投票の仕方でこんなに違う！-投票の数学理論-

啓明学院中学校高等学校　中学 3 年: 眞部光, 高校 1 年: 青野大志, 高校 1 年: 高橋 祥英, 高校 3 年: 立花優季

1　研究の要約

代表的な投票方法である、最大多数法式、決選投票式、ボルダ式、コンドルセ式について計算機シミュレーションで調べ、投票方法によって結論が変わる確率を計算した。また、自由民主党総裁選で使われたドント方式を計算機で調べて、ドント式よりも普通の比例配分式の方が適切であるという結論を得た。次にボルダで使うポイント制で、和だけでなく、積を使うことを考え、和と積による評価逆転について数学的に研究した。

2　研究の動機と目的

学校の先生に選挙の方法について教えてもらう機会があった。小学校以来、クラスの議決は手をあげて一番多い案に決めるか、上位 2 つ（2 名）を選んで決選投票するのが当たり前と考えてきたが、世界ではいろんな投票の方法が使われているようであり、投票方法によって結果が変わり得ることを知った。これは社会にとっては非常に重要なことなのできちんと研究してみたいと考えた。面白い結果が得られたので、多くの人に知ってもらいたいと考えている。

3　方法

研究方法としては、計算機によるシミュレーションと数学的な理論を使う。計算機言語としては、Mathematica, Python, C を使った。Mathematica は膨大な数学的関数と多くのリスト操作を持っているため、複雑なプログラムを作るのが容易であるが、欠点としては数値計算は速くない。計算速度を必要とする場合はリスト処理の得意な Python を用い、特に速度重視の場合にはプログラムを作るのがやや難しいが C を用いた。可能な場合は全検索し、それが膨大な数になって不可能な場合は乱数を用い、実験回数を増やすことで精度を上げた。

4　結果と考察

どのようなことをやってみたかを書いていきながら、考察を加えていく。まず、普通の学校でありそうな状況を作ってみた。

例 1. ある学校のクラスでは、文化祭に行うクラスの展示の案が 5 つあり、a は輪投げ、b はスーパーボールすくい、c は射的、d はお化け屋敷、e はモグラ叩きであった。これから投票で決めるのだが、42 人の生徒の好みは次の図 1 のように、5 つの場合に別れていたとする。これは各自の心の中にあるもので、実際には選挙の途中で変わってしまうこともあるが、ここでは最後まで変わらないとして考える。

普通は学校では生徒が一人一回手をあげるというやり方で決める。その場合は次の 2 つのいずれかの方法を使う。

(i) 一番希望が多い案を選ぶ。この場合 c の射的が一番多いので、それに決まる。

(ii) 過半数を超えたものを選ぶ。この場合該当する案がないので、上位 2 つの案である c と d で決選投票する。決選投票になると、生徒の気持ちは図 1 のようになっているので、c, d 以外を無視して c と d だけを比べると、c を上とする人 16 に対して、d を上とする人 26 なので、この場合は d になる。

しかしこの (i) と (ii) 以外にはやり方はないのだろうか。佐伯 [1] や坂井 [2] によると、伝統的に使われている方法として、他にボルダの方法とコンドルセの方法というものがある。

(iii) ボルダの方法では、ポイント制で、$5, 4, 3, 2, 1$ 点を好みの順につけて、それらの和を求める。ここでは1位に5点、2位に4点、3位に3点、4位に2点、5位に1点をつける。この場合では a が $3\times16+2\times14+1\times12 = 88$ 点、b が $2\times16+1\times14+3\times12 = 82$ 点、c が $5\times16+4\times14+2\times12 = 160$ 点、d が $1\times16+5\times14+4\times12 = 134$ 点、e が $4\times16+3\times14+5\times12 = 166$ 点となり、e が一位となる。ボルダの方法が民主主義には一番適しているという考えがあり、確かにこの図を見ると、全員から一定の支持を得ているのは確かにこの e である。ボルダの方法は政治の世界は一部の国で使われているだけだが、コンテストでは主流である。

上の (i),(ii), (iii) ではすべて別の結果になっている。しかも、これらはよく使われる投票方法なのである。これでは投票方法の公正さなど信じられなくなる。多くの人が納得しやすい方法としてコンドルセの方法というものがある。

(iv) コンドルセの方法では選択肢 2 つずつを選んで比べる。そして、他のどの選択肢と 1 対 1 で比べても優位なものがあると、それをコンドルセ勝者と呼ぶ。コンドルセ勝者がいたら、その人（その案）を選ぶことは多くの人の支持を受けるはずである。しかし、難しいことはコンドルセ勝者は必ずしも存在しないということである。

今回の例のデータで考えてみる。*2* つずつ比べた場合に、まず e は a, b, d より優位であることがわかる。これを便宜上 $e > a, b, d$ と書く。また $d > c$ であるが、$c > e$ も成り立つ。したがって、d, c, e は2つずつ比較すると、どれが一番良いという判定ができない。ちょうどジャンケンのグー、チョキ、パーのような状態である。

結果として、みんなが図 *1* のような好みを持っているとコンドルセの方法で選ばれる案はない。しかし、他にも考えることはできる。

(v) ポイント制で、それらの積を使う。これは後で詳しく書くが、対数を使ってから和を求めることと同じで、和を使う方法の１つである。

この例を見ると、投票方法によって結果が全く違ってしまう。高校生にとって文化祭は大事だが、社会の中にはもっと大事な決定も多くある。多くの人の命に関わるような決定もあるのだから、大変な問題を含んでいる。

図 1.

	16 人	14 人	12 人
1st	c	d	e
2nd	e	c	d
3rd	a	e	b
4th	b	a	c
5th	d	b	a

4.1 選挙方式の比較

坂井 [2] は多数決とボルダの方法とコンドルセの方法を理論的に比較しているが、統計的、あるいは確率的な比較は行なっていない。佐伯 [1] では確率的な計算は全て他の研究者の結果を引用している。私たちは例 1 のような状況で、投票方法の違いによる逆転がどのくらいの率で起きるかを計算機で調べてみた。ただし、全ての場合を計算すると、42 人の投票で、選択肢が 5 つだから、5^{42} で 2.27×10^{29} になってしまう。そのために、乱数を使って、42 人が 5 つの選択肢から 1 つを選ぶという状態を 100000 回作って、集計するという方法を用いた。コンピュータ言語は Python でプログラムを作った。

例 2. 42 人のクラスで上位 *2* つを選んで決選投票する場合 と単純な多数決を比較してみた。同じ順位になってしまって 1 つの案を選ぶことができないケースが 16 %程度ある。しかしながら、もっと大事なことは 2 つの方法で 1 つの案が選ばれるが、それが異なる場合がかなり多いということである。

図 2.

多数決と決選投票で共に 1 つの案が選ばれそれが一致する	51.78 %
多数決と決選投票で共に 1 つの案が選ばれるが異なる	32.00 %

例 3. 42 人のクラスで上位 2 つを選んで決選投票する場合 とポイントを $1,2,3,4,5$ 点ずつつけてポイントを足し合わせて選ぶ方法を比較してみた。この場合も、2 つの方法で 1 つの案が選ばれるが、それが異なる場合がかなり多い。

図 3.

和によるポイント制と決選投票で共に 1 つの案が選ばれそれが一致する	58.34 %
和によるポイント制と決選投票で共に 1 つの案が選ばれるが異なる	27.11 %

私達は一般的な和により集計する方式以外に、ポイントをかけ合わせて集計する方式に関してもシミュレーションを行なってみた。ここでは、2 つの方法で 1 つの案が選ばれるが、それが異なる場合がかなり多い。このことから積を使う評価方法は、かなり違った状況を作り出すことがわかる。

例 4. 42 人のクラスで上位 2 つを選んで決選投票する場合 とポイントを $1,2,3,4,5$ 点ずつつけてポイントをかけ合わせて選ぶ方法の比較。

図 4.

積によるポイント制と決選投票で共に 1 つの案が選ばれそれが一致する	34.84 %
積によるポイント制と決選投票で共に 1 つの案が選ばれるが異なる	50.50 %

例 5. 42 人のクラスで上位 2 つを選んで決選投票する場合 とポイントを $1,2,3,4,5$ 点ずつつけてポイントをかけ合わせて選ぶ方法を比較。

図 5.

積によるポイント制と和によるポイント制で共に 1 つの案が選ばれそれが一致する	43.90 %
積によるポイント制と和によるポイント制で共に 1 つの案が選ばれるが異なる	56.02 %

4.2 コンドルセのパラドックス

例 6. 例 1 における d,c,e のように、2 つずつ比較すると、どれが一番良いという判定ができない場合がある。このような状況を循環順序が存在するという。しかし、循環順序があってもコンドルセ勝者が存在することはある。例えば、e は a,b,c,d のどれと比べても優位で、しかし a,b,c の中ではどれが一番良いかが判定できないというような場合は、e はコンドルセ勝者だが、循環順序は存在する。1 つの案だけを選ぶなら、それでもいいが、コンテストのように順位をつける場合には困る。ここでは 42 名の投票者が 5 つの選択肢に関して、投票した時、コンドルセ勝者が出る確率。循環順序が出る確率を調べた。このような研究は佐伯 [1] では論文 [3] を参照している。しかし論文 [3] ではコンドルセ勝者が存在するかどうかに絞って正規分布近似で計算していた。私達は計算機を用いて、全ての循環順序を探しながら、コンドルセ勝者を探した。ここでは数学ソフト *Mathematica* を用いた。*Mathematica* はリストを処理するための多くの関数があるので、このような複雑な計算には適している。

図 6.

循環順序なしで、コンドルセ勝者が出る。	46.1 %
循環順序があるが、コンドルセ勝者が出る。	8.25 %
循環順序が生まれる	26.3 %

4.3 自民党総裁選

昨年の自民党総裁選では、党員の投票結果をドント式で配分して、党員の票を国会議員数と同じ総計 382 にそろえてから、国会議員票 382 との合計で競うという方式であった。ここで過半数の候補者が出れば、その人が総裁になるはずだったが、過半数の票をとった候補者がいなかったため、決選投票になり、その場合は党員票は都道府県の数になる方式が使われた。なお、ドント方式は比例代表制のようなタイプの選挙では多くの国で使われているが、総裁選挙で使用が合理的かどうかに疑問を持ち調べてみた。

まずドント式の定義を書く。

定義 1. n 人の有権者が m 人の候補者に投票し、この中から各候補者の得票 p_i を $1,2,3,\ldots$ で割った値を計算し一覧表を作り、この一覧表から値の大きい順に h 個を見つけ、それをもとに候補者への配分を決める

昨年の総裁選では、党員票数は河野 太郎氏、岸田 文雄氏、高市 早苗氏、野田 聖子氏がそれぞれ、$335,046$（44.1 %） $219,338$ （28.9 %） $147,764$ （19.4 %） $57,927$ （7.6 %）となり、ドント式で $169,110,74,29$ という票配分になった。比率が $168.388 : 110.235 : 74.2635 : 29.1131$ なので、比例配分で、小数の部分が多い順に配分すると、同じ票数になる。この結果を見ると、ドント式は比例配分と同じだから、納得できると考えるべきか、あるいは比例配分と同じなら、多くの人が納得しやすい比例配分を使えばよいのではないかという疑問もおこる。

実はドント式と比例配分がずれる場合がある。一応今回の総裁選とあまり変わらないような党員票で違いが出るケースを計算機で探したところ、次のような場合が見つかった。実際には 4 名の候補者が 100 万を超える党員から票を得る場合を全部調べると、膨大な数になってしまうので、投票結果の比率をランダムに作り、ドント方式と比例配分方式の結果が異なるものを探した。それから、得票数の差を見ていき、最も差が小さいものを出力させた。

例 7. 例えば候補者 A, B, C, D の党員票が $614402, 261101, 120163, 108670$ なら、ドント式では $214, 90, 41, 37$ で、比例配分で $212, 90, 42, 38$ となる。実際の比率は $212.527, 90.3172, 41.5655, 37.59$ であるから、小数の部分をみると、大きさは D, C, A, B の順であるが、D, C には配分されず、A に 2 票が配分されている。この結果で、議員票を加えた後で A が過半数を取ったり、あるいは D, C の議員票が多くて、上位 2 位になるチャンスがあったにも関わらず、ドント式でそれが実現しないというようなことになると重大な問題である。

もし初めから候補者が 2 名だとすると、次のようなこともあり得る。候補者 A, B の党員票が $550726, 553610$ とすると、ドント式ではそれぞれの票は、$190, 192$ となり、もし比例配分すると、$191, 191$ となる。なお実際の比率は $190.501, 191.499$ である。微妙だが、候補者 B を 192 とするのは不合理ではないだろうか。

4.4 相加平均と相乗平均

前の章で投票によって選択肢を評価するときに、ポイントの和を使うか、あるいは積を使うかということが出ていた。このことは、相加平均と相乗平均の比較として扱うこともできる。普通は相加平均と相乗平均をそれぞれに適した分野に使うため、数学的には相加平均 $\geq$ 相乗平均ということが知られているだけである。和を使った時の順位と積を使った時の順位が逆転する場合が重要と考えて、それを調べる。先行研究については色々と調べてみたが今のところ見つかっていない。

定義 2. 正の数字の列 $\{a_i : i = 1, 2, \cdots, n\}$ と $\{b_i : i = 1, 2, \cdots, n\}$ に対して、和と積による評価が逆転するとは次の (1) または (2) が成り立つこととする。

$$\sum_{i=1}^{n} a_i > \sum_{i=1}^{n} b_i \quad \text{かつ} \quad \prod_{i=1}^{n} a_i < \prod_{i=1}^{n} b_i. \tag{1}$$

$$\sum_{i=1}^{n} a_i < \sum_{i=1}^{n} b_i \quad \text{かつ} \quad \prod_{i=1}^{n} a_i > \prod_{i=1}^{n} b_i. \tag{2}$$

補題 1. 数字の列 $\{a_i : i = 1, 2, \cdots, n\}$ と $\{b_i : i = 1, 2, \cdots, n\}$ に対して、和と積による評価が逆転するならば、それぞれから、同じ数字を取り除いても、また、それぞれに同じ数字を加えても和と積による評価が逆転する。

このことは明らかであるので、証明は省く。

和と積が逆転することが起きる条件について研究していくが、a_i, b_i としてどのような数字を使うかによって問題の難易度が変わってくる。

まず、数字として $1, 2, 3$ のみを使う場合を研究する。まず例から始める。

例 8. $\{a_1, a_2, a_3, a_4, a_5, a_6, a_7\} = \{1, 2, 2, 2, 2, 2, 3\}$, $\{b_1, b_2, a_3, b_4, b_5, b_6, b_7\} = \{1, 1, 1, 3, 3, 3, 3\}$ とすると、(2) を満たす。補題 *1* により、両方から同じ数字を取り去ってできる $\{2, 2, 2, 2, 2\}$ と $\{1, 1, 3, 3, 3\}$ も (2) を満たす。　この例の考え方を次の補題で使う。

補題 2. $\{a_1,\cdots,a_n\}$, $\{b_1,\cdots,b_n\}$ が (2) を満たすとすると、両方の数字の組に共通して存在する数字を両方の組から取り除くと、$\{a_1,\cdots,a_n\}$ には *2* だけが残り、$\{b_1,\cdots,b_n\}$ には *1* と *3* が残る。*(1)* を満たす場合は、$\{a_1,\cdots,a_n\}$ と $\{b_1,\cdots,b_n\}$ が逆になる。

証明. 使う数字は $1,2,3$ のみなので、$\{a_i : i=1,2,\cdots,n\}$ と $\{b_i : i=1,2,\cdots,n\}$ の両方から同じ数字を取り去ると、一方に 1 種類の数字のみ、もう一方には 2 種類の数字になる。しかし、数字 1 のみ、数字 3 のみでは逆転は起きないので、可能性としては一方が 2 のみで、他方が 1 と 3 となる。 □

定理 1. $i=1,2,\cdots,n$ に対して、$a_i=2$ で、b_i は 1 か 3 であるとする。
(a) (2) が満たされるとすると、$b_i=3$ となる i の個数を m とすると、m は次の式を満たす。

$$\frac{n\log 2}{\log 3} > m > \frac{n}{2}. \tag{3}$$

(b) *(1)* が成り立つことはない。

証明. (a) (2) により、

$$2n < 3m + (n-m) \quad \text{かつ} \quad 2^n > 3^m \times 1^{n-m}$$

となり、これから

$$n < 2m \quad \text{かつ} \quad n\log 2 > m\log 3$$

となる。これから、(3) を得る。
(b) (1) が成り立つためには、(a) における不等式を逆にしたものが成り立つはずである。しかし、$\frac{\log 2}{\log 3} > \frac{1}{2}$ なので、解は存在しない。

□

定理 2. 数字の列

$$\{a_i : i=1,2,\cdots,n\} \tag{4}$$

と

$$\{b_i : i=1,2,\cdots,n\} \tag{5}$$

に対して、$a_1, b_1 = 1,2,3$ とする。定義 *2* の (2) が満たされるような $1,2,3$ の組み合わせの数は次のようになる。

$$\sum_{x=0}^{n}\sum_{p=0}^{n-x}\sum_{q=0}^{n-p-x}\left(\frac{n!}{p!(q+x)!(n-x-p-q)!}\right)\left(\sum_{y=\lceil\frac{x+1}{2}\rceil}^{\lfloor\frac{x\log 2}{\log 3}\rfloor}\frac{n!}{(p+x-y)!q!(n-x-p-q+y)!}\right). \tag{6}$$

証明. (4) と (5) の両方に含まれる数字を取り去ったあとに、(4) には 2 のみが x 個残り、(5) には 3 が y 個、1 が $x-y$ 個残るとする。すると、扱うのは整数だけなので、定理 1 により、$\lceil\frac{x+1}{2}\rceil \le y \le \lfloor\frac{x\log 2}{\log 3}\rfloor$ となる。ここで、$\lceil\ \rceil$ と $\lfloor\ \rfloor$ はそれぞれ小数点切り上げと切り捨ての関数である。取り去った数字のうちで、1 が p 個、2 が q 個あるとすると、3 は $n-x-p-q$ 個となる。このようにして、x,y,p,q を固定すると、(4) の方には 1,2,3 がそれぞれ $p, q+x, n-x-p-q$ 個あり、(5) の方には 1,2,3 がそれぞれ $p+x-y, q, n-x-p-q+y$ 個ある。x,p,q を固定した状態で (4) において考えることができる数列の個数は $\frac{n!}{p!(q+x)!(n-x-p-q)!}$ である。x,p,q を固定した状態で (5) の方は y を動かすので、$\sum_{\lceil\frac{x+1}{2}\rceil}^{\lfloor\frac{x\log 2}{\log 3}\rfloor}\frac{n!}{(p+x-y)!q!(n-x-p-q+y)!}$ となる。そして、x を $0,1,\cdots,n$ まで動かし、p を $0,\cdots,n-x$, q を $0,\cdots,n-x-p$ とすると、(6) を得る。 □

$$\sum_{i=1}^{n}\binom{n}{k}^2 = \binom{2n}{n} \tag{7}$$

例 9. (6) において、$n=5,6,7,8,9,10,\cdots$ とすると、得られる値は $10, 270, 4606, 63840, 787122,$ $\cdots$ となる。これは (2) が満たされる場合なので、(1) の場合も考えるとこれらの倍になる。このようなことが起きる率は、これらの数を 3^{2n} で割ることによって求められるが、$0.000338702, 0.00101611, 0.001926, 0.00296608$ $, 0.0040634, 0.00517176, 0.00626379, 0.00732411, \cdots$ となり、図 1 のようになる。なお、このグラフは $n=100$ まで計算している。このグラフによれば一定の数に近づいていくようである。しかし私たちはまだ数学的に証明することはできていない。

なお、計算機を使うとしても、式 (6) のようなものが得られると和と積の逆転が起きる場合を全検索で調べるのに比べて、はるかに計算量は少なくなる。ただし、このような公式を作るのは使う数字が多くなればなるほど難しくなる。次には数字 $1,2,3,4$ を使うことを考えるが、公式は複雑になる。

次に、数字として $1,2,3,4$ のみを使う場合を研究する。

補題 3. $i=1,2,\cdots,n$ に対して、$a_i=3$ で、b_i は 1 か 2 か 4 であるとする。
(i) 和と積逆転の式 (1) は成り立たない。
(ii) 和と積逆転の式 (2) を満たすような a_i,b_i がある。

証明. (i) (1) を満たすような b_i があるとする。$\{b_i : i=1,2,\cdots,n\}$ のうちで、4 が p 個、2 が q 個、残りの $n-p-q$ 個が 1 であるとする。すると、 (1) により、次の式が成り立つ。

$$3n > 4p+2q+(n-p-q) \ \text{ かつ } \ 3^n < 4^p 2^q$$

これから、

$$2n > 3p+q \ \text{ かつ } \ n\log 3 < 2p\log 2 + q\log 2$$

したがって、

$$2n > 3p+q \ \text{ かつ } \ \frac{n\log 3}{\log 2} < 2p+q$$

$p+q \le n$ であるから、 次の p,q の不等式が得られるが、グラフに描くことで解がないことがわかる。ここではグラフを省略する。

$$q < -3p+2n \tag{8}$$

$$q > -2p+\frac{n\log 3}{\log 2} \tag{9}$$

$$q \le -p+n \tag{10}$$

(ii) (2) を満たす不等式の表す領域が存在することもグラフを描くことでわかる。 □

補題 3 と同じようなやり方で次のことも証明することができる。補題 4

補題 4. 和と積逆転の式 (2) について、次のような場合は満たすような数字がある。それ以外の $1,2,3,4$ の組み合わせに関しては、逆転が起きることはない。
(a) $i=1,2,\cdots,n$ に対して、$a_i=3$ で、b_i は 1 か 4。
(a) $i=1,2,\cdots,n$ に対して、$a_i=3$ で、b_i は 2 か 4。
(b) $i=1,2,\cdots,n$ に対して、$a_i=2$ で、b_i は 1 か 3 か 4。
(c) $i=1,2,\cdots,n$ に対して、$a_i=2$ で、b_i は 1 か 3。
(d) $i=1,2,\cdots,n$ に対して、$a_i=2$ で、b_i は 1 か 4。
(e) $i=1,2,\cdots,n$ に対して、a_i は 2 か 3 で、b_i は 1 か 4。
(f) $i=1,2,\cdots,n$ に対して、a_i は 2 か 4 で、b_i は 1 か 3。

例 10. 補題 *3* と補題 *4* を使いながら、逆転が起きるような場合を検索してから、数値計算することで、和と積の逆転が起こる確率は、$n3,4,5,6,7,8,9,\cdots$ のとき、次のようになる $0.00439453, 0.00854492, 0.012064, 0.0150293, 0.0175496, 0.0197169, 0.0216037, \cdots$ この数値をグラフにすると図 *2* のようになる。グラフの傾きが減っていき、一定の数に収束することが予想される。$n=50$ まで計算できたのは、数学の補題をうまく活用したからである。全検索ではこのような大きな値の n に関して計算することはできない。なお補題を使った計算と C 言語を使った計算は $n=9$ までは一致した。

ここでは、数字 $1,2,3,4,5$ を使うケースを調べる。この場合にはまだ何も公式を作ることができていないので、計算機による全検索の結果を書く。

例 11. 使う数字が $1,2,3,4,5$ となると、和と積の逆転が起こる確率は、n の大きさによって次のようになる $n=2,3,4,5,6,7,8,9$ に対して、$0.0128, 0.02112, 0.0265421, 0.0306647, 0.0340243, 0.0368301, 0.0391996, 0.041218$ となる。なお、$n=9$ の場合は C 言語で *13671 秒 (Macbook Pro M1)* を要した。この数値をグラフにすると図 *3* のようになる。

ここでは、a_i, b_i が実数値の場合を考える。

例 12. $x+y<b_1+b_2$ かつ $xy>b_1b_2$ を満たすような実数の場合はどうなるかを考えた。a_1, a_2 の代わりに x, y を使うのは、グラフ的に理解しやすいからである。ここでは数字を 4 つしか使わないが、もっと多くの数字を使うとかなり複雑な積分になるので、まだ手をつけていない。なお、数字の大きさに制限をつけないと割合を考えることはできないので、0 以上で 10 以下の数字に限定して考える。
(i) 積分を使う方法
不等式 $y<-x+b_1+b_2$ と $y>\frac{b_1,b_2}{x}$ を満たす 10 以下の正の数 x, y, b_1, b_2 がどのくらいあるかを調べる。まず、b_1, b_2 を固定するとき、不等式を満たす x, y の領域の面積は $b_1 \leq b_2$ なら $\int_{b_1}^{b_2}(-x+b_1b_2-\frac{b_1b_2}{x})dx$ となり、$b_1<b_2$ なら積分範囲が逆になる。このようにしておいてから、b_1, b_2 をそれぞれ 1 から 10 まで積分すればよい。すると、次のような重積分を求めれば良いことになる。$\int_0^{10}\int_0^{b_2}\int_{b_1}^{b_2}(-x+b_1b_2-\frac{b_1b_2}{x})dxdb_1db_2$ $+\int_0^{10}\int_{b_2}^{10}\int_{b_2}^{b_1}(-x+b_1b_2-\frac{b_1b_2}{x})dxdb_1db_2$ これは *Mathematica* による数値積分で求めたところ、416.334 となった。各変数が 0 から 10 なので、割合としては 10000 で割って、0.0416334 となる。なお、これは逆転となる場合の半分なので、2 倍して、0.0832668 となる。
(ii) 同じことを *Mathematica* の乱数で計算してみた。10000000 回の実験で 416920 回起きたので、割合は 0.041692 となる。2 倍して、0.083384 となる。積分の値にかなり近い。
(iii) 次に *Python* で計算してみた。実数の確率を計算することは不可能に近い。そのため、今回は 0〜10000 の乱数を生成して、1000 で割り、この方法で、0〜10 の範囲で小数第 *3* 位までの乱数を再現した。この方法を使用した理由は *3* つある。
1. 全通り調べるのには途方もない時間がかかる。
2. 浮動小数点数の乱数では計算時間がかかりすぎた。
3. 小数第 *4* 位、*5* 位の計算結果と比較しても誤差の範囲内であった。

また、乱数を使用しているため、計算時間は数秒程度である。
結果 $n=2,3,4,5,6,7,8,9,10,11$ $0.083938, 0.110072, 0.124098, 0.13201, 0.13754, 0.142798, 0.145638,$ $0.148022, 0.150532, 0.15144.$ この数値をグラフにすると図 4 のようになる。グラフは $n=500$ まで計算したが、このように整数で乱数を作って計算しても、*Python* で *779* 秒かかった。n が大きくなると、ほぼ一定になっている。

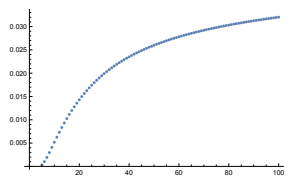

図 1: $1, 2, 3$

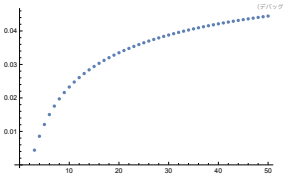

図 2: $1, 2, 3, 4$

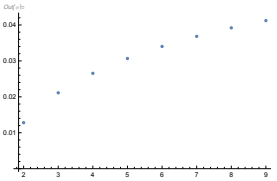

図 3: $1, 2, 3, 4, 5$

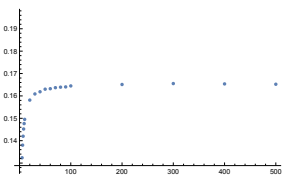

図 4: 実数

5　結論と今後の課題及び感想

5.1　結論

比例代表制に適したドント方式が、今回の総裁選の 382 とか、あるいはもっと多い数の議員票の分配に使われるのが適切とは限らない。自民党の総裁選挙については、ドント方式を使うことで不公平と考えられる例をいくつも発見したので、比例配分方式を使うべきだと考える。またドント方式という多くの人にとって未知な方法を使うメリットを使うことは、どうしても必要でない限りやめるべきである。なお、ドント式の計算においてプログラムミスがあってはいけないので、Python と Mathematica という全く違う言語でそれぞれ別の人が作り、結果が合うことを確かめた。

次に和による評価と積による評価の逆転という問題は、私達が見つけた数学的に研究する価値のある問題のようである。

5.2　今後の課題

投票において、ポイントの積を使うことは、すでに、例 4 と例 5 でわかったように、他の評価方法とかなり違った状況を作り出す。違っているから良くないとは言えない。どんな特徴があるのかを詳細に調べる必要がある。

次に和による評価と積による評価の逆転という問題は、数字の列の長さ n を増やした時、率が単調に一定の極限に近づくという予想が出ているが、これを証明したい。また、和の代わりに、積を使うということは、各ポイントに log を使ってから和を求めて比べることと同じになる。このことを考えると、単にポイントを足すのではなく、何かの関数を使ってから和を求めるという形で一般化することも可能である。このことも考えてみたい。

選挙方法の理論としては Arrow の不可能性定理が良く知られている。私達は選択肢が 3 で、投票者 3 名の場合は Arrow の不可能性定理を Python による並列プログラムで検証できた。これは別紙のプログラム 11 にある。文献 [4] では投票者 2 名で計算機で検証しておいて、数学的帰納法によって Arrow の定理の別証明しているが、私達は選択肢 3 で、投票者 3 名の場合なら、Arrow の条件を少し変えながら、結果がどう変わるかリアルタイムで調べることができる環境を作ることができた。

5.3　感想

小学校以来のことを思い出すと、多くの決定が決選投票方式で行われてきたように思う。何人かの先生達に聞いて見ると学校での会議でもほとんどはそうやって決めているそうだ。もちろん、手を挙げて行う場合は一番楽にできる方法ではある。ポイント制などは計算が面倒ということはあっただろう。ただ、多くの人がスマホを持っていて、投票をアプリ化したらポイント制もすぐにできる。そういうことも考える時代に来ているのではないか。また、学校で習う数学の中に、投票理論を含めるべきではないだろうか。そして、シミュレーション結果なども教えて、投票方法についての知識を国民が持つべきではないだろうか。そうでないと、知識を持つ一部の人達によって制度を作られてしまって、他の人はうまく牛耳られるかもしれない。

参考文献

[1] 佐伯胖, 「きめ方」の論理　ちくま学芸文庫　2018

[2] 坂井豊貴, 社会的選択理論への招待, 日本評論社 2018

[3] Mark B.Garman and Morton I. Kamien, THE PARADOX OF VOTING:PROBABILITY CALCULATIONS, Behavioral Science, Vol.13, 1968.

[4] Pingzhong Tang and Fangzhen Lin,Computer-aided proofs of Arrow ’s and other impossibility theorems , Artificial Intelligence 173 ,2009.

6　付録　プログラミングのコード

啓明学院中学校高等学校　中学 3 年: 眞部光, 高校 1 年: 青野大志, 高校 1 年: 高橋 祥英, 高校 3 年: 立花優季
コンテストの原稿の 8 ページの原稿において使われたコンピュータプログラムをここに書きます。

プログラム 1. 例 *1* に使った *Mathematica* のプログラミングコード

投票シミュレーション

順位付け関数

```
from itertools import groupby
import numpy as np
def juni(list1):
    list1=np.array(list1)
    indexList = list1.argsort()[::-1].tolist()
    list1.sort()
    list1 = list1[::-1]
    result = []
    cursor = 0
    for (_, group) in groupby(list1.tolist()):
        size = len(list(group))
        result.append(indexList[cursor:cursor + size])
        cursor += size
    return result
```

ポイント制 (和)

```
import numpy as np
import jyuni as jy
def votep(x):
    a0=np.zeros_like(x[0])
    for i in range(len(x)):
        for j in range(len(x[0])):
            a0[j]+=x[i].index(j)
    return(sorted(jy.juni(a0)[-1]))
```

ポイント制 (積)

```
import numpy as np
import jyuni as jy
def votes(x):
    a0=np.ones_like(x[0])
    for i in range(len(x)):
        for j in range(len(x[0])):
            a0[j]*=(x[i].index(j)+1)
    return(jy.juni(a0)[-1])
```

上位二人による決選投票

```
import jyuni as jy
```

```
def votek(x):
    def _votek(y):
        a0=[]
        for i in range(len(y)):
            a0.append(y[i][0])
        a1=[]
        for j in sorted(y[0]):
            a1.append(a0.count(j))
        a2=jy.juni(a1)
        return(a2)
    b0=_votek(x)
    if len(b0[0])==1:
        b1=b0[0]+b0[1]
    elif len(b0[0])>=2:
        b1=b0[0]
    b2=[]
    for i in x:
        b2.append(list(filter(lambda y: y in b1,i)))
    b3=[]
    for i in _votek(b2)[0]:
        b3.append(sorted(b1)[i])
    return(b3)
```

投票結果作成関数

```
import random
def vote_maker(n,m):#n 個選択肢 m 人分の投票結果を random 生成
    ans=[]
    for i in range(m):
        ans.append(random.sample(list(range(n)),n))
    return(ans)
```

積和

総積関数

```
def prod(x):
    a=1
    for i in x:
        a*=i
    return(a)
```

総和関数

```
def summ(x):
    a=0
    for i in x:
        a+=i
    return(a)
```

積和逆転重複組み合わせ組捜索関数（個数出力）

```
def sekiwajk(x,y):
    t=0
    l1=list(range(1,y+1))
    all=list(itertools.combinations_with_replacement(l1,x))
    for u in range(len(all)):
        for v in range(len(all)):
            if  summ(all[u])<summ(all[v]) and prod(all[u])>prod(all[v]):
                t+=1
    return(t)
```

積和逆転重複順列組捜索関数（個数出力）

```
def sekiwajj(x,y):
    t=0
    l1=list(range(1,y+1))
    all=list(itertools.product(l1,repeat=2*x))
    for v in all:
        w1=v[:x]
        w2=v[x:2*x]
        if summ(w1)<summ(w2) and prod(w1)>prod(w2):
            t+=1
    return(t)
```

和によるポイント制と上位二人による決選投票制の比較

```
import random
from numpy import left_shift
import votep as vp
import votek as vk
Import vortemaker as vm

#和によるポイント制と決選投票の比較

def vhikakupk(n,m,k):#n 個選択肢 m 人 k 回数
    b1=b2=b3=b4=b5=b6=b7=b8=b9=0
    c1=c2=c3=c4=c5=c6=c7=c8=c9=0
    d1=d2=d3=d4=d5=d6=d7=d8=d9=0
    for j in range(k):
       a1=vm.vote_maker(n,m)
        pp=vp.votep(a1)
        kk=vk.votek(a1)
        if len(pp) and len(kk)==1:
            b1+=1
            if pp==kk:
                b2+=1
            elif pp!=kk:
                b3+=1
            else:
```

```
            print("Error!!")
            print(pp,kk)
            break
    elif len(pp)==1 and len(kk)!=1:
        b5+=1
        if pp in kk:
            b6+=1
        elif pp not in kk:
            b7+=1
        else:
            print("Error!!")
            print(pp,kk)
            break
    elif len(pp)!=1 and len(kk)==1:
        b8+=1
        if kk in pp:
            b6+=1
        elif kk not in pp:
            b7+=1
        else:
            print("Error!!")
            print(pp,kk)
            break
    elif (len(pp) and len(kk)!=1) and len(pp)>len(kk):
        c1+=1
        if kk in pp:
            c3+=1
        elif kk not in pp:
            c4+=1
        else:
            print("Error!!")
            print(pp,kk)
            break
    elif (len(pp) and len(kk)!=1) and len(pp)<len(kk):
        c5+=1
        if pp in kk:
            c3+=1
        elif pp not in kk:
            c4+=1
        else:
            print("Error!!")
            print(pp,kk)
            break
    elif (len(pp) and len(kk)!=1) and len(pp)==len(kk):
        c6+=1
        if pp in kk:
            c3+=1
        elif pp not in kk:
            c4+=1
```

```
            else:
                print("Error!!")
                print(pp,kk)
                break
        else:
            print("!!!Error!!!")
            print(pp,kk)
            break

    print(“和によるポイント制決選投票共に一人のみ決まる場合---------------------->”,b1/k)
    print(“和によるポイント制決選投票共に一人で結果が一致する場合―――――――->”,b2/k)
    print(“和によるポイント制決選投票共に一人で結果が異なる場合------------------>”,b3/k)
    print(“和によるポイント制決選投票の少なくともどちらかが一人でない場合―
                                                                    ――>
”,(b5+b8+c1+c5+c6)/k)
    print(“和によるポイント制が一人で決選投票が一人でない場合---------------->”,b5/k)
    print(“和によるポイント制が一人でなく決選投票が一人の場合---------------->”,b8/k)
    print("和によるポイント制決選投票のどちらかが一人でなく一致がある場合--->",b6/k)
    print("和によるポイント制決選投票のどちらかが一人でなく一致がない場合--->",b7/k)
    print(“和によるポイント制決選投票共に一人でない場合---------------------->”,(c1+c5+c6)/k)
    print(“和によるポイント制決選投票共に一人でなくポイント制の方が多い場合->",c1/k)
    print("和によるポイント制決選投票共に一人でなく決選投票の方が多い場合--->",c5/k)
    print(“和によるポイント制決選投票共に一人でなく両者等しい場合―――――――>”,c6/k)
    print(“和によるポイント制決選投票共に一人でなく一致がある場合―――――――>",c3/k)
    print(“和によるポイント制決選投票共に一人でなく一致がない場合―――――――>",c4/k)
    return

print(vhikakupk(5,42,100000))
```

出力データ↓

```
# 和によるポイント制決選投票共に一人のみ決まる場合------------------> 0.85301
# 和によるポイント制決選投票共に一人で結果が一致する場合-----------> 0.58232
# 和によるポイント制決選投票共に一人で結果が異なる場合-------------> 0.27069
# 和によるポイント制決選投票の少なくともどちらかが一人でない場合---> 0.14699
# 和によるポイント制が一人で決選投票が一人でない場合---------------> 0.13593
# 和によるポイント制が一人でなく決選投票が一人の場合---------------> 0.0
# 和によるポイント制決選投票のどちらかが一人でなく一致がある場合---> 0.0
# 和によるポイント制決選投票のどちらかが一人でなく一致がない場合---> 0.13593
# 和によるポイント制決選投票共に一人でない場合----------------------> 0.01106
# 和によるポイント制決選投票共に一人でなくポイント制の方が多い場合-> 0.00047
# 和によるポイント制決選投票共に一人でなく決選投票の方が多い場合---> 0.00082
# 和によるポイント制決選投票共に一人でなく両者等しい場合-----------> 0.00977
# 和によるポイント制決選投票共に一人でなく一致がある場合-----------> 0.0
# 和によるポイント制決選投票共に一人でなく一致がない場合-----------> 0.01106
```

この他に、積によるポイント制と決選投票制を比較する vhikakusk、和によるポイント制と積によるポイント制を比較する vhikakusp などがある。

実際に記載しているデータの出所
↓

```
import votes as vs
import votep as vp
import votek as vk
import votemaker as vm
```

和、決選投票比較

```
b1=b2=b3=b4=b5=0
k=10000
for j in range(k):
    a1=vm.vote_maker(5,42)
    pp=vp.votep(a1)
    kk=vk.votek(a1)
    if len(pp) and len(kk)==1:
        b1+=1
        if pp==kk:
            b2+=1
        elif pp!=kk:
            b3+=1
    elif (len(pp) or len(kk))!=0:
        b4+=1
        if len(pp)!=1 and len(kk)!=0:
            b5+=1
print("和によるポイント制決選投票共に一人のみ決まる場合----------------->",b1/k)
print("和によるポイント制決選投票共に一人で結果が一致する場合----------->",b2/k)
print("和によるポイント制決選投票共に一人で結果が異なる場合------------->",b3/k)
print("和によるポイント制決選投票の少なくともどちらかが一人でない場合--->",b4/k)
print("和によるポイント制決選投票共に一人でない場合---------------------->",b5/k)
```

積、決選投票比較

```
b1=b2=b3=b4=b5=0
k=10000
for j in range(k):
    a1=vm.vote_maker(5,42)
    ss=vs.votes(a1)
    kk=vk.votek(a1)
    if len(ss) and len(kk)==1:
        b1+=1
        if ss==kk:
            b2+=1
        elif ss!=kk:
            b3+=1
    elif (len(ss) or len(kk))!=0:
        b4+=1
        if len(ss)!=1 and len(kk)!=0:
            b5+=1
print("積によるポイント制決選投票共に一人のみ決まる場合----------------->",b1/k)
print("積によるポイント制決選投票共に一人で結果が一致する場合----------->",b2/k)
print("積によるポイント制決選投票共に一人で結果が異なる場合------------->",b3/k)
```

```
print("積によるポイント制決選投票の少なくともどちらかが一人でない場合--->",b4/k)
print("積によるポイント制決選投票共に一人でない場合---------------------->",b5/k)
```

積、和比較

```
b1=b2=b3=b4=b5=0
k=10000
for i in range(k):
    a1=vm.vote_maker(5,42)
    pp=vp.votep(a1)
    ss=vs.votes(a1)
    if len(pp) and len(ss)==1:
        b1+=1
        if pp==ss:
            b2+=1
        elif pp!=ss:
            b3+=1
    elif (len(pp) or len(ss))!=0:
        b4+=1
        if len(pp)!=1 and len(ss)!=0:
            b5+=1
print("積によるポイント制和によるポイント制に一人のみ決まる場合------------------>",b1/k)
print("積によるポイント制和によるポイント制共に一人で結果が一致する場合------------>",b2/k)
print("積によるポイント制和によるポイント制共に一人で結果が異なる場合-------------->",b3/k)
print("積によるポイント制和によるポイント制の少なくともどちらかが一人でない場合--->",b4/k)
print("積によるポイント制和によるポイント制共に一人でない場合---------------------->",b5/k)
```

プログラム 2. 例 *6* に使った *Mathematica* のプログラミングコード

```
[caption=hoge,label=fuga]
allloopdata = {}; allloopdata0 = {};
con = 0; Do[Clear[data]; dd = {1, 2, 3, 4, 5}; nn = 42;(**nn は人数**)

 data = Table[RandomSample[dd], {n, 1, nn}];
 dat[1, 2] =
  Map[{Position[#, 1][[1, 1]], Position[#, 2][[1, 1]]} &, data];
 dat[1, 3] =
  Map[{Position[#, 1][[1, 1]], Position[#, 3][[1, 1]]} &, data];
 dat[1, 4] =
  Map[{Position[#, 1][[1, 1]], Position[#, 4][[1, 1]]} &, data];
 dat[1, 5] =
  Map[{Position[#, 1][[1, 1]], Position[#, 5][[1, 1]]} &, data];
 dat[2, 1] =
  Map[{Position[#, 2][[1, 1]], Position[#, 1][[1, 1]]} &, data];
 dat[2, 3] =
  Map[{Position[#, 2][[1, 1]], Position[#, 3][[1, 1]]} &, data];
 dat[2, 4] =
  Map[{Position[#, 2][[1, 1]], Position[#, 4][[1, 1]]} &, data];
 dat[2, 5] =
  Map[{Position[#, 2][[1, 1]], Position[#, 5][[1, 1]]} &, data];
```

```
dat[3, 1] =
 Map[{Position[#, 3][[1, 1]], Position[#, 1][[1, 1]]} &, data];
dat[3, 2] =
 Map[{Position[#, 3][[1, 1]], Position[#, 2][[1, 1]]} &, data];
dat[3, 4] =
 Map[{Position[#, 3][[1, 1]], Position[#, 4][[1, 1]]} &, data];
dat[3, 5] =
 Map[{Position[#, 3][[1, 1]], Position[#, 5][[1, 1]]} &, data];
dat[4, 1] =
 Map[{Position[#, 4][[1, 1]], Position[#, 1][[1, 1]]} &, data];
dat[4, 2] =
 Map[{Position[#, 4][[1, 1]], Position[#, 2][[1, 1]]} &, data];
dat[4, 3] =
 Map[{Position[#, 4][[1, 1]], Position[#, 3][[1, 1]]} &, data];
dat[4, 5] =
 Map[{Position[#, 4][[1, 1]], Position[#, 5][[1, 1]]} &, data];
dat[5, 1] =
 Map[{Position[#, 5][[1, 1]], Position[#, 1][[1, 1]]} &, data];
dat[5, 2] =
 Map[{Position[#, 5][[1, 1]], Position[#, 2][[1, 1]]} &, data];
dat[5, 3] =
 Map[{Position[#, 5][[1, 1]], Position[#, 3][[1, 1]]} &, data];
dat[5, 4] =
 Map[{Position[#, 5][[1, 1]], Position[#, 4][[1, 1]]} &, data];
datb[1, 2] = Select[dat[1, 2], #[[1]] < #[[2]] &];
datb[1, 3] = Select[dat[1, 3], #[[1]] < #[[2]] &];
datb[1, 4] = Select[dat[1, 4], #[[1]] < #[[2]] &];
datb[1, 5] = Select[dat[1, 5], #[[1]] < #[[2]] &];
datb[2, 1] = Select[dat[2, 1], #[[1]] < #[[2]] &];
datb[2, 3] = Select[dat[2, 3], #[[1]] < #[[2]] &];
datb[2, 4] = Select[dat[2, 4], #[[1]] < #[[2]] &];
datb[2, 5] = Select[dat[2, 5], #[[1]] < #[[2]] &];
datb[3, 1] = Select[dat[3, 1], #[[1]] < #[[2]] &];
datb[3, 2] = Select[dat[3, 2], #[[1]] < #[[2]] &];
datb[3, 4] = Select[dat[3, 4], #[[1]] < #[[2]] &];
datb[3, 5] = Select[dat[3, 5], #[[1]] < #[[2]] &];
datb[4, 1] = Select[dat[4, 1], #[[1]] < #[[2]] &];
datb[4, 2] = Select[dat[4, 2], #[[1]] < #[[2]] &];
datb[4, 3] = Select[dat[4, 3], #[[1]] < #[[2]] &];
datb[4, 5] = Select[dat[4, 5], #[[1]] < #[[2]] &];
datb[5, 1] = Select[dat[5, 1], #[[1]] < #[[2]] &];
datb[5, 2] = Select[dat[5, 2], #[[1]] < #[[2]] &];
datb[5, 3] = Select[dat[5, 3], #[[1]] < #[[2]] &];
datb[5, 4] = Select[dat[5, 4], #[[1]] < #[[2]] &];
Clear[conwin];
conwin = {};
If[Length[datb[1, 2]] > nn/2 && Length[datb[1, 3]] > nn/2 &&
  Length[datb[1, 4]] > nn/2 && Length[datb[1, 5]] > nn/2,
 conwin = Append[conwin, 1],];
```

```
 If[Length[datb[2, 1]] > nn/2 && Length[datb[2, 3]] > nn/2 &&
   Length[datb[2, 4]] > nn/2 && Length[datb[2, 5]] > nn/2,
  conwin = Append[conwin, 2],];
 If[Length[datb[3, 2]] > nn/2 && Length[datb[3, 1]] > nn/2 &&
   Length[datb[3, 4]] > nn/2 && Length[datb[3, 5]] > nn/2,
  conwin = Append[conwin, 3],];
 If[Length[datb[4, 2]] > nn/2 && Length[datb[4, 3]] > nn/2 &&
   Length[datb[4, 1]] > nn/2 && Length[datb[4, 5]] > nn/2,
  conwin = Append[conwin, 4],];
 If[Length[datb[5, 2]] > nn/2 && Length[datb[5, 3]] > nn/2 &&
   Length[datb[5, 4]] > nn/2 && Length[datb[5, 1]] > nn/2,
  conwin = Append[conwin, 5],];

 ddd0 = Subsets[{1, 2, 3, 4, 5}, {3}];
 threeloop = {};
 Do[sub = ddd0[[kk]];
  ddd = {sub, Reverse[sub]};
  Do[Clear[loop]; loop = ddd[[k]];
  If[((Length[datb[loop[[1]], loop[[2]]]] >
nn/2) && (Length[datb[loop[[2]], loop[[3]]]] >
nn/2) && (Length[datb[loop[[3]], loop[[1]]]] > nn/2)),
threeloop = Append[threeloop, loop],], {k, 1, 2}], {kk, 1,
Length[ddd0]}];ddd1 = Subsets[{1, 2, 3, 4, 5}, {4}];
fourloop = {};
 Do[Clear[loop0, loop]; loop0 = ddd1[[k]];
  loopset = Select[Permutations[loop0], #[[1]] == Min[#] &];
  Do[loop = loopset[[kk]];
If[((Length[datb[loop[[1]],loop[[2]]]]>nn/2)&&(Length[datb[loop[[2]],loop[[3]]]]>
nn/2) && (Length[datb[loop[[3]], loop[[4]]]] >
nn/2) && (Length[datb[loop[[4]], loop[[1]]]] > nn/2)),
fourloop = Append[fourloop, loop],], {kk, 1,
Length[loopset]}], {k, 1, Length[ddd1]}];
ddd2 = Select[Permutations[{1, 2, 3, 4, 5}], #[[1]] == Min[#] &];
fiveloop = {};
Do[Clear[loop]; loop = ddd2[[k]];
If[((Length[datb[loop[[1]], loop[[2]]]] >
nn/2) && (Length[datb[loop[[2]], loop[[3]]]] >
nn/2) && (Length[datb[loop[[3]], loop[[4]]]] >
nn/2) && (Length[datb[loop[[4]], loop[[5]]]] >
nn/2) && (Length[datb[loop[[5]], loop[[1]]]] > nn/2)),
fiveloop = Append[fiveloop, loop],], {k, 1, Length[ddd2]}];
allloop = Join[threeloop, fourloop, fiveloop, conwin];
allloopdata0 = Append[allloopdata0, {threeloop, fourloop, fiveloop, conwin}];
allloopdata = Append[allloopdata, {Length[threeloop], Length[fourloop],
Length[fiveloop], Length[conwin]}]
, {nnn, 1, 10000}];
```

プログラム 3. 例 7において自民党総裁選で使われたドント方式の計算をするプログラムを *Python* 用と *Mathematica* 用。

ドント方式　(Python)

```
import numpy as np
import jyuni as jy
def dondt(x,n):
    a=np.array(x) # 得票数
    h=np.zeros_like(a)  # 配分（最初はすべて 0）
    for _ in range(n):
        b=a/(h+1)
        h[random.choice(jy.juni(b)[0])]+=1
    return(list(h))
```

ドント方式　Mathematica

以下の関数はドント式と最大剰余を使う比例配分の両方を出力し、小数まで出した比例と比べる。

```
dont[x_] := Block[{a},
data = Table[{a[1, t], a[2, t], a[3, t], a[4, t]} = {x[[1]]/t,
x[[2]]/t, x[[3]]/t, x[[4]]/t}, {t, 1, 400}];
data2 = Sort[Flatten[data], Greater];
la = data2[[382]];ind = Flatten[Table[{i, t}, {i, 1, 4}, {t, 1, 400}], 1];
ca = Select[ind, a[#[[1]], #[[2]]] >= la &];
d = Map[#[[1]] &, ca];
aa0 = {Count[d, 1], Count[d, 2], Count[d, 3], Count[d, 4]};
aas = Apply[Plus, aa0]; aa = aa0/aas*382;
bb = x/Apply[Plus, x]*382; bb2 = Map[Floor[#] &, bb];
sb = 382 - Apply[Plus, bb2];sb2 = 4 - sb; bb3 = bb - bb2;
bb4 = Sort[bb3, Greater];
fff[x0_] := Block[{xx = x0},
Which[xx == bb4[[1]], Ceiling[Max[0, 4 - sb2]/10], xx == bb4[[2]],
Ceiling[Max[0, 3 - sb2]/10], xx == bb4[[3]],
Ceiling[Max[0, 2 - sb2]/10], xx == bb4[[4]], 0]];
bbb = Map[fff[#] &, bb3] + bb2;{aa, bbb, bb // N}]
```

プログラム 4. 例 7において自民党総裁選で使われたドント方式の問題点を見つけるための使ったプログラム。

```
import random
t=1000#試行回数
n=4#分割する個数
p=2#出てくる差
q=[]
for _ in range(t):
    b=[]
    w=1104336
    for i in range(n-1):
        a=random.choice(list(range(w)))
        b.append(a)
        w=w-a
    b.append(w)
    dondt=dv.dondt(b,382)
    hirei=hv.hirei(b,382)
    if dondt!=hirei:
```

```
        for j in range(n):
            if abs(dondt[j]-hirei[j])==p:
                q.append(b)
for i in range(len(q)):
    w=q[i]
    r=[]
    for j in w:
        r.append(len(list(str(j))))
    if r.count(6)==4:
        print(w)
```

プログラム 5. 例 *9*で行なった計算の *Mathematica* プログラムは次のようになる。

図 7.
$$\texttt{dn}[n\_] := \sum_{x=5}^{n}\sum_{p=0}^{n-x}\sum_{q=0}^{n-x-p}\left(n!/((p!)*((q+x)!)*((n-x-p-q)!))*\left(\sum_{y=\texttt{Ceiling}[(x+1)/2]}^{\texttt{Floor}[x*\texttt{Log}[2]/\texttt{Log}[3]]}(n!/((q!)*((p+x-y)!)*(n-x-p-q+y)!))\right)\right);$$

プログラム 6. 例 *10* のための *Mathematica* プログラム

```
Clear[d3vs14, d3vs124, d2vs134, d2vs13, d2vs14, d23vs14]; Do[
Clear[data, data2, data3, data4, data0, data02, data03];
(**3 vs 1,4**)
data = IntegerPartitions[x, {2}];
data2 = Map[Permutations[#] &, data];
data3 = Flatten[Map[Permutations[#] &, data], 1];
d3vs14[x] =
Select[data3,
3^x > 1^#[[1]]*4^#[[2]] && 3 x < #[[1]] + 4 #[[2]] &];
Clear[data, data2, data3, data4, data0, data02, data03];
(**3 vs 1,2,4**)
data = IntegerPartitions[x, {3}];
data2 = Map[Permutations[#] &, data];
data3 = Flatten[Map[Permutations[#] &, data], 1];
d3vs124[x] =
Select[data3,
3^x > 1^#[[1]]*2^#[[2]]*4^#[[3]] &&
3 x < #[[1]] + 2 #[[2]] + 4 #[[3]] &];
 Clear[data, data2, data3, data4, data0, data02, data03];
 (**2 vs 1,3,4**)
 data = IntegerPartitions[x, {3}];
 data2 = Map[Permutations[#] &, data];
 data3 = Flatten[Map[Permutations[#] &, data], 1];
 d2vs134[x] =
  Select[data3,
   2^x > 1^#[[1]]*3^#[[2]]*4^#[[3]] &&
     2 x < #[[1]] + 3 #[[2]] + 4 #[[3]] &];
 Clear[data, data2, data3, data4, data0, data02, data03];
 (**2 vs 1,3**)
 data = IntegerPartitions[x, {2}];
 data2 = Map[Permutations[#] &, data];
 data3 = Flatten[Map[Permutations[#] &, data], 1];
 d2vs13[x] =
```

```
 Select[data3,
  2^x > 1^#[[1]]*3^#[[2]] && 2 x < #[[1]] + 3 #[[2]] &];
Clear[data, data2, data3, data4, data0, data02, data03];
(**2 vs 1,4**)
data = IntegerPartitions[x, {2}];
data2 = Map[Permutations[#] &, data];
data3 = Flatten[Map[Permutations[#] &, data], 1];
d2vs14[x] =
 Select[data3,
  2^x > 1^#[[1]]*4^#[[2]] && 2 x < #[[1]] + 4 #[[2]] &];
Clear[data, data2, data3, data4, data0, data02, data03];
(**3 vs 2,4**)
data = IntegerPartitions[x, {2}];
data2 = Map[Permutations[#] &, data];
data3 = Flatten[Map[Permutations[#] &, data], 1];
d3vs24[x] =
Select[data3,
3^x > 2^#[[1]]*4^#[[2]] && 3 x < 2 #[[1]] + 4 #[[2]] &];
Clear[data, data2, data3, data4, data0, data02, data03];
(**2,3 vs 1,4**)
data = IntegerPartitions[x, {2}];
data2 = Map[Permutations[#] &, data];
data3 = Flatten[Map[Permutations[#] &, data], 1];
data0 = IntegerPartitions[x, {2}];
data02 = Map[Permutations[#] &, data];
data03 = Flatten[Map[Permutations[#] &, data], 1];
data4 = Flatten[
Table[Join[data3[[k]],data03[[h]]],{k,1,Length[data3]},{h,1,Length[data03]}],1];
d23vs14[x] =
Select[data4,
2^#[[1]]*3^#[[2]]>1^#[[3]]*4^#[[4]] && 2#[[1]]+3#[[2]] < #[[3]]+ 4#[[4]] &];
Clear[data, data2, data3, data4, data0, data02, data03];
(**2,4 vs 1,3**)
data = IntegerPartitions[x, {2}];
data2 = Map[Permutations[#] &, data];
data3 = Flatten[Map[Permutations[#] &, data], 1];
data0 = IntegerPartitions[x, {2}];
data02 = Map[Permutations[#] &, data];
data03 = Flatten[Map[Permutations[#] &, data], 1];
data4 = Flatten[
Table[Join[data3[[k]],data03[[h]]],{k,1,Length[data3]},{h,1,Length[data03]}],1];
d24vs13[x]=Select[data4,2^#[[1]]*4^#[[2]] > 1^#[[3]]*3^#[[4]] &&
2 #[[1]] + 4 #[[2]] < #[[3]] + 3 #[[4]] &];
Print[{d3vs14[x], d3vs24[x], d3vs124[x], d2vs134[x], d2vs13[x],
d2vs14[x], d23vs14[x], d24vs13[x]}, {x, 2, 9}]
```

プログラム 7. 例 *10* のための *Mathematica* プログラムの続き。テキスト形式でないので画像ではる。

バッグ) In[∘]:=

$$n = 9;\ \left(\sum_{x=3}^{n}\sum_{p=0}^{n-x}\sum_{q=0}^{n-x-p}\sum_{r=0}^{n-x-p-q}(n!/(((p)!)*((q)!)*((r+x)!)*((n-x-p-q-r)!)))*\mathrm{If}\Big[\mathrm{Length}[d3vs14[x]]>0,\ \sum_{k=1}^{\mathrm{Length}[d3vs14[x]]}(n!/(((p+d3vs14[x][[k,1]])!)*((q)!)*((r)!)*((n-x-p-q-r+d3vs14[x][[k,2]])!))),0\Big]\right)+$$

$$\left(\sum_{x=3}^{n}\sum_{p=0}^{n-x}\sum_{q=0}^{n-x-p}\sum_{r=0}^{n-x-p-q}(n!/(((p)!)*((q)!)*((r+x)!)*((n-x-p-q-r)!)))*\mathrm{If}\Big[\mathrm{Length}[d3vs24[x]]>0,\ \sum_{k=1}^{\mathrm{Length}[d3vs24[x]]}(n!/(((p)!)*((q+d3vs24[x][[k,1]])!)*((r)!)*((n-x-p-q-r+d3vs24[x][[k,2]])!))),0\Big]\right)+$$

$$\left(\sum_{x=3}^{n}\sum_{p=0}^{n-x}\sum_{q=0}^{n-x-p}\sum_{r=0}^{n-x-p-q}(n!/(((p)!)*((q)!)*((r+x)!)*((n-x-p-q-r)!)))*\mathrm{If}\Big[\mathrm{Length}[d3vs124[x]]>0,\ \sum_{k=1}^{\mathrm{Length}[d3vs124[x]]}(n!/(((p+d3vs124[x][[k,1]])!)*((q+d3vs124[x][[k,2]])!)*((r)!)*((n-x-p-q-r+d3vs124[x][[k,3]])!))),0\Big]\right)+$$

$$\left(\sum_{x=3}^{n}\sum_{p=0}^{n-x}\sum_{q=0}^{n-x-p}\sum_{r=0}^{n-x-p-q}(n!/(((p)!)*((q+x)!)*((r)!)*((n-x-p-q-r)!)))*\mathrm{If}\Big[\mathrm{Length}[d2vs134[x]]>0,\ \sum_{k=1}^{\mathrm{Length}[d2vs134[x]]}(n!/(((p+d2vs134[x][[k,1]])!)*((q)!)*((r+d2vs134[x][[k,2]])!)*((n-x-p-q-r+d2vs134[x][[k,3]])!))),0\Big]\right)+$$

$$\left(\sum_{x=3}^{n}\sum_{p=0}^{n-x}\sum_{q=0}^{n-x-p}\sum_{r=0}^{n-x-p-q}(n!/(((p)!)*((q+x)!)*((r)!)*((n-x-p-q-r)!)))*\mathrm{If}\Big[\mathrm{Length}[d2vs13[x]]>0,\ \sum_{k=1}^{\mathrm{Length}[d2vs13[x]]}(n!/(((p+d2vs13[x][[k,1]])!)*((q)!)*((r+d2vs13[x][[k,2]])!)*((n-x-p-q-r)!))),0\Big]\right)+$$

$$\left(\sum_{x=3}^{n}\sum_{p=0}^{n-x}\sum_{q=0}^{n-x-p}\sum_{r=0}^{n-x-p-q}(n!/(((p)!)*((q+x)!)*((r)!)*((n-x-p-q-r)!)))*\mathrm{If}\Big[\mathrm{Length}[d2vs14[x]]>0,\ \sum_{k=1}^{\mathrm{Length}[d2vs14[x]]}(n!/(((p+d2vs14[x][[k,1]])!)*((q)!)*((r)!)*((n-x-p-q-r+d2vs14[x][[k,2]])!))),0\Big]\right)+$$

$$\left(\sum_{x=3}^{n}\sum_{p=0}^{n-x}\sum_{q=0}^{n-x-p}\sum_{r=0}^{n-x-p-q}\mathrm{If}\Big[\mathrm{Length}[d23vs14[x]]>0,\ \sum_{k=1}^{\mathrm{Length}[d23vs14[x]]}((n!/(((p)!)*((q+d23vs14[x][[k,1]])!)*((r+d23vs14[x][[k,2]])!)*((n-x-p-q-r)!)))*(n!/(((p+d23vs14[x][[k,3]])!)*((q)!)*((r)!)*((n-x-p-q-r+d23vs14[x][[k,4]])!)))),0\Big]\right)+$$

$$\left(\sum_{x=3}^{n}\sum_{p=0}^{n-x}\sum_{q=0}^{n-x-p}\sum_{r=0}^{n-x-p-q}\mathrm{If}\Big[\mathrm{Length}[d24vs13[x]]>0,\ \sum_{k=1}^{\mathrm{Length}[d24vs13[x]]}((n!/(((p)!)*((q+d24vs13[x][[k,1]])!)*((r)!)*((n-x-p-q-r+d24vs13[x][[k,2]])!)))*(n!/(((p+d24vs13[x][[k,3]])!)*((q)!)*((r+d24vs13[x][[k,4]])!)*((n-x-p-q-r)!)))),0\Big]\right)$$

図 8.

プログラム 8. 例 *11* における *C* によるプログラム使用した数字が *1,2,3,4,5* で、数字を *6* 個ずつ使ったものの、積と和の逆転を見つける。

```
#include<stdio.h>
#include<time.h>

void ft_sekiwa() {
    unsigned int a, b, c, d, e, f, g, h, I, j, k, l;
    unsigned long counter = 0;
    unsigned long all = 0;
    time_t start_time, end_time;
    start_time = time(NULL); #開始時間を測定
    for (a = 1; a <= 5; a++) {
        for (b = 1; b <= 5; b++) {
            for (c = 1; c <= 5; c++) {
                for (d = 1; d <= 5; d++) {
                    for (e = 1; e <= 5; e++) {
                        for ( f = 1; f <= 5; f++) {
                            for (g = 1; g <= 5; g++) {
                                for (h = 1; h <= 5; h++) {
                                    for (i = 1; i <= 5; i++) {
                                        for (j = 1; j <= 5; j++) {
                                            for (k = 1; k <= 5; k++) {
                                                for (l = 1; l <= 5; l++) {
                                                    all++;
                                                    if (( a+b+c+d+e+f < g+h+I+j+k+l ) && ( a*b*c*d*e*f > g*h*I*j*k*l )) {
                                                        counter++;
                                                    }
                                                }}}}}}}}}}}}
    end_time = time(NULL); #終了時間を測定
    printf(“個数:%lu\n 割合:%f\n 時間:%ld 秒\n”, counter, float(counter) / float(all) * 2, end_time - start_time);
}

int main() {
    ft_sekiwa();
    return 0;
}
```

プログラム 9. 例 *12* の (i) における計算では次の *Mathematica* のプログラムを使った。

```
NIntegrate[-x + b1 + b2 - (b1*b2)/x, {b2, 0, 10}, {b1, 0, b2}, {x, b1,b2}]
NIntegrate[-x + b1 + b2 - (b1*b2)/x, {b2, 0, 10}, {b1, b2, 10}, {x,b2, b1}]
```

プログラム 10. 例 *12* の (ii) における計算では次の *Mathematica* のプログラムを使った

。

```
co = 0; Do[{a1, a2, b1, b2} = {RandomReal[{0, 10}],
RandomReal[{0, 10}], RandomReal[{0, 10}], RandomReal[{0, 10}]};
If[a1 + a2 < b1 + b2 && a1 a2 > b1 b2,co = co + 1,],{nn,1,10000000}];co
```

プログラム 11. 小節 *5.2* で書いたことについて。次のプログラムは *Arrow* の不可能性定理の計算機による実証のためのもので、並列処理で行なっている。

```
import itertools
from itertools import permutations
from multiprocessing import Pool
from multiprocessing import Process
import time
start = time.time()

def arr(sss, tu):
    ad1 = [[0 for i in range(4)] for j in range(4)]
    ad2 = [[0 for i in range(4)] for j in range(4)]
    ad3 = [[0 for i in range(4)] for j in range(4)]

    for k in range(1, 4):
        for j in range(1, 4):
            t1 = tu[0]
            if t1.index(k) < t1.index(j):
                ad1[k][j] = ad1[k][j]+1

    for k in range(1, 4):
        for j in range(1, 4):
            t1 = tu[1]
            if t1.index(k) < t1.index(j):
                ad2[k][j] = ad2[k][j]+1

    for k in range(1, 4):
        for j in range(1, 4):
            t1 = tu[2]
            if t1.index(k) < t1.index(j):
                ad3[k][j] = ad3[k][j]+1

    x = [[1, 2, 3], [1, 3, 2], [2, 1, 3], [2, 3, 1], [3, 1, 2], [3, 2, 1]]

# (1)The case of three
```

```
    if ad1[1][2]+ad2[1][2]+ad3[1][2] == 3:
        x = [i for i in x if i.index(1) < i.index(2)]

    if ad1[1][3]+ad2[1][3]+ad3[1][3] == 3:
        x = [i for i in x if i.index(1) < i.index(3)]

    if ad1[2][1]+ad2[2][1]+ad3[2][1] == 3:
        x = [i for i in x if i.index(2) < i.index(1)]

    if ad1[2][3]+ad2[2][3]+ad3[2][3] == 3:
        x = [i for i in x if i.index(2) < i.index(3)]

    if ad1[3][1]+ad2[3][1]+ad3[3][1] == 3:
        x = [i for i in x if i.index(3) < i.index(1)]

    if ad1[3][2]+ad2[3][2]+ad3[3][2] == 3:
        x = [i for i in x if i.index(3) < i.index(2)]

# (2) The case for two to one.
# (2.1) The case for 1 and 2

    if ad1[1][2]+ad2[1][2] == 2 and ad3[1][2] == 0:
        x = [i for i in x if i.index(1)*(-1)**sss[0] < i.index(2)*(-1)**sss[0]]

    if ad1[1][2]+ad3[1][2] == 2 and ad2[1][2] == 0:
        x = [i for i in x if i.index(1)*(-1)**sss[1] < i.index(2)*(-1)**sss[1]]

    if ad2[1][2]+ad3[1][2] == 2 and ad1[1][2] == 0:
        x = [i for i in x if i.index(1)*(-1)**sss[2] < i.index(2)*(-1)**sss[2]]

# (2.2) The case for 1 and 3

    if ad1[1][3]+ad2[1][3] == 2 and ad3[1][3] == 0:
        x = [i for i in x if i.index(1)*(-1)**sss[3] < i.index(3)*(-1)**sss[3]]

    if ad1[1][3]+ad3[1][3] == 2 and ad2[1][3] == 0:
        x = [i for i in x if i.index(1)*(-1)**sss[4] < i.index(3)*(-1)**sss[4]]

    if ad2[1][3]+ad3[1][3] == 2 and ad1[1][3] == 0:
        x = [i for i in x if i.index(1)*(-1)**sss[5] < i.index(3)*(-1)**sss[5]]

# (2.3) The case for 2 and 3

    if ad1[2][3]+ad2[2][3] == 2 and ad3[2][3] == 0:
        x = [i for i in x if i.index(2)*(-1)**sss[6] < i.index(3)*(-1)**sss[6]]

    if ad1[2][3]+ad3[2][3] == 2 and ad2[2][3] == 0:
        x = [i for i in x if i.index(2)*(-1)**sss[7] < i.index(3)*(-1)**sss[7]]
```

```
    if ad2[2][3]+ad3[2][3] == 2 and ad1[2][3] == 0:
        x = [i for i in x if i.index(2)*(-1)**sss[8] < i.index(3)*(-1)**sss[8]]

# (2.4) The case for 2 and 1

    if ad1[2][1]+ad2[2][1] == 2 and ad3[2][1] == 0:
        x = [i for i in x if i.index(2)*(-1)**sss[9] < i.index(1)*(-1)**sss[9]]

    if ad1[2][1]+ad3[2][1] == 2 and ad2[2][1] == 0:
        x = [i for i in x if i.index(
            2)*(-1)**sss[10] < i.index(1)*(-1)**sss[10]]

    if ad2[2][1]+ad3[2][1] == 2 and ad1[2][1] == 0:
        x = [i for i in x if i.index(
            2)*(-1)**sss[11] < i.index(1)*(-1)**sss[11]]

# (2.5) The case for 3 and 1

    if ad1[3][1]+ad2[3][1] == 2 and ad3[3][1] == 0:
        x = [i for i in x if i.index(
            3)*(-1)**sss[12] < i.index(1)*(-1)**sss[12]]

    if ad1[3][1]+ad3[3][1] == 2 and ad2[3][1] == 0:
        x = [i for i in x if i.index(
            3)*(-1)**sss[13] < i.index(1)*(-1)**sss[13]]

    if ad2[3][1]+ad3[3][1] == 2 and ad1[3][1] == 0:
        x = [i for i in x if i.index(
            3)*(-1)**sss[14] < i.index(1)*(-1)**sss[14]]

# (2.6) The case for 2 and 3

    if ad1[3][2]+ad2[3][2] == 2 and ad3[3][2] == 0:
        x = [i for i in x if i.index(
            3)*(-1)**sss[15] < i.index(2)*(-1)**sss[15]]

    if ad1[3][2]+ad3[3][2] == 2 and ad2[3][2] == 0:
        x = [i for i in x if i.index(
            3)*(-1)**sss[16] < i.index(2)*(-1)**sss[16]]

    if ad2[3][2]+ad3[3][2] == 2 and ad1[3][2] == 0:
        x = [i for i in x if i.index(
            3)*(-1)**sss[17] < i.index(2)*(-1)**sss[17]]
    return(x)

x = [[1, 2, 3], [1, 3, 2], [2, 1, 3], [2, 3, 1], [3, 1, 2], [3, 2, 1]]
pr = []

for n in range(6):
```

```
    for m in range(6):
        for p in range(6):
            pr.append([x[n], x[m], x[p]])

y=[[[1, 2, 3],[2 ,3,1],[3, 1, 2]],[[1, 2, 3],[3, 1, 2],[2, 3, 1]],
[[2, 3, 1],[1, 2, 3],[3, 1, 2]],[[2, 3, 1],[3, 1, 2],[1, 2, 3]],
[[3, 1, 2],[1, 2, 3],[2, 3, 1]],[[3, 1, 2],[2, 3, 1],[1, 2, 3]],
[[3,2,1],[2,1,3],[1,3,2]],[[3,2,1],[1,3,2],[2,1,3]],
[[2,1,3],[3,2,1],[1,3,2]],[[2,1,3],[1,3,2],[3,2,1]],
[[1,3,2],[3,2,1],[2,1,3]],[[1,3,2],[2,1,3],[3,2,1]]]

for j in range(12):
   pr.remove(y[j])

dd = []
for e in itertools.product([1, 0], repeat=18):
    dd.append(list(e))

ks = len(dd)

coun = 0
bestd = []

def threading_area(input):
    (dd, coun, j) = input
    global answer
    coun = coun+1
    num = 1
    for i in range(12):
        num = num*len(arr(dd[j], y[i]))
    if num > 0:
        print(dd[j])
        return j + 1
    return

if __name__ == '__main__':
    p = Pool(16)
    result = []
    values = [(dd, coun, x) for x in range(ks)]
    result += p.map(threading_area, values)
    result = [a for a in result if a != None]
    print(len(result))

print(time.time() - start
```