

確率 1 で衝突判定を解く時間・空間効率の良い個体群プロトコル

新谷 拓海 首藤 裕一

法政大学

概要

本研究では、個体群プロトコルにおける衝突判定について取り組む。個体と呼ばれる n 個の状態機械で構成されるネットワークを個体群と呼ぶ。衝突判定問題とは、各個体に 1 以上 n 以下の任意の整数が入力して与えられて実行を開始し、この入力値に重複が存在するかどうかを全個体に認識させる問題である。具体的には、すべての個体の入力が異なるときはすべての個体が **false** を出力し、そうでないときはすべての個体が **true** を出力することを目的とする。

本研究では、衝突判定問題を確率 1 で解くアルゴリズムを提案する。著者の知る限り、このアルゴリズムは初の確率 1 で多項式状態かつ劣線形時間で衝突判定問題を解くアルゴリズムである。具体的には、状態数と安定時間はそれぞれ $O(n^3 \log^2 n)$ 、 $O(\sqrt{n} \log n)$ である。

1 はじめに

本研究では、個体群プロトコルにおける衝突判定について取り組む。個体群プロトコル [4] とは、**個体**と呼ばれる状態機械が交流と呼ばれる通信を介して相互に状態を更新していくことで計算を進めるネットワークのモデルである。ネットワークを構成する全個体を総称して個体群と呼び、以後、個体群を構成する個体の数を n とする。各ステップで、交流可能な組み合わせのうち一様ランダムに 2 個の個体を選ばれ交流する。交流した個体はアルゴリズムが定める状態遷移関数に従って自分の状態を変化させる。本研究では、時間計算量は交流回数を個体数 n で割った値、すなわち並列時間で評価する。

実行開始時、 n 個の個体は 1 以上 n 以下の任意の整数が入力として与えられる。その値のことをランクという。衝突判定問題とは、同じランクを持つ個体が複数存在するか否かを判定するものである。全ての個体のランクが異なるときは全ての個体が **false** を出力し、そうではないときは **true** を出力することを目的とする。衝突判定問題に密接に関連する問題として自己安定順位付けと呼ばれる問題がある。これは、任意の初期状況から n 個の個体に 1 から n のランクがちょうどひとつずつ割り当てられた状況に遷移させることを目的とした問題である。既存の自己安定順位付けアルゴリズムの多くが、その実行の過程で衝突判定を行っており、さらに、その衝突判定に要する時間計算量や空間計算量がアルゴリズム全体の計算量のなかで支配的な割合を占める。したがって、計算量の小さい衝突判定アルゴリズムを与えることは自己安定順位付け問題の計算量を改良することに寄与する。

以降では、既存の衝突判定に必要な計算量について述べる。表 1 はそれらをまとめたものである。ただし、上二つ [7][6] は自己安定順位づけ問題に利用されている衝突判定に必要な計算量である。Cai らの自己安定順序付けアルゴリズム [7] は、同じランクを持つ個体同士が直接交流するときにランクの衝突を検知

表 1: 衝突判定

	安定時間	状態数	成功確率
[7]	$O(n)$	$O(n)$	1
[6]	$O(\log n)$	$O(n^{3n})$	1
[11]	$O(\sqrt{n} \log n)$	$O(n^3 \log^2 n)$	$1 - O(1/n)$
本研究	$O(\sqrt{n} \log n)$	$O(n^3 \log^2 n)$	1

するという愚直な検知方法を用いている。同じランクの個体同士が交流したとき、片方のランクを 1 加える (n ならば 1 に戻す) ことで自己安定順位付け問題を解いている。このような直接交流したときに衝突を検知する方法は、状態数 $O(n)$ 、安定時間 $O(n)$ で衝突判定を解くことができる。また、Burman ら [6] は自身のランクを全個体に伝搬することによって同じランクを持つ個体同士が直接交流せずに衝突を検知する方法を用いた。このアルゴリズムは状態数 $O(n^{3n})$ 、安定時間 $O(\log n)$ で判定できる。私 [11] は衝突検知する範囲を平方分割することによって自身のランクを伝搬する数を制限した。以上より、成功確率は $1 - O(1/n)$ に緩和したが、状態数 $O(n^3 \log^2 n)$ 、安定時間 $O(\sqrt{n} \log n)$ で衝突判定するアルゴリズムを提案した。

このように私 [11] は多項式状態かつ劣線形時間による衝突判定を実現したが、成功確率が $1 - O(1/n)$ であった。本研究では、状態数と安定時間は維持した状態で、成功確率 1 に改善したアルゴリズムを提案する。

残る未解決問題として、各個体が初期状態で始まるのではなく、(個体ごとに異なり得る) 任意の状態から実行を開始するものとしたときに、多項式状態かつ劣線形時間で衝突判定が可能なアルゴリズムが存在するか否かの問いが残る。

諸事情により、本稿では得られた主定理を述べるにとどめ、具体的なアルゴリズムの戦略や構成はワークショップ当日の発表で紹介する。ただし、個体数の知識の緩和に議論や提案アルゴリズムの応用例である緩安定名前付けアルゴリズムについては、当日の発表では簡単にしか触れない、あるいは、割愛するので、それぞれ 4 節および 5 節でやや詳細に説明する。

2 諸定義

個体群とは個体と呼ばれる有限個の状態機械で構成されるシステムである。各個体は状態を持ち、交流の発生によってその状態を変化させる。個体群は、単純有向グラフ $G = (V, E)$ で表す。 V は個体群を構成する個体の集合であり、 $|V| = n$ とする。また、 $E \subseteq V \times V$ は発生し得る交流の集合を表す。 $(a, b) \in E$ のとき、個体 a と個体 b がそれぞれ呼びかけ側、応答側として交流し得る。本研究では、 G が完全グラフとなる個体群のみを考える。すなわち、 $E = (V \times V) \setminus \{(v, v) | v \in V\}$ とする。また、時間とは交流回数を個体数で割った値、並列時間のことを表す。

プロトコル $P(Q, X, Y, R, I, O, \delta)$ は、状態集合 Q と入力記号の集合 X 、出力記号の集合 Y 、乱数の定義域 $R \subset \mathbb{N}$ 、出力関数 $O : Q \rightarrow Y$ 、状態遷移関数 $\delta : Q \times Q \times R \rightarrow Q \times Q$ で構成される。 Q, X, Y は有限集合である。実行開始時に、各個体 a には入力記号 $x \in X$ が与えられ、その入力記号と入力関数 $I : X \rightarrow Q$ により a の初期状態 $I(x)$ が決まる。 O は各個体の出力を決定する。個体 a が状態 $q \in Q$ を持つとき、 a の出力は $O(q)$ である。 δ はある 2 個体が交流したとき、それぞれの個体の状態と交流ごとに生成される乱数に基づいて、交流後の 2 個体の状態を決定する。

スケジューラとは、各時点において発生する交流を定める抽象的存在である。本研究で想定するスケジューラは一様ランダムスケジューラである。よって、発生し得る交流のうちから等確率に選ばれる。本研究では、個体群プロトコルに関連する多くの研究 [1][2][3][8][10] と同様に完全グラフかつ一様ランダムスケジューラであると仮定しているため、任意の個体 $a, b \in V$ に対して、各時刻で a, b がそれぞれ呼びかけ側、応答側となる交流が発生する確率はそれ以前の交流とは独立に常に $1/n(n-1)$ である。

衝突判定問題について説明する。実行開始時、 n 個の個体は 1 から n までのうち任意の整数値が入力として与えられている。すなわち、 $X = \{1, 2, \dots, n\}$ である。この入力値のことを以降では**ランク**と呼ぶ。衝突判定問題はランクが同じ個体が複数存在するとき、全ての個体が **true** を出力し、そうではないとき、全ての個体が **false** を出力することを目的とする。

状況 C から到達可能な全ての状況の出力が変わらないならば、状況 C を安定状況という。また、初期状況から安定状況になるまでに必要な時間を安定時間という。

3 主定理

定理 1. 状態数 $O(n^3 \log^2 n)$ 、期待安定時間 $O(\sqrt{n} \log n)$ 、確率 1 で衝突判定問題を解くアルゴリズムが存在する。

4 個体数の知識に関する議論

本稿においては衝突判定問題の入力値（ランク）の定義域を $\{1, 2, \dots, n\}$ に限定し、 n の正確な知識が全個体に与えられているものと仮定している。この仮定は、衝突判定アルゴリズムを任意の初期状況から n 個の個体に 1 から n までの識別子を割り当てる**自己安定順位付け問題**の異常検知に用いるという衝突判定問題の動機づけを考えると、自然なものであるといえる。一方で、任意の初期状況から n 個の個体に（1 から n という範囲に限定せず）異なる識別子を割り当てるという**自己安定名前付け問題**に应用することを考えると、衝突判定問題を、 $(n+1)$ 以上かもしれない任意の正整数が入力として与えられるものとして一般化してもよい。この場合、 n の正確な知識が与えられるとする仮定は自然ではないかもしれない。

幸いなことに、定理 1 を示すために用いる提案アルゴリズムは、個体数 n の正確な知識は必要とせず、 $n_U = O(n)$ を満たす n の上界 n_U と $n_L = \Omega(n)$ を満たす n の下界 n_L が与えられれば期待安定時間 $O(\sqrt{n} \log n)$ で衝突判定問題を解く。さらには、与えられた n_U および n_L が上記の条件（ $n_U = O(n)$, $n_U \geq n$, $n_L = \Omega(n)$, $n_L \leq n$ ）を満たさなかったとしても、期待安定時間が $O(\sqrt{n} \log n)$ となることが保証されなくなるものの確率 1 で衝突判定問題を解く。したがって、Berenbrink ら [5] が 2019 年に提案した、高確率で $O(n^2 \log n)$ 時間で $\lfloor \log n \rfloor$ か $\lceil \log n \rceil$ のいずれかの値を計算する状態数 $O(\log^2 n \log \log n)$ のアルゴリズムと組み合わせることで、上界 n_U および下界 n_L の知識すら必要とせず、期待安定時間 $O(\sqrt{n} \log n)$ かつ多項式状態で衝突判定問題を解くアルゴリズムを実現できる。

5 緩安定名前付けアルゴリズム

また、個体数の上界 $n_U \geq n$ （ $n_U = O(n)$ ）が既知であれば、任意の初期状況から開始して全個体に 1 から n_U の範囲の相異なる識別子を付与する**緩安定名前付けアルゴリズム**を設計することができる。全個体が 1 から n_U の範囲の相異なる識別子を持つという名前付け問題の仕様を満たす状況を正当な状況と呼ぶ。このアルゴリズムは、ある定数のパラメータ c を持ち、任意の初期状況から実行を開始して期待時間

$O(c\sqrt{n}\log^2 n)$ 時間以内に正当な状況（全体の部分集合を構成する安全状況）に到達し、以後、 $\Omega(n^c)$ 期待時間のあいだ正当な状況から逸脱しないことを保証する。また、このアルゴリズムは多項式状態しか用いない。このアルゴリズムの構成は以下の通りである。

- 提案アルゴリズムで管理する識別子を記憶する変数を **rank** とする。
- 提案アルゴリズムは、Sudo ら [9] の緩安定リーダ選挙アルゴリズムをサブモジュールとして用いる。このアルゴリズムは、任意の初期状況から $O(c\sqrt{n}\log n)$ 時間以内に安全状況に到達し、その後、 $\Omega(n^c)$ 期待時間のあいだ唯一のリーダを維持し続ける。また、このアルゴリズムは、全個体を $\Theta(c\log n)$ 時間ごとに同期を取る既存の緩安定 phase clock を備える。この phase clock は、安全状況に到達後、 $\Omega(n^c)$ 時間のあいだ、 $\Theta(c\log n)$ 時間ごとに全個体が同期をとってフェイズと呼ばれる変数を（ある大きな値 M の modulo で）1 ずつ増加させていくことを高確率で保証する。
- 提案アルゴリズムは $\Theta(\sqrt{n})$ フェイズごとに上記の衝突判定アルゴリズムを実行する。 $\Theta(\sqrt{n})$ フェイズは $\Theta(c\sqrt{n}\log n)$ 時間に相当するので、識別子に衝突があれば、衝突判定アルゴリズムは高確率で衝突を検知する。衝突がなければ、このアルゴリズムは衝突を誤検知することはない。
- 衝突判定アルゴリズムが衝突を検知した場合には、全個体は $\Theta(\sqrt{n})$ フェイズのあいだ衝突判定アルゴリズムを停止し、以降で説明する名前付けアルゴリズムを実行する。この名前付けアルゴリズムは $\Theta(\sqrt{n})$ フェイズで定数確率で名前付けを完了する。失敗した場合にも、その後、衝突判定アルゴリズムが期待時間 $\Theta(\sqrt{n})$ ごとに高確率で衝突を検知し、あらためて名前付けアルゴリズムを実行することになるので、問題とならない。
- 名前付けアルゴリズムは、最初の 1 フェーズで全個体の識別子を高確率で消失させる。（変数 **rank** に特別な値 0 を割り当てる。）
- $\tau = \lfloor \sqrt{n_U} \rfloor - 1$ とする。名前付けアルゴリズムでは、各個体 v は変数 $v.\text{Right} \in [1, n_U]$ を持つ。名前付けアルゴリズムは、次の $\Theta(\tau)$ フェイズで、 τ 個の個体の $(\text{rank}, \text{Right})$ にそれぞれ $(1, \tau), (\tau + 1, 2\tau), \dots, (\tau(\tau - 1) + 1, \tau^2)$ を高確率で割り当てる。この作業は、リーダが識別子を持たない個体に出会うたびに順に上記の値を割り当てることで簡単に実現できる。
- 名前付けアルゴリズムは、次の $\Theta(\tau \log n)$ フェイズで、少なくとも $n - (n_U - \tau^2) = n - \Theta(\tau)$ 個の個体に識別子を定数確率で割り当てる。具体的には、 $v.\text{rank} < v.\text{Right}$ を満たす個体 v が $u.\text{rank} = 0$ である個体と交流するとき、 $m = \lfloor (v.\text{rank} + v.\text{Right})/2 \rfloor$ 、 $r = v.\text{Right}$ として $(v.\text{rank}, v.\text{Right})$ を $(v.\text{rank}, m)$ に、 $(u.\text{rank}, u.\text{Right})$ を $(m+1, r)$ に設定する。アルゴリズムのつくりから、いかなる 2 個体 u, v についても、区間 $[u.\text{rank}, u.\text{Right}]$ と $[v.\text{rank}, v.\text{Right}]$ は交わりを持たないので、名前付けアルゴリズムは異なる 2 個体に同じ名前を割り当てることはない。 $\Theta(\tau \log n)$ フェイズ以内にこの処理が定数確率で完了するのは、この作業に要するステップ数の期待値が高々 $\sum_{i=1}^{\log \tau} \sum_{j=1}^{\tau^{2^i}} \frac{n(n-1)}{2^j \cdot \Theta(\tau)} = O(n^{1.5} \log^2 n)$ であることによる。
- 名前付けアルゴリズムは、次の $\Theta(\tau)$ フェイズで残りの高々 $O(\tau)$ 個の識別子を持たない個体に識別子を高確率で割り当てる。具体的には、この時点で $\tau^2 + 1, \tau^2 + 2, \dots, n_U$ の値はどの個体にも識別子として割り当てられていないので、それらの識別子を 1 フェイズごとに順に識別子を持たない個体に割り当てる。具体的には、各 $i = 1, 2, \dots, n_U - \tau^2$ について、 i フェイズ目に識別子を持たない個体が乱数の生成および疫病 [4] によるその最大値の伝播を行うことで、最大の乱数を生成した個体が識別子 $i + \tau^2$ を獲得すれば良い。乱数の生成範囲を $[1, n_U^2]$ 程度の大きさにしておけば最大の乱数を生

成する個体が高確率でひとつであることがいえるので、高確率ですべての個体に識別子を割り当てることができる。

参考文献

- [1] Dan Alistarh, James Aspnes, David Eisenstat, Rati Gelashvili, and Ronald L Rivest. Time-space trade-offs in population protocols. In *Proceedings of the twenty-eighth annual ACM-SIAM symposium on discrete algorithms*, pages 2560–2579. SIAM, 2017.
- [2] Dan Alistarh, James Aspnes, and Rati Gelashvili. Space-optimal majority in population protocols. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2221–2239. SIAM, 2018.
- [3] Dan Alistarh and Rati Gelashvili. Polylogarithmic-time leader election in population protocols. In *Automata, Languages, and Programming: 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part II 42*, pages 479–491. Springer, 2015.
- [4] Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Computing*, 18(4):235–253, 2006.
- [5] Petra Berenbrink, Dominik Kaaser, and Tomasz Radzik. On counting the population size. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 43–52, 2019.
- [6] Janna Burman, Ho-Lin Chen, Hsueh-Ping Chen, David Doty, Thomas Nowak, Eric Severson, and Chuan Xu. Time-optimal self-stabilizing leader election in population protocols. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*, pages 33–44, 2021.
- [7] Shukai Cai, Taisuke Izumi, and Koichi Wada. How to prove impossibility under global fairness: On space complexity of self-stabilizing leader election on a population protocol model. *Theory of Computing Systems*, 50(3):433–445, 2012.
- [8] Othon Michail, Paul G Spirakis, and Michail Theofilatos. Simple and fast approximate counting and leader election in populations. In *Stabilization, Safety, and Security of Distributed Systems: 20th International Symposium, SSS 2018, Tokyo, Japan, November 4–7, 2018, Proceedings*, pages 154–169. Springer, 2018.
- [9] Yuichi Sudo, Ryota Eguchi, Taisuke Izumi, and Toshimitsu Masuzawa. Time-optimal loosely-stabilizing leader election in population protocols. In *35th International Symposium on Distributed Computing (DISC 2021)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2021.
- [10] Yuichi Sudo and Toshimitsu Masuzawa. Leader election requires logarithmic time in population protocols. *Parallel Processing Letters*, 30(01):2050005, 2020.
- [11] 新谷拓海. 個体群プロトコルにおける時間・空間効率の良い衝突判定. 法政大学 卒業論文, 2023.