# リーダ選挙個体群プロトコルのシミュレーションによる性能比較

森田　純平 *　　首藤　裕一 *　　増澤　利光 *

## 1　はじめに

個体群プロトコルモデルとは，計算能力が非常に制限されたモバイルデバイスで構成されたネットワークを抽象化したモデルである. 個々の計算機を個体，複数の個体で構成されるネットワーク全体を個体群と呼ぶ. 各個体は有限状態機械である. 各時刻 $t = 0, 1, 2, ...$ において 1 組の個体の間で交流と呼ばれる通信が発生し，その際互いの状態をもとに自身の状態を更新する. この交流によって個体の状態が更新されていくことで個体群全体の計算が進む. 個体群モデルで表現できるモバイルネットワークの例として，通信範囲が極めて狭いセンサノードを鳥の群れの一羽一羽につけることで構成されるセンサネットワークが挙げられる. このようなネットワークではノードの通信範囲が狭いために，それらのノードを付した 2 羽の鳥が十分に接近したときにのみ，2 つのノードのあいだで通信が可能となる. したがって，2 つの個体間の通信によって状態が更新されていく個体群モデルで表現できる.

個体群プロトコルモデルにおいて各個体は均一である，すなわち，個体は固有の識別子を持たず，すべての個体は実行開始時に同じ状態を持ち，同じ交流規則によって状態を更新していく. 一方で，個体群中に 1 つのリーダ個体が存在すれば多くの問題を高速に実行できることが知られている. リーダ選挙プロトコルは，個体群プロトコルモデルにおいてただ 1 つの個体をリーダとして選出するプロトコルである. それぞれのリーダ選挙プロトコルの性能は時間計算量 (ただ 1 つのリーダ個体を選出して安定するまでにかかる時間) と空間計算量 (リーダ選挙プロトコルを実行するのに必要な各個体の状態数) で評価される. これらは個体数 $n$ についての関数を用いて漸近的に評価されるが，これは $n$ が十分に大きいときの定数係数を無視した評価である. 現実的な個体数 (100〜1 億など) の場合の性能を正確に表しているとは限らない. そのため，漸近的計算量では劣るプロトコルが，現実的な個体数の場合には優れている可能性がある. そこで，本研究ではシミュレーションによって個体数 $n$ を現実的な値に設定した際のプロトコルの性能比較を行う.

## 2　諸定義

### 2.1　個体群プロトコル

個体群とは，個体と呼ばれる有限個のノードで構成されるシステムである．各個体は状態を持つ．個体の状態は後述する交流の発生によって変化する．個体群は単純有向グラフ $G(V, E)$ で表す．$V = \{v_1, .., v_n\}(n > 1)$ は個体群を構成する個体の集合である．$E \subseteq V \times V \backslash \{(v, v) | v \in V\}$ は

*大阪大学　大学院情報科学研究科

交流可能な個体の順序対の集合を表す．具体的には，$(u,v) \in E$ のとき，個体 $u$ は呼びかけ側，個体 $v$ は応答側として交流可能である．本稿では，$G$ が完全 (有向) グラフとなる個体群のみを考える．すなわち，$E = V \times V \backslash \{(v,v)|v \in V\}$ とする．

プロトコル $P(Q, X, Y, I, O, \delta)$ は，状態集合 $Q$ と入力記号集合 $X$，出力記号集合 $Y$，入力関数 $I : X \to Q$，出力関数 $O : Q \to Y$，状態遷移関数 $\delta : Q \times Q \to Q \times Q$ で構成される．$Q$ と $X, Y$ は有限集合である．入力関数 $I$ は，実行開始時に個体が取得した入力に応じて個体に初期状態を与える．プロトコルの実行では，各ステップにおいて，1 組の個体ペアが選択され，その個体間で交流が発生するものとする．関数 $\delta$ は，ある 2 つの個体の交流が発生した際，交流後の各個体の状態を決定する．例として，状態 $p, q, p', q'$ が $\delta(p,q) = (p', q')$ を満たすとする．状態 $p$ の個体 $u$ と状態 $q$ の個体 $v$ がそれぞれ呼びかけ側，応答側として交流を行った場合，状態遷移関数 $\delta$ に従い，交流後の各個体の状態はそれぞれ $p', q'$ となる．また，交流は各時点において正確に 1 つだけ発生する[1]．スケジューラは，各ステップに発生する交流を一様ランダムに選択する．すなわち, 各ステップにおいて発生する交流は，発生し得る交流のうちから等確率かつ独立に選ばれる．プロトコル $P$ が収束する (問題の解を求めて安定する) までの交流回数の期待値を期待交流回数と呼ぶ．プロトコルの収束時間は, 期待交流回数を個体数 $n$ で割った**並列時間**で評価する．

### 2.2 リーダ選挙

リーダ選挙問題とは，個体群中のただひとつの個体をリーダ状態に，その他の個体をフォロワ状態にすることを目的とする問題である．具体的には，ただひとつの個体が記号 $l$ を出力し，その他のすべての個体は記号 $f$ を出力する状況に個体群が安定することを求める問題である．

## 3 プロトコルのシミュレーション

ここではシミュレーションによりプロトコルの個体数と収束時間の関係と，収束時間の分布を示す．

### 3.1 比較するプロトコル一覧

シミュレーションの対象とするプロトコルが紹介されている論文を，それぞれの期待収束時間，空間計算量，本稿で用いる呼称とともに表 1 に示す．

各プロトコルの動作を簡単に説明する．

**プロトコル 1**

初期状況ですべての個体はリーダ状態である．リーダどうしが交流すると応答側がフォロワになる．すべての個体ペアはいずれ交流するので，いずれリーダの数は 1 になり，その後変化することはない．

**プロトコル 2**

初期状況ですべての個体はリーダ状態である．各個体は seeding，lottery，tournament，minion のうち 1 つのモードを取り，seeding モードから開始しこの順に遷移する．seeding モードでは

[1]実際のネットワークでは, 交流はいつ発生するか分からず (非同期式モデル), 複数の交流が同時に発生することもある．しかし, 各個体は同時には 1 つの交流にしか参加できないとする限り, 同時に発生する交流を 1 つと仮定しても一般性は失われない．

表 1: 比較するプロトコル一覧

| 論文 | 期待収束時間 | 空間計算量 | 呼称 |
|---|---|---|---|
| [1] | $O(n)$ | $O(1)$ | プロトコル 1 |
| [2] | $O(\log^{5.3} n \log\log n)$ | $O(\log^2 n)$ | プロトコル 2 |
| [5] | $O(\log^3 n)$ | $O(\log^3 n)$ | プロトコル 3 |
| [4] | $O(\log^2 n)$ | $O(\log n)$ | プロトコル 4 |
| [6] | $O(\log n)$ | $O(n)$ | プロトコル 5 |
| [3] | $O(\log n)$ | $O(\log n)$ | プロトコル 6 |

全体の *coin* の値が一様ランダムに近づくのを待つため一定回数の交流をする．ここで，*coin* とは各個体が交流のたびに true，false を入れ替える変数であり，初期値は false である．一定回数の交流を終えると lottery モードに移行する．lottery モードでは，各個体は *coin* の値が false であるような個体と交流するまでの交流回数を変数 *payoff* に記憶し，tournament モードに移行する．tournament モードでは，交流のたびに 2 個体が $(payoff, level)$ を辞書順に大小比較し小さい方は相手の $(payoff, level)$ を記録してフォロワになり，minion モードに移行する．ここで *level* とは，tournament モードで $\log(Max\_payoff)$ 回の交流をするたびにインクリメントされる変数であり，*payoff* が同じリーダが複数現れた時のためのタイブレイカーとして用いる．ここで *Max_payoff* とは，*payoff* の取りうる上限値であり，$10\log n$ 以上の値をとる．$(payoff, level)$ が等しい個体が交流した場合は，2 個体の *coin* の値により勝敗を決定する．minion モードを取る個体は遭遇した最大の $(payoff, level)$ の組を記録することでその組をシステム全体に伝搬し，$(payoff, level)$ の小さいリーダをフォロワにする．

**プロトコル 3**

初期状況ですべての個体はリーダ状態である．各個体はただ 1 つの変数 $s \in \{-m, ..., -1, 1, 2, ..., m, m+1\}$ をもつ．ここで，$m$ は $\Theta(\log^3 n)$ の定数である．$s$ の初期値は 1 であり，各個体は $s > 0$ のときリーダ状態，$s < 0$ のときフォロワ状態を取る．競争の際には $s$ の絶対値で大小比較し，リーダ状態で勝った場合は $s$ をインクリメントし負けた場合は相手の $s$ の絶対値に $-1$ をかけた数値を自身の $s$ に記録する．ただし，$s$ が $m+1$ に達する個体が複数出現したとき，それ以上インクリメントできないことによる引き分けが発生するため，$s = m+1$ の場合は，これを避けるためインクリメントする代わりに $m$ を記録する．フォロワは遭遇した $s$ の中で絶対値の最も大きいものをシステム全体に伝搬し，リーダをフォロワにする．

**プロトコル 4**

初期状況ですべての個体はリーダ状態である．すべての個体は全体の個体数 $n$ を知っているものとする．各個体は marking と tournament のうちいずれかのフェイズを取り，marking フェイズから開始する．marking フェイズでは自身の変数 *counter* を 0 から最低でも $3\log\log n$ まで上げ，以降はコイン投げにより上げ，一度でも裏が出る，または *counter* が $4\log\log n$ に達すると tournament フェイズに移行する．ここで，コイン投げとは，交流相手の交流のたびに true(表)，false(裏) を入れ替える変数 *coin* を参照することである．*counter* が $4\log\log n$ に達して移行した場合は自身の変数 *marker* を true にする．tournament フェイズでは *counter* の大小比較により競争し，相手の *marker* が true なら自身の *counter* を上げる．ここではタイブレイカーとして 2 個体の *coin* の値を用いる．フォロワは遭遇した最大の *counter* をシステム全体に伝搬し，*counter* の小さいリーダをフォロワにする．

**プロトコル 5**

初期状況ですべての個体はリーダ状態である．すべての個体は個体数の上界 $N$（ただし，ある定数 $b$ に対し，$N \geq n^b$）を知っており，変数 $e, r$ により競争する．$e$ は相手をフォロワにする，または引き分けになるほど大きくなる変数で，$r$ は乱数である．$e$ をインクリメントする際には，乱数 $r$ を $0, 1, \ldots, n$ の範囲から抽出する[2]．個体数の上界 $N$ の知識は $e$ をインクリメントする頻度の調節に利用される．互いの $e$ を比較して競争し，$r$ をタイブレイカーとして用いる．フォロワは遭遇した最大の $(e, r)$ の組をシステム全体に伝搬し，$(e, r)$ の組の小さいリーダをフォロワにする．

**プロトコル 6**

初期状況ですべての個体はリーダ状態である．このプロトコルは全個体で緩やかに同期を取り，3 つのサブプロトコル QuickElimination()，Tournament()，BackUp() を順次実行する．QuickElimination() では，各リーダはプロトコル 2 と同様に，連続して *coin*=true の個体と交流した回数を変数 *levelQ* に記録し，その最大値を個体群全体に伝播させる．自身の *levelQ* よりも大きな *levelQ* を観測したリーダはフォロワになる．Tournament() では，QuickElimination() で生き残った各個体が，変数 *coin* を利用して各自乱数 *rand* を生成する．次に，全個体の中で最大の *rand* の値を個体群全体に伝播させる．QuickElimination() と同様に，自身の *rand* より大きな *rand* を観測したリーダはフォロワになる．BackUp() では，Tournament() で生き残った各リーダが，変数 *levelB* の大小比較を行うことで競争し，リーダをひとつに絞る．各個体は，一定の頻度で変数 *levelB* を 1 増加することを試みる．この試行は，必ず成功するわけではなく，変数 *coin* を用いることで確率 1/2 で成功するようにする．すなわち，確率 1/2 で *levelB* は増加せず，もとの値を維持する．こうすることで，一定周期ごとにリーダの数が半減していき，やがてひとつのリーダが選出される．QuickElimination() と Tournament() の実行には $O(\log n)$ 並列時間を要し，確率 $1 - O(1/\log n)$ でリーダが 1 つになる．複数のリーダが生き残ってしまった場合は，BackUp() が $O(\log^2 n)$ 並列時間 (期待値) でリーダを 1 つにする．結果として，プロトコル 6 全体の実行で，リーダがひとつになるまでの並列時間の期待値は $O(\log n) + \frac{1}{\log n} \cdot O(\log^2 n) = O(\log n)$ となる．

## 3.2 シミュレーション結果

個体数 500 から 10000 まで刻み幅 500 で，それぞれの個体数に対して試行回数を 100 回としてシミュレーション実験を行った．横軸を個体数，縦軸を収束時間としたときの関係を図 1 に示す．図 1 ではプロトコル 3〜6 の関係がわからないので，プロトコル 3 〜 6 の実験結果のみを図 2 に示す．図 2 のシミュレーションでは，評価対象のプロトコルの収束時間が早いため，精度を向上させるために試行回数を 10000 とした．図 1 によると，プロトコル 1 とプロトコル 2 は他に比べて極端に遅くなっているが，これは漸近的な評価と一致する．一方で，図 2 によれば，プロトコル 3 とプロトコル 4 は収束時間が 40 ほどで安定しており，プロトコル 5 は最も早く収束し，収束時間は 7 ほどで安定していることが分かる．

[2] プロトコル 1，2，3，4，6 を含め，標準的な個体群プロトコルは，交流の際に各個体が乱数を直接利用する能力を仮定していない．したがって，上述の変数 *coin* などを管理することによって，ランダムスケジューラからランダムな要素を抽出する．一方で，このプロトコルは，交流のたびに直接乱数を抽出することが可能であるという仮定を必要とする．

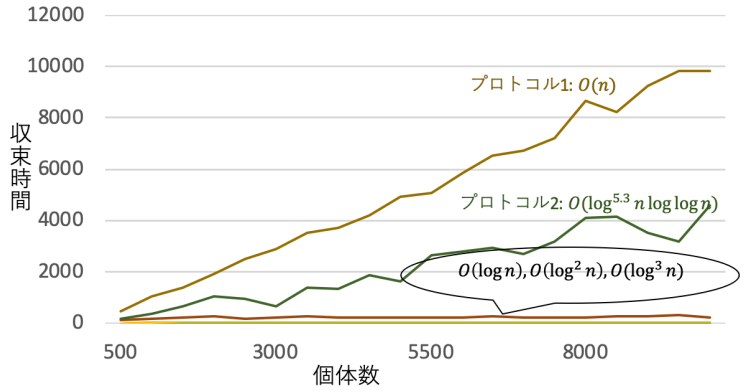


図 1: 全プロトコルの収束時間の比較

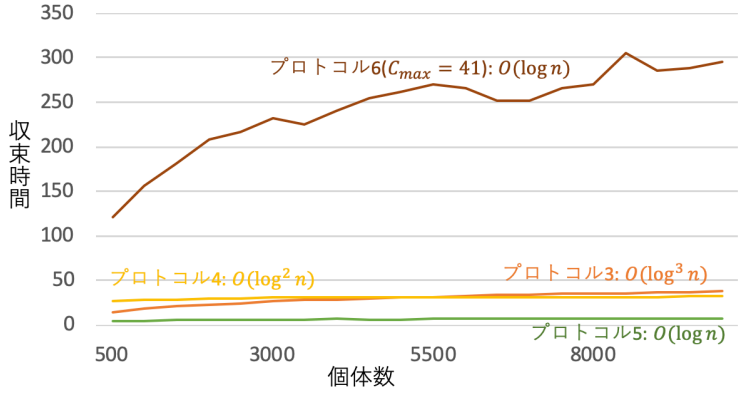


図 2: プロトコル 3〜6 の収束時間の比較

プロトコル 6 では各個体がサブプロトコルを切り替える早さを表す定数 $c_{\max}$ が存在するが，この数値によってプロトコルの収束時間は変化する．図 3 は，$c_{\max}$ を 5 から 41 までの範囲で動かしたときの収束時間の変化を示す．図 3 からは，$c_{\max}$ が 5 から 41 の範囲では $c_{\max}$ が小さいほど早く収束することが分かる．また，図 4 は $c_{\max}$ を 1 から 5 までの範囲で動かした時の収束時間の変化を示す．図 4 によると $c_{\max} = 4$ としたとき収束時間は最も短くなり，さらに小さくしていくと収束時間は長くなるが $c_{\max} = 1$ になると $c_{\max} = 2$ に比べて収束時間はやや短くなる．

個体数が大きいとき (10000 から 240000 まで) にプロトコル 3，プロトコル 4，プロトコル 5，プロトコル 6($c_{\max} = 4$) を図 5 により比較する．プロトコル 3，プロトコル 4，プロトコル 5 の試行回数は 100 回とし，プロトコル 6($c_{\max} = 4$) は収束時間の分散が大きいためより正確な平均値を調べるために試行回数を 10000 回とした．なお，プロトコル 6($c_{\max} = 4$) については，シミュレーションに必要なメモリ容量の都合上個体数 160000 までのデータしか取れていない．図 5 によると，個体数が 10000 から 160000 の範囲では，プロトコル 6($c_{\max} = 4$) はプロトコル 4 とおよそ等しい収束時間をもつことが分かる．

以上の結果から，これらのリーダ選挙プロトコルの時間計算量の漸近的評価の大小は，個体数 10000 程度までの規模であれば，収束時間の大小と必ずしも一致しないことが見て取れる．特にプ

ロトコル 3 とプロトコル 4 については，それぞれの漸近的な収束時間が $O(\log^3 n)$，$O(\log^2 n)$ であることが既存研究で示されているが，それらのプロトコルの収束時間の漸近的下界が示されているわけではない．したがって，実際の収束時間は，プロトコル 6 と同様に $O(\log n)$ である可能性があり，本シミュレーションの結果はその可能性を示唆している．ただし，実行時間の関係でプロトコル 3，4，6 の収束時間の比較は個体数 160000 までしかできていないので，プロトコル 3，4 の収束時間がそれぞれ $\Theta(\log^3 n)$，$\Theta(\log^2 n)$ である可能性は依然として残る．

以降では，各プロトコルが持つ特性調べるため，収束時間の分布を観察し，より詳細な考察を与える．

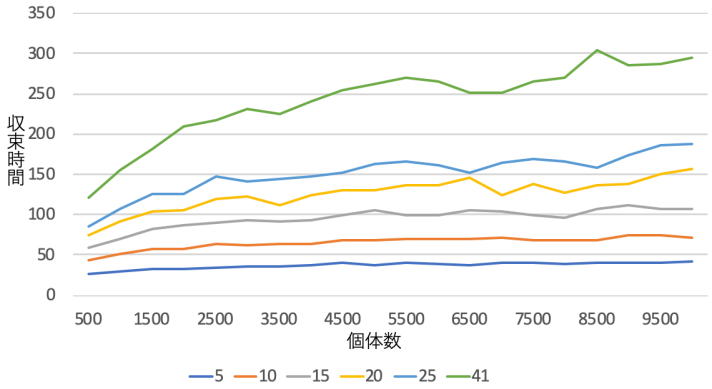


図 3: $c_{\max}$ を 5 から 41 まで動かしたときの収束時間の変化

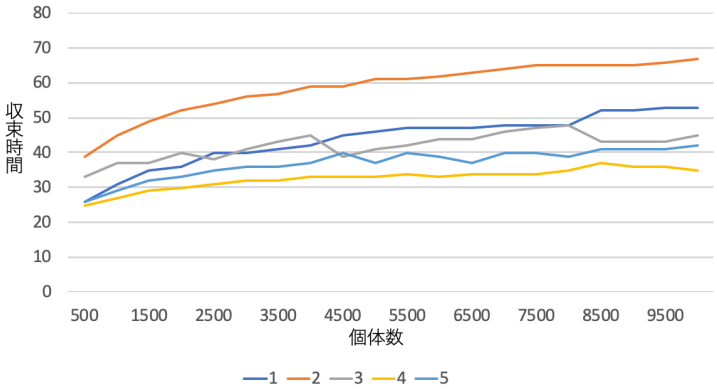


図 4: $c_{\max}$ を 1 から 5 まで動かしたときの収束時間の変化

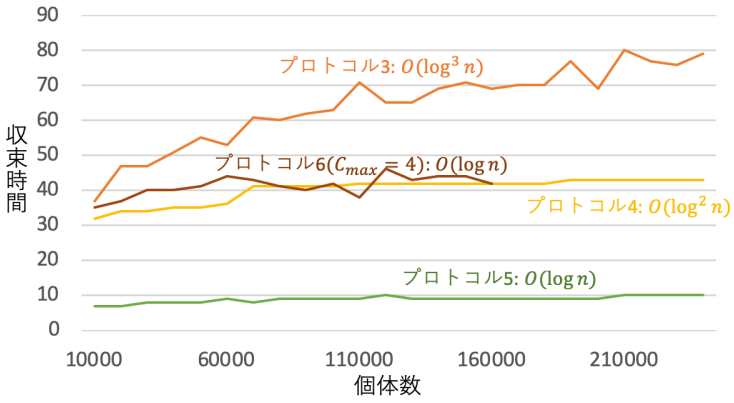


図 5: 個体数 10000 から 240000 まで動かしたときの収束時間の変化

個体数を 10000 としたときの，プロトコル 1，プロトコル 2，プロトコル 3，プロトコル 4，プロトコル 5，プロトコル 6($c_{\max} = 41$) の収束時間の分布を表すヒストグラムをそれぞれ図 6，図 7，図 8，図 9，図 10，図 11 に示す (試行回数 10000)．プロトコル 6 は収束時間の分散は比較的大きく，短時間で収束する可能性は高いが，そこで収束できなければ収束に長時間を要することが図 11 から分かる．前述の通り，QuickElimination() および Tournament() は $O(\log n)$ 時間で完了し，確率 $1 - O(1/\log n)$ でリーダをひとつに絞るが，そこで複数のリーダが生き残ると，BackUp() の実行でリーダをひとつに絞るまでに $O(\log^2 n)$ の期待時間を要する．図 11 の分布はプロトコル 6 のこの特性をよく表しているといえる．

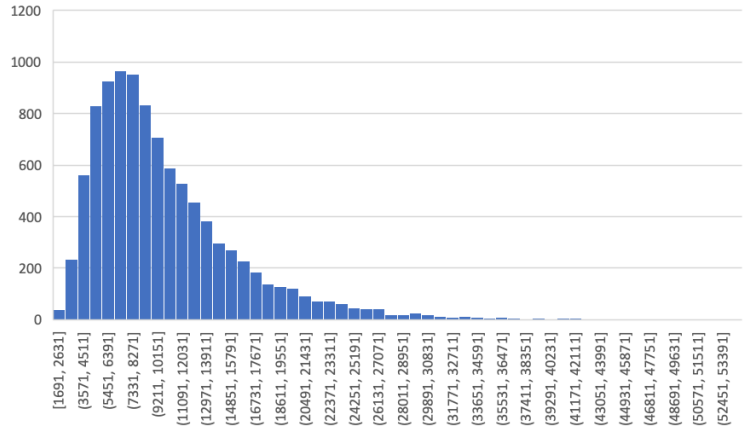


図 6: プロトコル 1 のヒストグラム

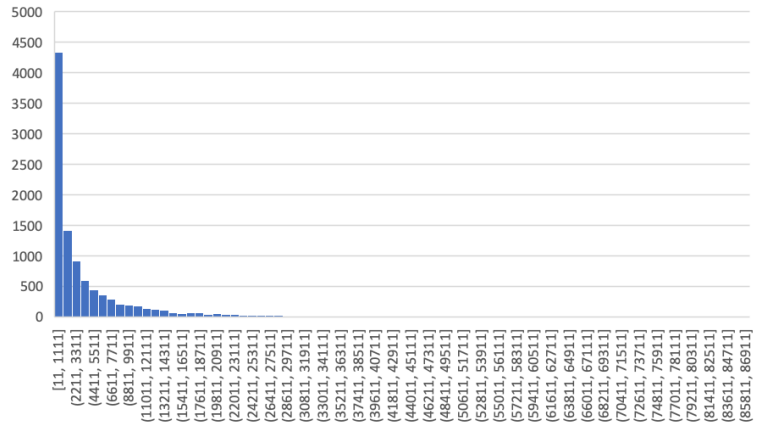


図 7: プロトコル 2 のヒストグラム

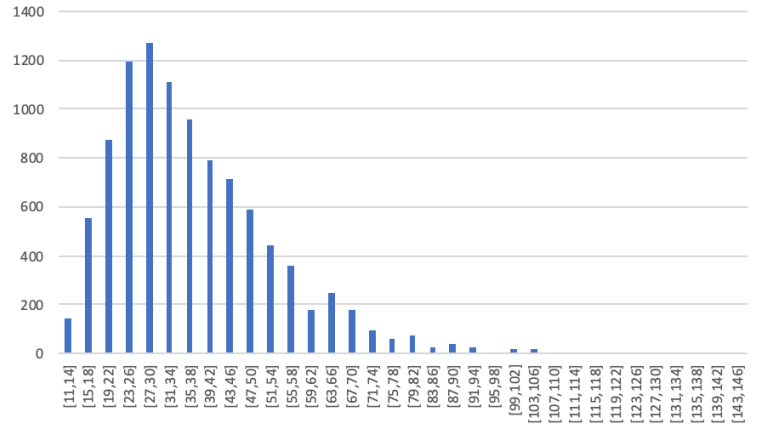


図 8: プロトコル 3 のヒストグラム

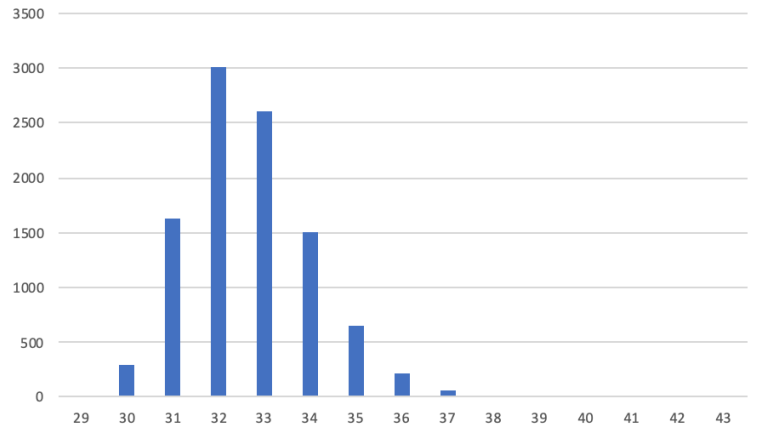


図 9: プロトコル 4 のヒストグラム

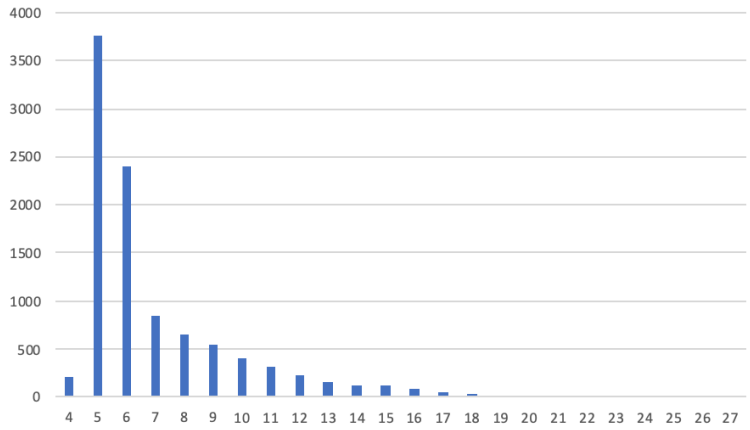


図 10: プロトコル 5 のヒストグラム

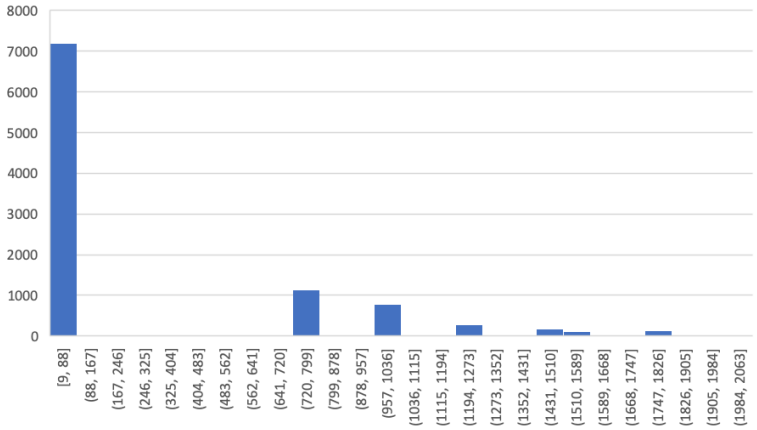


図 11: プロトコル 6($c_{\max} = 41$) のヒストグラム

## 4　まとめ

本稿では，シミュレーションによって，個体群プロトコルの個体数 $n$ を現実的な値に設定した際の 6 つのリーダ選挙プロトコルの性能を比較した．シミュレーションで評価できた範囲の個体数 (〜160000 個まで) においては，漸近的評価で収束時間の遅いプロトコルが漸近的評価で収束時間の早いプロトコルよりも短時間で収束する場合があることが分かった．

**謝辞**　本研究は，JSPS 科研費 19H04085 および 20H04140 の支援を受けたものです．

## 参考文献

[1] Angluin, Dana, et al. ”Computation in networks of passively mobile finite-state sensors.” Distributed computing 18(4), pages 235-253, 2006.

[2] Alistarh, Dan, et al. "Time-space trade-offs in population protocols." Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, pages 2567-2572, 2017.

[3] Sudo, Yuichi, et al. "Logarithmic expected-time leader election in population protocol model." International Symposium on Stabilizing, Safety, and Security of Distributed Systems, pages 3-11, 2019.

[4] Berenbrink, Petra, et al. "Simple and efficient leader election." 1st Symposium on Simplicity in Algorithms (SOSA 2018), pages 2-4, 2018.

[5] Alistarh, Dan, and Rati Gelashvili. "Polylogarithmic-time leader election in population protocols." International Colloquium on Automata, Languages, and Programming, pages 4-5, 2015.

[6] Michail, Othon, Paul G. Spirakis, and Michail Theofilatos. "Simple and Fast Approximate Counting and Leader Election in Populations." International Symposium on Stabilizing, Safety, and Security of Distributed Systems, pages 158-164, 2018.