが記載されていた。私は、これらのことをより詳しく調べ、大学で学んだプログラミングの知識を活用し、プログラムを作成して数値を出していく研究を行った。　今回の研究を通して、数学はわたしたちの生活に密接していることが実感できた。今後も、生活に密接する数学を調べていき、生徒がより興味を持てるような授業ができるように努めていきたい。(鳥井美希)』

愛知教育大学 中等教育教員養成課程 数学専攻 中村貴亮君は C++言語による「OpenMP による並列処理の整数問題への応用」という表題の卒業論文を作成した。その中で、鳥井美希さんの探求に協力して、“ コンドルセ勝者が決まる確率 ”についての計算実験を行いコンドルセ勝者が決まる確率について説明を加えているので、それらを含めて抜粋して紹介する。

‘ コンドルセのパラドックス ’に潜む数学の論文を探し出し、その内容を紹介し、コンピュータ上での計算を通して我々の興味を引き出した両君に感謝する。

浦田敏夫　　2011 年 4 月 8 日

最近、プログラム CondorcetWinner.cpp に関連して、順列を準備する手続きについては通常の手続きのほうが適切ではないかと思わせられたので、それに基づいた案を付録に置いた。

2013 年 10 月 21 日

目 次

第1章　投票のパラドックスとコンドルセ勝者

1.1　投票のパラドックス

我々が物事を決めるとき、大抵の場合、多数決を用いる。つまり、候補の中で最もよいと思うものに各々が票を投じ、その票が過半数となるものを総意とする。しかし、最良の選択ができないのではないかというのが投票のパラドックスである。本論文では、その中でも、コンドルセ勝者 (Condorcet Winner) が選出されない (または選出される) 確率について調べる。(William V. Gehrlein が CONDORCET'S PARADOX という論文を書いていることを、知って、多くのことを学ぶことができた。)

多数決選挙について。私たちの生活で一番身近にあるのが多数決である。ここではつぎのような手続きを考える。各投票者が n 人の候補者 $\mathrm{C}_1, \mathrm{C}_2, \cdots, \mathrm{C}_n$ を線形順位づけしたリストを作る；線形順位づけの意味は、このリストにおいて‘ どの候補も 1 番から n 番までの優先順位のどこかにその順位を与えられている ’ということである。(例えば、候補者 $\mathrm{C}_1, \mathrm{C}_2, \cdots, \mathrm{C}_n$ の優先順位付け 1 番 C_{i_1}、2 番 C_{i_2}、3 番 C_{i_3}、$\cdots$ 、n 番 C_{i_n} を

$$\mathrm{C}_{i_1} > \mathrm{C}_{i_2} > \cdots > \mathrm{C}_{i_n}$$

と表す。) すべての投票者による n 人の候補者への線形順位づけリストを表にして、各候補について 1 番であると選んだ人の人数を得票 (数) とする。

(単純) 過半数多数決 Majority Voting： 過半数の票を得た候補者は当選者である。
もし候補者が二人 (二者択一) で投者数が奇数であれば、単純多数決の場合は必ず当選者が決まる。

相対多数決 Plurality Voting： 最高得表数を得た候補がただ一人いるときは、この最高得票者を当選者とする。

候補者 3 人の場合の例 1.

候補者 3 人に対して、投票者 n 人が選挙をする状況を考える。投票者の数 n を、ここでは簡単のため 7 人とする。候補者を C_1、C_2、C_3、投票者を V_1、V_2、$\cdots$、V_7 とする。投票者は、候補者 3 人に対して、1 位から 3 位まで順位をつける。

	C_1	C_2	C_3
V_1	1	2	3
V_2	3	1	2
V_3	1	3	2
V_4	3	1	2
V_5	3	2	1
V_6	1	3	2
V_7	3	2	1

その結果が右の表のようになったとする。
過半数多数決 (Majority Voting) では過半数当選者はいないが、相対多数決 (Plurality Voting) では C_1 が当選する。

しかし、以下の方法ではどうだろうか。まず、C_1 と C_2 を見ていく。C_1 が C_2 より、よい順位をもらったのは、3 人。一方、 C_2 が C_1 より、よい順位をもらったのは、4 人。よって、C_1 と C_2

だけに着目する (下左表) と、$C_1 < C_2$ となり、C_2 の方が勝ちとなる。

	C_1	C_2
V_1	1	2
V_2	3	1
V_3	1	3
V_4	3	1
V_5	3	2
V_6	1	3
V_7	3	2

	C_2	C_3
V_1	2	3
V_2	1	2
V_3	3	2
V_4	1	2
V_5	2	1
V_6	3	2
V_7	2	1

同様に見ていくと、C_2 と C_3 の間 (上右表) では、$C_2 < C_3$ となり、C_3 の勝ち。C_3 と C_1 では、$C_1 < C_3$ で C_3 の勝ちとなっている。総当たりの結果は以下の表だ。

＼	C_1	C_2	C_3
C_1	＼	×	×
C_2	○	＼	×
C_3	○	○	＼

すると、$C_1 < C_2 < C_3$ となる (この様な候補 C_3 はコンドルセ勝者と呼ばれる)。相対多数決での当選者 C_1 がコンドルセ勝者 C_3 より適切な当選者であるということには疑問が生じると思われる (コンドルセ パラドックス)。

1.2 コンドルセ勝者 (Condorcet's Winner) とは

当選者を選ぶにあたっての選び方はいろいろある。私たちの普段の生活で、代表を決めることはよくある。今回の研究における選挙方法は、実際の日本の選挙では行われていないが、コンドルセ選挙 (手続き) と呼ばれる (選挙) 方法である。

コンドルセ選挙
投票者が、候補者を (当選の順位に対して) 線形順位づけした表を作る。この表をもとに、候補者を二人ずつごとに二者択一多数決して勝者を決める。たとえば、候補者 L、M、N、… 等に対して、まず L 対 M、次に M 対 N、そして N 対 L、… 候補者のすべての組あわせについて多数決をしていきそれぞれで勝者を決めていく。

コンドルセ勝者 (当選者) とは，二者択一多数決を通して残りのすべての候補に勝つことのできる候補のことである.

候補者 3 人の場合の例 2.

候補者 3 人に対して、投票者 n 人が選挙をする状況を考える。投票者の数は、直接関係しないので、ここでは簡単のため 7 人とする。候補者を C_1、C_2、C_3、投票者を V_1、V_2、…、V_7 とする。投票者は、候補者 3 人に対して、1 位から 3 位まで順位をつける。

その結果が以下の表のようになったとする。

	C_1	C_2	C_3
V_1	1	2	3
V_2	2	1	3
V_3	1	3	2
V_4	3	2	1
V_5	3	2	1
V_6	1	3	2
V_7	3	1	2

過半数多数決 (Majority Voting) では過半数当選者はいない。相対多数決 (Plurality Voting) では C_1 が当選する。しかし、以下の方法ではどうだろうか。

まず、C_1 と C_2 を見ていく。C_1 が C_2 より、よい順位をもらったのは、3 人。一方、C_2 が C_1 より、よい順位をもらったのは、4 人。よって、C_1 と C_2 だけに着目すると、$C_1 < C_2$ となり、C_2 の方が勝ちとなる。

	C_1	C_2
V_1	(1)	2
V_2	2	(1)
V_3	(1)	3
V_4	3	(2)
V_5	3	(2)
V_6	(1)	3
V_7	3	(1)

	C_2	C_3
V_1	(2)	3
V_2	(1)	3
V_3	3	(2)
V_4	2	(1)
V_5	2	(1)
V_6	3	(2)
V_7	(1)	2

	C_1	C_3
V_1	(1)	3
V_2	(2)	3
V_3	(1)	2
V_4	3	(1)
V_5	3	(1)
V_6	(1)	2
V_7	3	(2)

同様に見ていくと、C_2 と C_3 の間では、$C_2 < C_3$ となり、C_3 の勝ち。C_3 と C_1 では、$C_3 < C_1$ で C_1 の勝ちとなっている。　総当たりの結果は以下の表だ。

＼	C1	C2	C3
C1	＼	×	○
C2	○	＼	×
C3	×	○	＼

すると、$C_1 < C_2 < C_3 < C_1 < C_2 < C_3 < C_1 \cdots$ となり、勝者を決めることができない。コンドルセ勝者が存在しない場合には、‘ コンドルセ勝者が決められない ’という。

1.3　コンドルセ選挙の二者択一多数決における三すくみ

候補者 3 人の場合の例 2 のように、選挙を行った結果、票が分かれてしまい当選者が決まらないことがある。たとえば、(コンドルセ選挙の) 二者択一による選挙で、候補者 L、M、N の間で L 対 M で L が勝ち、次に M 対 N で M が勝ち、そして N 対 L で N が勝つとする。このとき、当選者を決めることができない。このような状態は、投票者が合理的な判断をしているのに全体としての (不合理な結果) 決定不能が起こるという－投票のパラドックスのひとつである。このような特別な結果として当選者が決まらないことを三すくみ（決定不能）という。

この研究で扱う‘ 投票のパラドックス ’は、選挙で多数決というシステムで選ばれた当選者は最良と思われる人が選ばれたはずであるのに、実は最悪と思われる人が選らばれていたり、候補者が同点となり当選者を決められないということが起こったりする現象を言い表わしているものである－数学または論理的意味でのパラドックスの存在を主張しているのではない。この投票のパラドックスに関連して 、二者択一多数決を利用する選挙での“ 投票の順番を操作することで当選者を操作できる不公正はアジェンダパラドックス (Agenda Paradox) などと呼ばれることがある。これらの確率についても調べ、十進 BASIC でプログラムを作り確率の実験を試みたが、省略する。

コンドルセ選挙における二者択一多数決を通して勝数が最多の候補がただ一人いるときは、この候補を (ここでは) 準当選者と呼ぶことにする。

命題 候補者が 3 名または 4 名の場合は、コンドルセ選挙の準当選者はコンドルセ勝者である。

証明.

コンドルセ選挙にコンドルセ勝者ではない準当選者がいるとする。候補者が 3 名の場合、準当選者の定義から、この準当選者の勝ち数は 1 以下で他の候補の勝ち数は 0 でなければならない。二者択一多数決の組み合わせは3 通りであるから、勝ち数も 3 なければならないから、不合理である。候補者が 4 名の場合、準当選者の定義から、この準当選者の勝ち数は 2 以下で他の候補の勝ち数は 1 以下でなければならない。この場合、勝ち数の和は 2+1+1+1 = 5 以下となるが、二者択一多数決の組み合わせは $4 \times 3 \div 2 = 6$ 通りであるから、勝ち数も 6 なければならないから、不合理である。

1.4 コンドルセ勝者が決まらない確率

William V.Gehrlein (参考文献 [5]) はコンドルセ勝者についての研究を行っていた。この研究をもとに、コンドルセ勝者が決まらない確率を調べてみる。

今、$P(m,n)$ を m 人の候補者と n 人の投票者の場合にコンドルセ勝者が決まらない確率とする。この確率の計算について考えよう。そのためにそれぞれの投票行動の確率について調べる。投票行動の確率は以下のようにして求められる。

二つの前提

1. 投票者は真に独立である
2. 投票者は m 人の候補者 $\mathrm{C}_1, \mathrm{C}_2, \cdots, \mathrm{C}_m$ に対して（線形）優先順位

$$\mathrm{O}_i \ : \ \mathrm{C}_{i_1} > \mathrm{C}_{i_2} > \cdots > \mathrm{C}_{i_m}$$

をつける。このとき、各投票者の順位づけの可能性は $K = m!$ 通りである。それらの順位づけを

$$\mathrm{O}_1 \ : \ \mathrm{C}_1 > \mathrm{C}_2 > \cdots > \mathrm{C}_m$$

$\cdots$

$$\mathrm{O}_K \ : \ \mathrm{C}_m > \mathrm{C}_{m-1} > \cdots > \mathrm{C}_1$$

とし、投票者が順位づけ O_i を選ぶ確率は等しく $\dfrac{1}{K}$ とする。

例として、$m=3$ の場合について考えると、投票者の順位づけの可能性は、以下の $6(=3!=6)$ 通りである。

$$
\begin{aligned}
&\mathrm{O}_1 \ : \ \mathrm{C}_1 > \mathrm{C}_2 > \mathrm{C}_3 \\
&\mathrm{O}_2 \ : \ \mathrm{C}_1 > \mathrm{C}_3 > \mathrm{C}_2 \\
&\mathrm{O}_3 \ : \ \mathrm{C}_2 > \mathrm{C}_3 > \mathrm{C}_1 \\
&\mathrm{O}_4 \ : \ \mathrm{C}_2 > \mathrm{C}_1 > \mathrm{C}_3 \\
&\mathrm{O}_5 \ : \ \mathrm{C}_3 > \mathrm{C}_1 > \mathrm{C}_2 \\
&\mathrm{O}_6 \ : \ \mathrm{C}_3 > \mathrm{C}_2 > \mathrm{C}_1 .
\end{aligned}
$$

具体的に、考える：

1. m 人の候補者に n 人が投票を行う。
2. 各投票者の投票行動の可能性は $K=m!$ 通りで、投票行動 O_i に対する投票結果 (投票者数) は n_i 人であるとする（$n_1+n_2+\cdots+n_K=n, 0\le n_i\le n$)。
3. n 人の投票において順位付け O_i を選んだ投票者の数に対する確率変数を X_i とする。

このとき

$$
\begin{aligned}
P(x_i=n_i, i=1,2,3\cdots,K) &= \frac{n!}{n_1!n_2!\cdots n_K!}\prod_{i=1}^{K}\Big(\frac{1}{K}\Big)^{n_i} \\
&= \frac{n!}{n_1!n_2!\cdots n_K!}\Big(\frac{1}{K}\Big)^{n} \qquad \cdots \quad (*)
\end{aligned}
$$

と計算される。

説明.

投票者が順位づけ O_i を選ぶ確率は $\dfrac{1}{K}$ であるから、投票行動の確率は以下のようにあらわせる。

$$
A_{n_1n_2\cdots n_K} \ : \ n \text{ 人を } n_1\text{人}, n_2\text{人}, \cdots, n_K\text{人にわける場合の組み合わせの数}
$$

と置くと、(K 通りの) 全ての順位づけの n 人の投票者の投票による組み合わせの総数 K^n に対し

$$
\sum_{n_1+n_2+\cdots+n_K=n} A_{n_1n_2\cdots n_K} \ = \ K^n
$$

が成り立つ。したがって

$$
\begin{aligned}
P(x_i=n_i, i=1,2,3\cdots,K) &= A_{n_1n_2\cdots n_K}\times\Big(\frac{1}{K}\Big)^{n_1}\times\Big(\frac{1}{K}\Big)^{n_2}\times\cdots\times\Big(\frac{1}{K}\Big)^{n_K} \\
&= A_{n_1n_2\cdots n_K}\times\Big(\frac{1}{K}\Big)^{n_1+n_2+\cdots+n_K} \\
&= A_{n_1n_2\cdots n_K}\times\Big(\frac{1}{K}\Big)^{n}.
\end{aligned}
$$

ここで

$$
\begin{aligned}
A_{n_1n_2\cdots n_K} &= \frac{n!}{n_1!(n-n_1)!}\times\frac{(n-n_1)!}{n_2!(n-n_1-n_2)!}\times\cdots\times\frac{(n-n_1-n_2-\cdots-n_{K-1})!}{n_K!0!} \\
&= \frac{n!}{n_1!n_2!\cdots n_K!}
\end{aligned}
$$

よって

$$P(x_i = n_i, i = 1, 2, 3\cdots, K) \quad = \ A_{n_1 n_2 \cdots n_K} \times \left(\frac{1}{K}\right)^n \ = \ \frac{n!}{n_1! n_2! \cdots n_K!} \times \left(\frac{1}{K}\right)^n.$$

参考.

$P(x_i = n_i, i = 1, 2, 3\cdots, K)$ の求め方は、下記の確率の問題の解の計算法の拡張された考え方である。

さいころの確率の問題：さいころを 3 回振って 1 回 3 がでて、2 回 5 が出る確率を求めよ。
1 から 6 の目が出る確率は $\frac{1}{6}$ である。また、3 と 5 が出る組み合わせは ${}_3C_2 = 3$ 通り。よって、確率は、$3 \times \left(\frac{1}{6}\right)^3 = \frac{1}{72}$ である。

さて P(3,3) を求めよう。

1. 3 人の投票者が同じ順位づけをしたとき、コンドルセ勝者は決まる。この順位づけの場合の数は ${}_6C_1 = 6$ 通りある。

$$\sum_{n_i=3} A_{n_1 n_2 \cdots n_6} = {}_6C_1 \times A_{30\cdots0} = 6 \times A_{30\cdots0}.$$

2. 2 人が同じ順位づけをしたときの場合の数は ${}_6C_2 \times 2 = 6 \times 5 = 30$ 通りある。このときも、コンドルセ勝者は決まる。

$$\sum_{n_i=2, n_j=1} A_{n_1 n_2 \cdots n_6} = ({}_6C_2 \times 2) \times A_{210\cdots0} = 30 \times A_{210\cdots0}.$$

3. 3 人がそれぞれ違う順位づけをしたときの場合の数は、${}_6C_3 = 6 \times 5 \times 4 \div 6 = 20$ 通りである。

$$\sum_{n_h=1, n_i=1, n_j=1} A_{n_1 n_2 \cdots n_6} = ({}_6C_3) \times A_{1110\cdots0} = 20 \times A_{1110\cdots0}.$$

しかしこのとき、$\{\mathrm{O}_1, \mathrm{O}_3, \mathrm{O}_5\}$ あるいは $\{\mathrm{O}_2, \mathrm{O}_4, \mathrm{O}_6\}$ の 2 通りの順位づけはコンドルセ勝者が決まらない (3 すくみ) である。

$(*)$ より各投票行動の確率は、それぞれ

1. の場合 $\quad 6 \times \frac{3!}{3!0!0!} \times \left(\frac{1}{6}\right)^3 \ = \ \left(\frac{1}{6}\right)^2$

2. の場合 $\quad 30 \times \frac{3!}{2!1!0!} \times \left(\frac{1}{6}\right)^3 \ = \ 15 \times \left(\frac{1}{6}\right)^2$

3. の場合 $\quad 20 \times \frac{3!}{1!1!1!} \times \left(\frac{1}{6}\right)^3 \ = \ 20 \times \left(\frac{1}{6}\right)^2$

上の三つの場合の確率の和は 1 となり、すべての投票行動の確率が挙げられている。この中でコンドルセ勝者が決まらない確率は、$A_{103050} + A_{020406} = 2 \times A_{1110\cdots0}$ より

$$\begin{aligned} 2 \times \frac{3!}{1!1!1!} \times \left(\frac{1}{6}\right)^3 \ &= \ 2 \times \left(\frac{1}{6}\right)^2 \\ &= \ \frac{1}{18} = 0.0555\cdots \end{aligned}$$

である。こうして、$P(3,3)=\dfrac{1}{18}=0.055555\cdots$ であることが示された。さらに、$P(3,n)$ と $P(4,n)$ の関係を与える May の定理が知られている (参考文献 [4], [5])。

† May の定理 によれば、$\overline{P}(m,n)$ をコンドルセ勝者が決まる確率とするとき $\overline{P}(4,n)=2\overline{P}(3,n)-1$ が成り立つことが示されている。よって

$$\begin{aligned} 1-\overline{P}(4,n) &= 1-\big(2\overline{P}(3,n)-1\big) \\ &= 2\big(1-\overline{P}(3,n)\big). \end{aligned}$$

すなわち、コンドルセ勝者が決まらない確率

$$P(4,n)=2P(3,n)$$

が成り立っているので、$P(4,3)=2P(3,3)=\dfrac{1}{9}=0.11111\cdots$ となるであろう。

第2章　コンドルセ勝者が決まらない確率 (候補者数 3 または 4 の場合のシミュレーション)

2.1　候補者への線形順位づけの無作為標本

各投票者が候補者 $C_1, C_2, \cdots, C_n$ を線形順位づけした表を作る。
コンドルセ選挙において、投票者 m 人、候補者 n 人のとき、投票行動において各候補が線形順位付けされる確率を、$n!$ 通りの線形順序付けのどの線形順序に対しても同様に等しく $\frac{1}{n!}$ であると仮定する。(『 Impartial Culture Condition(IC) − Under this condition, each of the $n!$ linear preference orders are assumed to be equally likely so $p_i^n = \frac{1}{n!}$ for $i = 1, 2, \cdots, n!$ 』([5] P.169)

無作為手続き 数字 $1, 2, \cdots, n$ をただ一度ずつ使って並べる仕方 ($n!$ 通り) から無作為に (線形順位づけした) 並べ方を取り出す手続き.

b : 投票者数,　n : 候補者数,　i : 投票者 1, 2, ⋯, b の投票行動 のとき、つぎのように行えばよい (BASIC で記述)。

```
DIM voter(b,n)
   FOR i=1 TO b                    !  各投票者 i について
      FOR j=1 TO n                 !   j 番目に候補者 j を対応させる
         LET voter(i,j)=j
      NEXT j
   NEXT i

   FOR i=1 TO b                    !  各投票者 i について
      FOR j=1 TO n                   !  無作為に線形順位づける
         LET vote=INT(RND*(n+1-j)+1)
         LET tem=voter(i,n+1-j)
        LET voter(i,n+1-j)=voter(i,vote)
         LET voter(i,vote)=tem
      NEXT j
   NEXT i
```

説明
数字 1 から n の候補に、1 番から n 番の (番号) 札を持たせる。
最初に n 番の札を 1 から n のどれかの数 v に無作為に振り当てる。このとき、v 番目の札の持ち

主と n 番目の札の持ち主は札を交換する。こうして、n 番目の札 (を持つ数 v) が確定した。
次に、$n-1$ 番目の札を 1 から $n-1$ のどれかの数 v に無作為に振り当てる。このとき、v 番目の札の持ち主と $n-1$ 番目の札の持ち主は札を交換する。($n-1$ 番目の札 (を持つ数 v) が確定した。)

$\cdots$

次に 2 番の札を 1 から 2 のどれかの数 v に無作為に振り当てる。このとき、v 番目の札の持ち主と 2 番目の札の持ち主は札を交換する。(2 番目の札 (を持つ数 v) が確定した。)
このとき、1 番目の札 (を持つ数 v) も決まる。
ここでの v は乱数によって無作為になるように決められている。

この手続きにおいて、数字 $1, 2, \cdots, n$ をただ一度ずつ使った並び方 $a_1a_2a_3\cdots a_n$ が現れる確率を調べてみよう。$a_1a_2a_3\cdots a_n$ の並びの中で 1 は i_1 番目、2 は i_2 番目、$\cdots n$ は i_n 番目に存在するとする。すなわち、

$$\begin{pmatrix} 1 & 2 & 3 & \cdots & n \\ a_1 & a_2 & a_3 & \cdots & a_n \end{pmatrix} = \begin{pmatrix} i_1 & i_2 & i_3 & \cdots & i_n \\ 1 & 2 & 3 & \cdots & n \end{pmatrix}$$

このとき、n 番目の札を無作為に 1 番から n 番のどれかの番号 v に振り当てるとき、$(v=)i_n$ 番目に当たる確率は $\dfrac{1}{n}$ 。二回目に、$n-1$ 番目の札を無作為に 1 番から $n-1$ 番のどれかの番号 v に振り当てるとき、$(v=)i_{n-1}$ 番目に当たる確率は $\dfrac{1}{n-1}$ 。…つぎに、2 番目の札を無作為に 1 番から 2 番のどれかの番号 v に振り当てるとき、$(v=)i_2$ 番目に当たる確率は $\dfrac{1}{2}$ 。このとき、1 番目の札も i_1 番目に当たっているから $a_1a_2a_3\cdots a_n$ が現れる確率は $\dfrac{1}{n!}$ である。

2.2 コンドルセ勝者が決まらない確率を求めるプログラム

候補者が 3 人の時のコンドルセ勝者が決まらない確率を調べる BASIC 言語プログラムを作成した。コンドルセ選挙における二者択一多数決を通して、得票をカウントして勝敗を判定していくものである。第一章 1.3 で述べた 3 すくみ（決定不能）の確率を求めるプログラムである。プログラムのにおいて、コンドルセ勝者が決まらないことの判定は、すべての候補者の勝ち数を調べればわかるが、他の判定方法も考えられるかも知れない。複雑すぎるかも知れないが、以下に述べるパラメータを使ってみた。

コンドルセ勝者が決まるかどうかを調べるための (一つの) パラメータ。そのために、各投票者の投票が確定したときに、得点 tem を次のルールできめる。

ルール　候補者が 3 人を L, M, N とする。

L : M の組についての可能性は、L の得点が M の得点より多い場合 +1 で M の得点が L の得点より多い場合 -1 。
M : N の組についての可能性は、M の得点が N の得点より多い場合 +1 で N の得点が M の得点より多い場合 -1 。
N : L の組についての可能性は、N の得点が L の得点より多い場合 +1 で L の得点が N の得点より多い場合 -1 。

このときの tem の値の全ての可能性を表で表してみる。

L:M	M:N	N:L	tem
+1	+1	+1	3
+1	+1	-1	1
+1	-1	+1	1
-1	+1	+1	1
-1	-1	+1	-1
-1	+1	-1	-1
+1	-1	-1	-1
-1	-1	-1	-3

以上の表より、tem が +3, +1, -1, -3 の可能性はあるが、tem が 2, 0, -2 の可能性はない。以下候補者 3 人（L，M，N)、投票者 3 人（A，B，C）の場合について詳しく考えてみる。

$tem = +3$ の場合：　L：M，M：N，N：L の全てが +1 である。

- L：M について　　L > M
- M：N について　　M > N
- N：L について　　N > L

であるから、3 すくみで当選者を決めることができない (コンドルセ勝者が存在しない)。

$tem = +1$ の場合 L：M，M：N，N：L のうち、どれか 1 つが -1 で残りが +1 であるとき。この場合は 3 通りある。例えば、L：M が -1 となる場合を取りあげる。

L:M	M:N	N:L	tem
-1	+1	+1	+1

- L：M について　　L < M
- M：N について　　M > N
- N：L について　　N > L

であるから、3 すくみはなく L はコンドルセ勝者である。

L:M	M:N	N:L	tem
+1	-1	+1	+1

の場合、3 すくみはなく N はコンドルセ勝者である。

L:M	M:N	N:L	tem
+1	+1	-1	+1

の場合、3 すくみはなく L はコンドルセ勝者である。

$tem = -1$ の場合 L：M，M：N，N：L のうち、どれか一つが +1 他が -1 であるとき。この場合は 3 通りある。例えば、L：M が +1 となる場合を取りあげる。

L:M	M:N	N:L	tem
+1	-1	-1	-1

- L：M について　　L > M
- M：N について　　M < N
- N：L について　　N < L

であるから、3すくみはなく L はコンドルセ勝者である。

L:M	M:N	N:L	tem
-1	+1	-1	-1

の場合、3すくみはなく M はコンドルセ勝者である。

L:M	M:N	N:L	tem
-1	-1	+1	+1

の場合、3すくみはなく N はコンドルセ勝者である。

$tem = -3$ の場合： L：M, M：N, N：L の全ての値が $+1$ である。

- L：M について　　L < M
- M：N について　　M < N
- N：L について　　N < L

であるから、3すくみで当選者を決めることができない (コンドルセ勝者が存在しない)。
以上より、

補題 （コンドルセ勝者が存在していない場合の決定）

候補者数が 3 のとき、 決定不能（3すくみ）となるのは $tem = 3, -3$ のときである。

2.3 コンドルセ勝者が決まらない確率を求めるプログラム 候補者数 3

プログラムは以下である。

```
////////////////////////////////////////////////////////////////////
! NoCondorcetWinner.BAS
! Condoret Winner が決まらない確率
! 二者択一手続きで3すくみが起きる確率

INPUT a   ! Test 数
INPUT b   ! 投票者数
LET miss=0

FOR h=1 TO a

! A から G の投票者と候補者用の配列とその初期化

   DIM voter(b,3)   ! 配列
   DIM candidate(3)

   FOR i=1 TO b
```

```
   FOR j=1 TO 3
      LET voter(i,j)=j      ! 初期条件
   NEXT j
NEXT i

FOR i=1 TO 3
   LET candidate(i)=0      ! 初期条件
NEXT i

! 投票

RANDOMIZE
LET tem=0
LET vote=0
! 以下{1,2,3}の並び方の無作為の取り出し方
FOR i=1 TO b
   FOR j=1 TO 3
      LET vote=INT(RND*(4-j)+1)
      LET tem=voter(i,4-j)
      LET voter(i,4-j)=voter(i,vote)
      LET voter(i,vote)=tem
   NEXT j
NEXT i

! 二者択一

LET tem = 0     ! 初期条件

! candidate(1) と candidate(2) における二者択一
FOR j=1 TO 3
   LET candidate(j)=0      ! 初期条件
NEXT j

!  1:2
FOR j=1 TO b
   IF  voter(j,1) <voter(j,2)   THEN
      LET candidate(1)=candidate(1)+1
   ELSE
      LET candidate(2)=candidate(2)+1
   END IF
NEXT j

IF candidate(1)>candidate(2) THEN
```

```
   LET tem=tem+1
ELSEIF candidate(1)< candidate(2) THEN
   LET tem=tem-1
END IF
! candidate(2) と candidate(3) における二者択一
FOR j=1 TO 3
   LET candidate(j)=0    ! 初期条件
NEXT j

!  2:3
FOR j=1 TO b
   IF  voter(j,2) <voter(j,3)   THEN
      LET candidate(2)=candidate(2)+1
   ELSE
      LET candidate(3)=candidate(3)+1
   END IF
NEXT j

IF candidate(2)>candidate(3) THEN
   LET tem=tem+1
ELSEIF candidate(2)< candidate(3) THEN
   LET tem=tem-1
END IF

! candidate(3) と candidate(1) における二者択一
FOR j=1 TO 3
   LET candidate(j)=0     ! 初期条件
NEXT j

FOR j=1 TO b
   IF  voter(j,3) <voter(j,1)   THEN
      LET candidate(3)=candidate(3)+1
   ELSE
      LET candidate(1)=candidate(1)+1
   END IF
NEXT j

IF candidate(3)>candidate(1) THEN
   LET tem=tem+1
ELSEIF candidate(3)< candidate(1) THEN
   LET tem=tem-1
END IF
```

```
   ! 二者択一による当選者 : 決定不能
   IF tem=3 THEN
      LET miss=miss+1
      !  PRINT "当選者は決定不能 ( 3すくみ)"
   ELSEIF tem=-3 THEN
      LET miss=miss+1
      !  PRINT "当選者は決定不能 ( 3すくみ)"
   END IF

NEXT h

PRINT "決定不能 ( 3すくみ) でコンドルセ勝者が決まらない確率は";miss/a;"である
END
```

//

実行結果は次のようになった。

100000	3	0.05435
100000	3	0.05483
1000000	3	0.00556
10000	5	0.0793
100000	5	0.08562
1000000	5	0.0714
100000	7	0.07567
100000	7	0.07533
1000000	9	0.079606
1000000	9	0.078776
100000	11	0.07901
100000	11	0.07872

しかし、この値は投票のパラドックスが生じる確率 (参考文献 [1], p.61) とは少しずれが生じている。「公共選択」(参考文献 [1]) という本でいう‘ 投票のパラドックスが生じる確率 ’がどのような値なのかはっきりとはわからなかった。

2.4 コンドルセ勝者が決まらない確率を求めるプログラム 候補者数 4

では、候補者が 4 人の時のコンドルセ勝者が決まらない確率を調べてみよう。2.2 のプログラムの候補者が 3 人から 4 人に変化したプログラムで調べてみる。このとき、決まらないための必要十分条件は何か。コンドルセ勝者の定義から、それは二者択一で比べた結果、どの候補者も二者択一で勝ちが 3 回より少ない場合である。

コンドルセ勝者が決まらないとき、すなわち、コンドルセ選挙で候補者の勝ちの回数が 3 回より少ないときには、必ず 3 すくみが起こっている。4 人の候補者を A, B, C, D とし、以下で簡単

に証明をする。

1. ある候補者のコンドルセ選挙での得点が 3 のとき

＼	A	B	C	D
A	＼	○	○	○
B	×	＼	?	?
C	×	?	＼	?
D	×	?	?	＼

A,B,C,D の名前は自由であるので、3 回勝った者を A とする。このとき、A が全勝しているため、その他の BCD は少なくとも 1 回負けることになる。よって、3 勝した候補者はコンドルセ勝者となる。

2. 候補者のコンドルセ選挙での最高点が 2 のとき

＼	A	B	C	D
A	＼	○	○	×
B	×	＼	○	○
C	×	×	＼	?
D	○	×	?	＼

A,B,C,D の名前は自由であるので、2 回勝った者を A とする。このとき、4 候補者からの二者択一組み合わせは全部で 6 回あるので勝ち数の総和は 6 である。B,C,D の得点が 1 以下であることはあり得ないので、他の 2 回勝った者を B とし、A 対 B の戦いで勝った者を A とし、負けた者を B とする。そして A に勝った者を D とする。この場合の勝ち数の回数は {2,2,0,2 } か {2,2,1,1 } である。このようなとき、A は B に勝ち、B は D に勝ち、D は A に勝っている。すなわち、これは 3 すくみが起こっていると言える。もちろん、コンドルセ勝者はいない。

3. 候補者のコンドルセ選挙での最高点が 1 のとき

これは、あり得ない。なぜなら、4 人勝ち数の和は 6 なので、少なくとも誰かが 2 回勝つからだ。

よって、コンドルセ選挙での候補者の勝ちの回数が 3 回より少ない (コンドルセ勝者が決まらない) ときには、3 すくみが起きている。逆は成り立たない－というのは、コンドルセ勝者がいても、コンドルセ勝者以外の 3 候補者の間に 3 すくみが起こることがあり得るからである。
以下のプログラム NoCondorcetWinner4.BAS では、コンドルセ選挙で候補者の勝ちの最高回数が 3 回より少ないこと $maxw < 3$ を判定条件としている。

//

```
! NoCondorcetWinner4.BAS
! 4 候補者から Condoret Winner が決まらない確率
! 二者択一手続きで 3 すくみが起きる確率
```

```
INPUT a   ! Test 数
INPUT b   ! 投票者数
LET miss=0

FOR h=1 TO a

! A から G の投票者と候補者用の配列とその初期化

   DIM voter(b,4)   ! 配列
   DIM candidate(4)
   DIM win(4)

   FOR i=1 TO b
      FOR j=1 TO 4
         LET voter(i,j)=j       ! 初期条件
      NEXT j
   NEXT i

   FOR i=1 TO 4
      LET candidate(i)=0       ! 初期条件
   NEXT i

   ! 投票

   RANDOMIZE
   LET tem=0
   LET vote=0
   ! 以下{1,2,3,4}の並び方の無作為の取り出し方
   FOR i=1 TO b
      FOR j=1 TO 4
         LET vote=INT(RND*(5-j)+1)
         LET tem=voter(i,5-j)
         LET voter(i,5-j)=voter(i,vote)
         LET voter(i,vote)=tem
      NEXT j
   NEXT i

   ! 二者択一
   FOR j=1 TO 4
      LET  win(j)=0     ! 初期条件
   NEXT j

   ! candidate(1) と candidate(2) における二者択一
```

```
FOR j=1 TO 4
   LET candidate(j)=0      ! 初期条件
NEXT j

!  1:2
FOR j=1 TO b
   IF  voter(j,1) <voter(j,2)   THEN
      LET candidate(1)=candidate(1)+1
   ELSE
      LET candidate(2)=candidate(2)+1
   END IF
NEXT j

IF candidate(1)>candidate(2) THEN
   LET win(1)=win(1)+1
ELSEIF candidate(1)< candidate(2) THEN
   LET win(2)=win(2)+1
END IF

! candidate(2) と candidate(3) における二者択一
FOR j=1 TO 4
   LET candidate(j)=0      ! 初期条件
NEXT j

!  2:3
FOR j=1 TO b
   IF  voter(j,2) <voter(j,3)   THEN
      LET candidate(2)=candidate(2)+1
   ELSE
      LET candidate(3)=candidate(3)+1
   END IF
NEXT j

IF candidate(2)>candidate(3) THEN
   LET win(2)=win(2)+1
ELSEIF candidate(2)< candidate(3) THEN
   LET win(3)=win(3)+1
END IF

! candidate(3) と candidate(4) における二者択一
FOR j=1 TO 4
   LET candidate(j)=0      ! 初期条件
NEXT j
```

```
!  3:4
FOR j=1 TO b
   IF  voter(j,3) <voter(j,4)   THEN
      LET candidate(3)=candidate(3)+1
   ELSE
      LET candidate(4)=candidate(4)+1
   END IF
NEXT j

IF candidate(3)>candidate(4) THEN
   LET win(3)=win(3)+1
ELSEIF candidate(3)< candidate(4) THEN
   LET win(4)=win(4)+1
END IF

! candidate(4) と candidate(1) における二者択一
FOR j=1 TO 4
   LET candidate(j)=0      ! 初期条件
NEXT j

!  4:1
FOR j=1 TO b
   IF  voter(j,4) <voter(j,1)   THEN
      LET candidate(4)=candidate(4)+1
   ELSE
      LET candidate(1)=candidate(1)+1
   END IF
NEXT j

IF candidate(4)>candidate(1) THEN
   LET win(4)=win(4)+1
ELSEIF candidate(4)< candidate(1) THEN
   LET win(1)=win(1)+1
END IF

! candidate(1) と candidate(3) における二者択一
FOR j=1 TO 4
   LET candidate(j)=0      ! 初期条件
NEXT j

!  1:3
FOR j=1 TO b
```

```
      IF  voter(j,1) <voter(j,3)   THEN
         LET candidate(1)=candidate(1)+1
      ELSE
         LET candidate(3)=candidate(3)+1
      END IF
   NEXT j

   IF candidate(1)>candidate(3) THEN
      LET win(1)=win(1)+1
   ELSEIF candidate(1)< candidate(3) THEN
       LET win(3)=win(3)+1
   END IF

   ! candidate(2) と candidate(4) における二者択一
   FOR j=1 TO 4
      LET candidate(j)=0      ! 初期条件
   NEXT j

   !  2:4
   FOR j=1 TO b
      IF  voter(j,2) <voter(j,4)   THEN
         LET candidate(2)=candidate(2)+1
      ELSE
         LET candidate(4)=candidate(4)+1
      END IF
   NEXT j

   IF candidate(2)>candidate(4) THEN
      LET win(2)=win(2)+1
   ELSEIF candidate(2)< candidate(4) THEN
      LET win(4)=win(4)+1
   END IF

   ! 二者択一による当選者 : 決定不能
   LET max1=MAX(win(1),win(2))
   LET MAX2=MAX(max1,win(3))
   LET MAXw=MAX(MAX2,win(4))

   IF maxw<3 THEN
      LET miss=miss+1
      !  PRINT "当選者は決定不能（3すくみ)"
   END IF
NEXT h
```

```
PRINT "決定不能（３すくみ）でコンドルセ勝者が決まらない確率は";miss/a;"である
END
```

//

このプログラムの実行結果はこのようになった。

試行回数 a	投票者数 b	実験値 $P(4,n)$	$2P(3,n) = 21 - \overline{P}(3,n)$
100000	3	0.09937	0.11112
1000000	3	0.115539	0.11112
100000	5	0.13233	0.13888
1000000	5	0.13999	0.13888
100000	7	0.16308	0.15004
1000000	7	0.145206	0.15004
100000	9	0.15965	0.15596
1000000	9	0.155266	0.15596
100000	11	0.15463	0.15962
1000000	11	0.153157	0.15962

このように、パラドックスが生じる確率 (参考文献 [1], p.61) に近い値がでてくる。しかし、やはり、値が少し異なってくることもあった。

第3章　$P(3,n)$ の探求

3.1　候補者 3 、投票者 n のときの確率 $P(3,n)$

投票者が、候補者 3 人に順位をつける方法は、3! 通りである。具体的には、(C_1, C_2, C_3) は (1,2,3)、(1,3,2)、 (2,1,3)、 (2,3,1)、 (3,1,2) 、(3,2,1) の 6 通り。§ 1.4 で述べたように、投票者が n 人のとき、投票行動における起こり得る全ての事象は $3!^n$ 通りある。候補者 m 人、投票者 n 人のとき、$\overline{P}(m,n)$ をコンドルセ勝者が決まる確率とするとコンドルセ勝者が決まらない (存在しない) 確率は $P(m,n)(= 1-\overline{P}(m,n))$ である。候補者 m 人、投票者 n 人のとき、投票行動において起こり得る全てのを事象を調べることによって、一般的に $P(m,n)$ または $\overline{P}(m,n)$ を計算することは可能であろうか。$P(3,n)$ を求めていくとき、$n = 13$ で起こり得る全ての事象の数 $(3!)^{13} = 13060694016$ は多くのプログラミング言語が処理する符号なし整数型 (unsigned int) の最大値 4294967295 を超えてしまう。多倍長演算のライブラリを使うことを考える。多倍長演算ライブラリとしては、C++ ライブラリ NTL (http://www.shoup.net/ntl/) を使用することも考えたが、C++ 言語での並列処理のための OpenMP の parallel for のカウンタ変数には使えなかった。ところが、VC2010 では、64bit の整数型である long long int 型を使用して $2^{64} = 18446744073709551616$ 未満の整数が処理できることがわかったので、VC2010 上で C++言語による並列処理 OpenMP プログラム CondorcetWinner.cpp を書き確率を求めた。

```
////////////////////////////////////////////////////////////////////////
// CondorcetWinner.cpp

#include<iostream>
#include<cstdlib>
#include<ctime>
#include<omp.h>
#include<windows.h>
#include<MMSystem.h>

#pragma comment(lib,"winmm.lib")

// Project CondorcetWinner に  vcompd.lib を読み込む
// C:\Program Files\Microsoft Visual Studio 9.0\VC\lib\vcompd.lib

using namespace std ;

bool CondorcetCheck(int num_candidate,int num_voter,long long int num); // コンドルセの勝者か
```

```
void GetPermutation(int num_candidate,int num,int *ans);  // num 番目の順列を返す


int main(){


// OpenMP 使用しているかどうか

#ifdef _OPENMP

cout<<"\n++++++++++++++++++++++++++++++++++++++++++++"<<endl;
cout<<"OpenMP valid.  "<<omp_get_max_threads()<<"threads.";
cout<<"\n++++++++++++++++++++++++++++++++++++++++++++"<<endl<<endl;

#endif


///変数の宣言

DWORD startTime,endTime;

int num_candidate=0; // 候補者数
int num_voter=0; // 投票者数
long long int num_order=1; // 順位付けの数
long long int num_trials=1; // 試行数
long long int num_total=1; // 総数
long long int num_condorcet=0; // コンドルセ勝者数
float percent=0.0; // コンドルセ勝者が決定する確率


// 候補者数，投票者数を入力

cout<<"コンドルセの勝者が決定する確率を計算します．"<<endl
<<"候補者の数，投票者の数を入力してください．"<<endl<<endl
<<"候補者：";
cin >>num_candidate;
if(!cin || num_candidate<1)return 0; // (候補者数) が自然数以外の場合，終了
cout<<"投票者：";
cin >>num_voter;
if(!cin || num_voter<1)return 0; // (投票者数) が自然数以外の場合，終了

startTime = timeGetTime();
```

```
//総数の計算

for(int i=1; i<=num_candidate; i++)num_order*=i; // 順位付けの数＝(候補者数)の階乗
for(int i=0; i<num_voter-1; i++){
num_trials*=num_order;
num_total*=num_order;
}
num_total*=num_order; // 総数＝(順位付けの総数)の(投票者数)乗

cout<<"起こり得る事象の総数は，\n"<<num_total<<"通りです．"<<endl;


// CondorcetWinnerがいるかどうか，1からnum_trialsまで調べる

#pragma omp parallel
{
#pragma omp for
for(long long int i=0; i<num_trials; i++){

if(CondorcetCheck(num_candidate, num_voter, i)){ //CondorcetCheckが1の時
#pragma omp critical
num_condorcet++;
}
}
}


endTime = timeGetTime();

// 結果の表示

percent=(float)num_condorcet/(float)num_trials; //(コンドルセ勝者数)/(総数)

cout<<num_total<<"中，"<<num_condorcet*num_order<<"回．"<<endl;
cout<<"コンドルセの勝者が決まる確率："<<percent<<endl;
cout<<"コンドルセの勝者が決まらない確率："<<1.0-percent<<endl;
cout<<"計算時間は，"<<endTime-startTime<<"msです．";

int end;
cin>>end;

return 0;
}
```

```
void GetPermutation(int num_candidate,int num,int *ans){


int order=1;

for(int i=1; i<=num_candidate; i++)order*=i;

int tem_num=num;
int *place;
place=new int[num_candidate];
for(int i=0; i<num_candidate; i++)place[i]=0;

int *key;
key=new int[num_candidate];
for(int i=0; i<num_candidate; i++)key[i]=0;


for(int i=0; i<num_candidate; i++)key[i]=i;

for(int i=1; i<=num_candidate; i++){
place[num_candidate-i]=tem_num%i;
tem_num=(int)tem_num/i;
}


for(int i=0; i<num_candidate; i++){
bool flag=1;
for(int j=0; j<num_candidate; j++){
if(key[j]==place[i] && flag){
ans[i]=j;
key[j]=-5;
flag=0;
for(int k=j; k<num_candidate; k++){
key[k]--;
}
}
}
}

delete[] place;
delete[] key;
```

```
}

bool CondorcetCheck(int num_candidate,int num_voter,long long int num){

long long int tem_num=num;
long long int num_order=1;

int *ans;
ans=new int[num_candidate];
for(int k=0; k<num_candidate; k++)ans[k]=k;

int *vote_choice;
vote_choice=new int[num_voter];
for(int i=0; i<num_voter; i++)vote_choice[i]=0;

int *vote;
vote=new int[num_candidate*num_voter];
for(int i=0; i<num_candidate*num_voter; i++)vote[i]=0;

int *win;
win=new int[num_candidate*num_candidate];
for(int i=0; i<num_candidate*num_candidate; i++)win[i]=0;

int *count_points;
count_points=new int[num_candidate];
for(int i=0; i<num_candidate; i++)count_points[i]=0;

for(int i=1; i<=num_candidate; i++)num_order*=i;

for(int i=0; i<num_voter-1; i++){
vote_choice[i]=tem_num%num_order;
tem_num/=num_order;
}


for(int i=0; i<num_voter; i++){

GetPermutation(num_candidate,vote_choice[i],&*ans);

for(int j=0; j<num_candidate; j++)vote[i*num_candidate+j]=ans[j];

}
```

```
delete[] ans;

for(int i=0; i<num_voter; i++){
for(int j=0; j<num_candidate; j++){
for(int k=0; k<num_candidate; k++){
if(vote[i*num_candidate+j]<vote[i*num_candidate+k]){
win[j*num_candidate+k]++;
}else if(vote[i*num_candidate+j]>vote[i*num_candidate+k]){
win[j*num_candidate+k]--;
}
}
}
}


for(int i=0; i<num_candidate; i++){
for(int j=0; j<num_candidate; j++){
if(win[i*num_candidate+j]>0){
count_points[i]++;
}
}
}

bool only_flag=0;
int max=0;

for(int i=0; i<num_candidate; i++){
if(max<count_points[i]){
max=count_points[i];
}
}

if(max==num_candidate-1)only_flag=1;


delete[] vote;
delete[] vote_choice;
delete[] win;
delete[] count_points;

return only_flag;

}
```

//
次節その結果である。

3.2 CondorcetWinner.cpp の結果

(m,n)	全事象 $(m!)^n$	コンドルセ勝者が存在	$\overline{P}(m,n)=1-P(m,n)$
(3,3)	216	204	0.944444
(3,5)	7776	7236	0.930556
(3,7)	279936	258936	0.924983
(3,9)	10077696	9291876	0.922024
(3,11)	362797056	333840744	0.920186
(3,13)	13060694016	12001884264	0.918932
(3,15)	470184984576	431639416944	0.91802
(4,3)	13824	12288	0.888889
(4,5)	7962624	6856704	0.861111
(4,7)	4586471424	3898343424	0.849966
(4,9)	2641807540224	2229811544064	0.844048
(5,3)	1728000	1451520	0.840000
(5,5)	24883200000	19918379520	0.800475
(6,3)	373248000	297768960	0.797778
(7,3)	128024064000	97452149760	0.761202
(8,3)	65548320768000	47801224396800	0.729252
(9,3)	47784725839872000	33501077884108800	0.701083

$n=3,5,7,9,11,13,15$ のとき、WILLIAM V. GEHRLEIN の論文の結果 (P.174) と一致する。

では、$n=4,6,8$ のときはどうだろうか。
WILLIAM V. GEHRLEIN の論文 [5, P.182] に以下のような式が示されている：

$$\overline{P}(m,n)=\mathrm{C}_0^m+\sum_{i=1}^{\frac{m-2}{2}}\mathrm{C}_i^m\overline{P}(2i+1,n).$$

ここで、係数は

m	C_0^m	C_1^m	C_2^m	C_3^m	C_4^m	C_5^m
4	-1	2				
6	3	-5	3			
8	-17	28	-14	4		
10	155	-255	126	-30	5	
12	-2073	3410	-1683	396	-55	6

.

つまり、次式が成り立つと述べられている：

$$\overline{P}(4,n) = 2\overline{P}(3,n)-1 \tag{3.1}$$
$$\overline{P}(6,n) = 3\overline{P}(5,n)-5\overline{P}(3,n)+3 \tag{3.2}$$
$$\overline{P}(8,n) = 4\overline{P}(7,n)-14\overline{P}(5,n)+28\overline{P}(3,n)-17 \tag{3.3}$$

ここで、計算しやすくするため、プログラム CondorcetWinner.cpp を実行して求めた $\overline{P}(m,n)$ を分母・分子の最大公約数で割り、既約分数にしてある。(最大公約数はプログラム getGCD.cpp で求めた。) こうして

$$\begin{aligned}\overline{P}(3,3) &= \frac{17}{18}, \qquad \overline{P}(3,5)=\frac{67}{72}, \qquad \overline{P}(3,7)=\frac{10789}{11664}\\ \overline{P}(3,9) &= \frac{774323}{839808}, \qquad \overline{P}(5,3)=\frac{21}{25}, \qquad \overline{P}(7,3)=\frac{33569}{44100}.\end{aligned}$$

$\frac{8}{9}=2\times\frac{17}{18}-1$ であるから、$\frac{8}{9}=\overline{P}(4,n)=2\overline{P}(3,n)-1$ が成り立つことも観察される。

$\overline{P}(4,5)=\frac{31}{36}$, $\overline{P}(4,7)=\frac{4957}{5832}$, $\overline{P}(4,9)=\frac{354419}{419904}$, $\overline{P}(6,3)=\frac{359}{450}$ および $\overline{P}(8,3)=\frac{536}{735}$ も関係式 (3.1), (3.2), (3.3) を満たすことが観察される。

例えば、$\overline{P}(3,9)=\frac{774323}{839808}$ に対して

$$\overline{P}(4,9)=2\times\frac{774323}{839808}-1=\frac{354419}{419904}$$

が成り立つ。また、$\overline{P}(7,3)=\frac{33569}{44100}$, $\overline{P}(5,3)=\frac{21}{25}$, $\overline{P}(3,3)=\frac{17}{18}$ に対して

$$\overline{P}(8,3)=4\times\frac{33569}{44100}-14\times\frac{21}{25}+28\times\frac{17}{18}-17=8040/11025=536/735$$

が成り立っている。

以上のように、プログラム CondorcetWinner.cpp を実行して求めた確率は、m が奇数のとき WILLIAM V. GEHRLEIN の表と一致し、m が偶数のとき関係式で求めた値と一致する。

WILLIAM V. GEHRLEIN の表は、小数表示であるため、実際には、CondorcetWinner.cpp の実行で求めた確率と一致するとは、断定できない。しかし、奇数 m と偶数 m において、論文に示されていた関係式をも満たしていることから、CondorcetWinner.cpp は、$\overline{P}(m,n)$ を求めることができていると考えられる。

3.3 $\overline{P}(m,n)$ の計算時間と計算可能性

Core i7 CPU 740 QM（1.73GHz）での結果

投票者数 n ＼ 候補者数 m

n ＼ m	3	4	5	6	7	8
3	7 m 秒	8 m 秒	24 m 秒	500 m 秒	24 秒	1790 秒
5	17 m 秒	336 m 秒	211 秒	(72 時間)	(20 年)	(9.3 万年)
7	63 m 秒	198 秒	(844 時間)			
9	1791 m 秒	(32 時間)				
11	75 秒					
13	3120 秒					
15	(35 時間)					

Core i7 CPU 920 (4.2GHz) での結果

投票者数 n ＼ 候補者数 m

$n \setminus m$	3	4	5	6	7	8	9
3	0 m 秒	16 m 秒	16 m 秒	249 m 秒	10 秒	741 秒	1164 分
5	0 m 秒	125 m 秒	87 秒	(36 時間)	(8 年)	(3.8 万年)	(2.9 億年)
7	16 m 秒	83 秒	(15 日)				
9	780 m 秒	(980 分)					
11	31 秒	(483 日)					
13	1269 秒						
15	848 分						
17	(24 日)						

実際に、計算に要した時間から、もしこのプログラムで計算をした場合どれくらいの時間が必要かを見積もってみた。括弧書きしたものが見積もった時間である。投票者が 2 人増えるごとに、全事象は、$(m!)^2$ 倍になる。また、1 通りを調べるのにかかる時間も、投票者の数に比例して多くなる。以上のことから、$\overline{P}(3,17)$ を求めるのにかかる時間は、$\overline{P}(3,15)$ を求めるのにかかる時間の $(3!)^2+\alpha=36$ 倍以上、$\overline{P}(4,11)$ は $\overline{P}(4,9)$ の $(4!)^2+\alpha=576$ 倍以上とした。

このプログラムは、投票者の数が増えるごとに指数関数的に時間がかかる。今回、Core i7 CPU 920 (4.2GHz) という高性能 CPU を使用したが、$\overline{P}(9,5)$ を計算することは、不可能だと言わざるを得ない。つまり、アルゴリズムを考え直すか、違ったアプローチの方法を考えなければ、これ以上、表をうめることは難しい。今回は、厳密な確率を求めるプログラム CondorcetWinner.cpp を作ったが、厳密な数字が必要でない場合には、コンピュータによるシミュレーションを行うことも有効な手であることがわかる。

研究を振り返って

並列処理プログラムによって、計算時間が縮まることを体験できた。しかし、CPU 側の技術も素晴らしく、インテル R ターボブースト・テクノロジーによって、並列化されていないプログラムにも高速に処理する工夫がなされていることがわかった。インテルの最新のコンパイラは、コンパイラが自動的に可能な限り、並列化するといった機能も備えているようだ。きっと、私が下手なプログラムを書くよりも、効率の良い並列処理プログラムを作ってくれるに違いない。今後は、コンパイラが自動で並列化できないような、難しい内容にも挑戦したい。　また、CondorcetWinner が決定する確率をプログラムで求める中で、全ての事象を計算することの大変さも味わった。特に今回は、全事象の増え方が爆発的だったため、候補者・投票者が少ないときには解けるが、多くなると全く手が出せないという結果になった。時間が永遠にあれば解けるといったアルゴリズムは、時間の限りのある我々にとって、何の役にも立たない。CondorcetWinner が決定する確率を今回以上に求めるためにも、数学的に問題に立ち向かい、様々なアプローチを試してみたいと感じた。(中村貴亮)

参考文献

[1] 小林良彰著, 公共選択 現代政治学叢書 9, 東京大学出版会, 1988.

[2] 松原望 : 計量社会科学ワークショップ, http://www.qmss.jp/qmss/ .

[3] 選挙を「数学的に」考えてみよう! 投票のパラドックス,「当確」の不思議 (協力 松原 望), Newton(ニュートン) 2009 年 10 月号, ニュートンプレス.

[4] Peter C. Fishburn, A proof of May's theorem p(m, 4) = 2p(m, 3), Behavioral Science, Volume 18, Issue 3(May 1973) , p.212.

[5] William V. Gehrlein, CONDORCET'S PARADOX, Theory and Dicision 15(1983), 161-197.

[6] VisualC++ の OpenMP, Visual Studio 2010, http://msdn.microsoft.com/ja-jp/library/tt15eb9t.aspx .

付録　newCondorcetWinner.cpp

```
//////////////////////////////////////////////////////////////////////
// newCondorcetWinner.cpp  (nakamura/urata)

#include<iostream>
#include <vector>
#include <algorithm>
#include<cstdlib>
#include<ctime>
#include<omp.h>
#include<windows.h>
#include<MMSystem.h>

#pragma comment(lib,"winmm.lib")

using namespace std ;


typedef struct LPoint{
int p[10];
} LPoint;

CONST long int WIDTH=39916802;   // 11!+2=39916800+2

LPoint lpt[WIDTH];// 10!=3628800,  11!=39916800,  12!=479001600,  13!=6227020800

LPoint newL={0, 1, 2, 3, 4, 5, 6, 7, 8, 9}, *w=&newL;


void SetPermutation_ordered(int num_candidate, LPoint []);  //順列 S_num_candidate を
準備
bool CondorcetCheckP(int num_candidate,int num_voter,long long int num, LPoint []);
//num 番目の投票行動に対し，コンドルセの勝者の存在を調べる.


int main(){

//OpenMP 使用しているかどうか

#ifdef _OPENMP
```

```
cout<<"\n++++++++++++++++++++++++++++++++++++++++++++"<<endl;
cout<<"OpenMP valid.  "<<omp_get_max_threads()<<"threads.";
cout<<"\n++++++++++++++++++++++++++++++++++++++++++++"<<endl<<endl;

#endif

///変数の宣言

DWORD startTime,endTime;

int num_candidate=10; //候補者数 10
int num_voter=0; //投票者数
long long int num_order=3628800; //順位付けの数 10!=3628800
long long int num_trials=1; //試行数
long long int num_total=1; //投票行動の総数
long long int num_condorcet=0; //コンドルセ勝者数
float percent=0.0; //コンドルセ勝者が決定する確率

char guess;

start:

num_candidate=0; //候補者数
num_voter=0; //投票者数
num_order=1; //順位付けの数
num_trials=1; //試行数
num_total=1; //投票行動の総数
num_condorcet=0; //コンドルセ勝者数
percent=0.0; //コンドルセ勝者が決定する確率

//候補者数，投票者数を入力

cout<<"\n コンドルセの勝者が決定する確率を計算します．"<<endl
<<"候補者の数 (< 11)，投票者の数を入力してください．"<<endl<<endl
<<"候補者：";
cin >>num_candidate;
if(!cin || num_candidate<1 || num_candidate>10)return 0;  //(候補者数) が自然数以
外の場合，終了
cout<<"投票者：";
cin >>num_voter;
if(!cin || num_voter<1)return 0;  //(投票者数) が自然数以外の場合，終了

startTime = timeGetTime();
```

```
//////順位付けの数の計算，順列 S_num_candidate の準備

for(int i=1; i<=num_candidate; i++)num_order*=i;  //順位付けの数＝（候補者数）の階乗 K = m!
cout<<"順位付けの数は，  "<<num_order<<" 通りです．"<<endl;

    SetPermutation_ordered( num_candidate, lpt);

////////////////////////////////////////////////////////////////////

//総数の計算

//for(int i=1; i<=num_candidate; i++)num_order*=i;  //順位付けの数＝（候補者数）の階乗
for(int i=0; i<num_voter-1; i++){
num_trials*=num_order;
num_total*=num_order;
}
num_total*=num_order;  //投票行動の総数＝（順位付けの数）の（投票者数）乗

cout<<"起こり得る事象の総数は，  "<<num_total<<" 通りです．"<<endl;

//CondorcetWinner がいるかどうか，1 から num_trials まで調べる
/*
num_voter 名の投票者の各投票行動に関して，コンドルセの勝者を決めるための計算は，この投票行動への
順列 S_num_candidate による置換に関して不変であるから，一人の投票行動を固定した num_total/num_order=num_trial
個の投票行動に対して，コンドルセの勝者の存在を調べれば充分である．

投票行動の総数 num_total =（順位付けの数）^（投票者数）=（num_candidate!）^num_voter
起こり得る事象の総数 "<<num_total<<" 通りの中で， num_order 通りの順列による置換は等しい結果をもたらす．
調べるべき事象の総数は，num_trials 通り．投票者 num_voter-1 の投票を lpt[0]=newL と固定してよい．
*/
#pragma omp parallel
{
#pragma omp for
for(long long int i=0; i<num_trials; i++){

if(CondorcetCheckP(num_candidate, num_voter, i, lpt)){ //CondorcetCheck が 1 の時
#pragma omp critical
```

```
num_condorcet++;
}
}
}

endTime = timeGetTime();

//結果の表示

percent=(float)num_condorcet/(float)num_trials;   // (コンドルセ勝者数/num_order)/num_trials=(コ
ンドルセ勝者数)/(総数)

cout<<num_total<<"中, "<<num_condorcet*num_order<<"回. "<<endl;
cout<<"コンドルセの勝者が決まる確率："<<percent<<endl;
cout<<"コンドルセの勝者が決まらない確率："<<1.0-percent<<endl;
cout<<"計算時間は, "<<endTime-startTime<<"ms です. \n";

// ending:
  cout << " \n";    // printf(" \n");
  cout << "もう一度やってみますか？ <yes(return)/no> = ";   // printf("もう一度やっ
てみますか？ <yes(return)/no> = ");
  cin >> guess;
  if( (guess != 'N') & (guess != 'n')){  goto start;}

return 0;
}


void SetPermutation_ordered(int num_candidate, LPoint lpt[] ){

  const int n = num_candidate;

  std::vector<int> data;
  // [0, 1, 2, ..., n-1] というサイズ n の配列を作成
  for(int i=0; i<n; ++i){
    data.push_back(i);
  }

  // 全ての順列を出力
  long long int num=0;
  do{
      for( int i=0; i<n; ++i) lpt[num].p[i] = data[i];
```

```
  num++;
  }while(next_permutation(data.begin(), data.end()));

  return;
}

bool CondorcetCheckP(int num_candidate,int num_voter,long long int num, LPoint lpt[] ){

long long int tem_num=num;  //temporary_number=num<num_trials
long long int num_order=1;

long long int *vote_choice;
vote_choice=new long long int[num_voter];
for(int i=0; i<num_voter; i++)vote_choice[i]=0;

int *vote;
vote=new int[num_candidate*num_voter];  //候補者数×投票者数の記録票
for(int i=0; i<num_candidate*num_voter; i++)vote[i]=0;

int *win;
win=new int[num_candidate*num_candidate];  //候補者数×候補者数の記録票 win[j*num_candidate+k] 候
補者 j,k の順位の優劣を記録
for(int i=0; i<num_candidate*num_candidate; i++)win[i]=0;

int *count_points;
count_points=new int[num_candidate];  //候補者数の記録票 count_points[i] 候補者 i の
優位得点
for(int i=0; i<num_candidate; i++)count_points[i]=0;

for(int i=1; i<=num_candidate; i++)num_order*=i;  //順位付けの数＝(候補者数)の階乗

for(int i=0; i<num_voter-1; i++){
vote_choice[i]=tem_num%num_order;  //試行番号 tem_num の num_order_進法表示を投票
行動 vote_choice[] に対応させる.
tem_num/=num_order;
}

for(int i=0; i<num_voter; i++){

        for(int j=0; j<num_candidate; j++) vote[i*num_candidate+j]=lpt[vote_choice[i]].p[j];  //
試行番号 tem_num における投票者 i の投票行動

}
```

```
for(int i=0; i<num_voter; i++){
for(int j=0; j<num_candidate; j++){
for(int k=0; k<num_candidate; k++){
if(vote[i*num_candidate+j]<vote[i*num_candidate+k]){
win[j*num_candidate+k]++;  // (投票者 i からの) 候補者 j の順位 > candidate k の順
位
}else if(vote[i*num_candidate+j]>vote[i*num_candidate+k]){
win[j*num_candidate+k]--;
}
}
}
}


for(int i=0; i<num_candidate; i++){
for(int j=0; j<num_candidate; j++){
if(win[i*num_candidate+j]>0){  // 候補者 i の総得点  > 候補者 j の総得点
count_points[i]++;
}
}
}

bool only_flag=0;
int max=0;

for(int i=0; i<num_candidate; i++){
if(max<count_points[i]){
max=count_points[i];
}
}

if(max==num_candidate-1)only_flag=1;  //コンドルセの勝者有り.

delete[] vote;
delete[] vote_choice;
delete[] win;
delete[] count_points;

return only_flag;
}
```