Vol.2022-AL-188 No.10
2022/5/20

グリッド上のコータリーの分散遷移問題

坂本 拓馬[1,a)]　山内 由紀子[1,b)]

概要：2 つの離散構造の実行可能解とその隣接関係が与えられたとき，隣接する実行可能解への遷移を繰り返すことで，ある実行可能解から別の実行可能解へ遷移可能か判定する問題を遷移問題という．本研究では分散システム上でのコータリーの遷移問題を提案する．コータリーは分散システムを構成するプロセスの部分集合族で，**コータリー**が含む各部分集合を**コーラム**と呼び，どの 2 つのコーラムの共通部分も空集合ではなく，一方のコーラムがもう一方のコーラムを包含しないという条件を満たす．本研究ではまず，コーラムの数に制約を与えた場合とコータリーの種類に制約を与えた場合について遷移可能性を考察する．次に，グリッドコータリーと呼ばれるコータリーに対する遷移列を生成するアルゴリズムを与える．

Distributed Reconfiguration Problem of Coteries on a Grid

1. はじめに

グラフの様々な最大化（最小化）問題に対して，**実行可能解**を発見する**探索問題**や実行可能解をすべて列挙する**列挙問題**が考えられてきた．これに対し，**遷移問題** [3] は初めに離散構造とそのインスタンス，**初期解**と**目標解**と呼ばれる実行可能解，実行可能解間の**隣接関係**の定義が与えられたとき，初期解と目標解の間を初期解から隣接する実行可能解を段階的に遷移して目標解に遷移可能かを判定する問題である．伊藤ら [3] によって，独立点集合や頂点被覆の遷移問題が P-SPACE 完全であることなど，多くの問題に対してその計算複雑性が明らかにされている．

分散システムは**通信リンク**を介して通信しながら協調動作する**プロセス**の集合である．各プロセスは分散システム全体の状況を知らず，通信リンクで接続された隣接プロセスと通信を行いながら，**局所変数**を更新することで計算を行う．自律的に分散システムを正しく効率的に動作させるために様々な離散構造が用いられている．たとえば，最小全域木は通信のバックボーン，頂点彩色は無線通信の帯域割り当てに用いられている．分散システム特有の局所性，並列性，非同期性を考慮した**分散遷移問題**が既にいくつか提案されている．Keren らは分散システム上における極大独立点集合の遷移問題を提案した [5]．頂点集合 S が，すべての頂点について距離 k 以内に S に含まれる頂点を持つとき，s を k-dominating という．実行可能解を 4-dominating を満たす独立点集合として，2 つの異なる実行可能解 S, S' は 1 回の遷移の間に除去または追加された頂点の集合が，与えられたグラフ上で独立点集合になっているとき隣接するとした．このとき，長さ 28 の遷移列を構成できることが証明されている．一方で，3-dominating にした場合について，4-dominating の場合よりも制約が強くなり常に遷移列は存在するが，遷移列の発見のための通信量と計算量に加え，遷移を終えるまでの遷移回数が定数ではなくなり $\Omega(n)$ になることが証明された．更に，実行可能解を極大独立点集合とした場合について，遷移不可能な場合が存在することが証明された．Marthe らは同期システムにおける頂点彩色遷移問題を提案した [1]．実行可能解を同色の頂点が隣り合わないこととして独立点集合の分散遷移問題と同様に，隣接する 2 つの実行可能解について，複数の頂点で色が異なっても良いが，色が異なる頂点同士は隣り合わないとした．グラフクラスを限定して，木，頂点の次数が最大 3 であるサブキュービックグラフ，辺が交差しないような頂点の配置が存在するトロイダルグラフに対して，与えられた実行可能解に用いられた色に，新たに色を 1 色追加することで頂点彩色遷移問題を解く方法が提案された．

本研究では分散システム上の**コータリー**の分散遷移問題を考える．コータリーは**コーラム**と呼ばれるプロセスの部分集合の集合，つまりプロセス集合の集合族であり，どの 2 つのコーラムについても，(i) 共通部分が空集合ではな

[1] 九州大学大学院システム情報科学府
福岡県福岡市西区元岡 744
[a)] sakamoto.takuma.406@s.kyushu-u.ac.jp
[b)] yamauchi@inf.kyushu-u.ac.jp

く，(*ii*) 一方が他方を包含することはない，という 2 つの条件を満たすものである．分散システムでは複数のプロセスが同時にアクセスすることを許さない**クリティカルセクション**と呼ばれる共有資源を持つことがある．自律的に動作する各プロセスが，クリティカルセクションを同時に使用しないことを保証する問題を**相互排除問題**という．コータリーはこの相互排除問題を解く際にに用いられる．相互排除アルゴリズムは様々なものが提案されているが，コータリーを用いると耐故障性とメッセージ複雑度を改善できることが知られている [4]．本研究では分散システムを停止させずにコータリーを別のコータリーに構成し直す問題をコータリーの分散遷移問題として解くことを目指した．

遷移問題における隣接関係のモデルはいくつか提案されいてる [3]．独立点集合を例として，独立点集合に含まれる頂点を 1 つ除去して，別の頂点を追加して得られる独立点集合と隣接するという隣接関係の設定などが挙げられる．コータリーに対する遷移問題は先行研究が無かったため新たに問題設定を行った．特に隣接関係を設定するにあたって，分散システム上でコータリーが局所変数として存在することと，隣接関係を単純化することを考慮して．2 つの異なるコータリー C と C' が 1 つのプロセス P を 1 つのコーラム Q に追加する，または Q から除去することで得られる場合に，C と C' は隣接するとした．

本研究の成果を説明する．初めに遷移によってコーラムの数が不変であることを示す．次に，あるプロセス P に対して，$Singleton = \{\{P\}\}$ によって定義される**シングルトンコータリー**というコータリーが常に遷移可能であり，すべての過半数のコーラムから成る**マジョリティコータリー**というコータリーが遷移不可能であることを示す．また，プロセス数を n，コーラムの数を q としたとき，$q = 1$ を満たしていれば常に遷移可能であり，$q = n - 1$ のとき遷移不可能な場合が存在することを示す．以上より $q = n - 1$ よりコーラムが少なければ常に遷移可能であると予想し，計算機実験でプロセス数が $3, 4, 5$ の場合に予想通りの結果が得られたことを報告する．最後に，プロセスの集合を $\{P_{i,j} | 1 \leq i \leq l, 1 \leq j \leq m\}$ として，$Q_{i,j} = \{P_{h,k} | h = i$ または $k = j\}$，$Grid = \{Q_{i,j} | 1 \leq i \leq l, 1 \leq j \leq m\}$ によって定義される**グリッドコータリー**について，2 種類の遷移列を組み合わせて，コーラムの数が変わらないという条件の下で常に遷移可能であることを示す．

2. 準備

2.1 遷移問題

ある離散構造とそのインスタンスに加えて実行可能解の定義と実行可能解間の隣接関係の定義を設定したとき，2 つの実行可能解 C_α，C_β の間を，隣接する実行可能解の系列で遷移可能かを判定する問題を**遷移問題**という．C_α を**初期解**，C_β を**目標解**と呼ぶ．初期解から目標解に遷移するまでに隣接する実行可能解へ遷移した回数を**遷移回数**と呼ぶ．つまり，遷移問題は初期解 C_α から目標解 C_β の間に，以下のような遷移列が存在するかを判定する問題である．

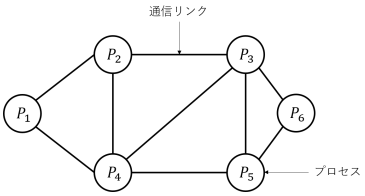

図 1 分散システム

$$C_\alpha = C_0, C_1, ..., C_k = C_\beta$$

ここで，C_i と C_{i-1} $(i = 1, 2, ..., k)$ は隣接する実行可能解であり，k は遷移回数である．また，遷移問題は，実行可能解を頂点，隣接関係を辺としたグラフに初期解と目標解に対応する頂点間に道が存在するかを判定する問題としても表せる．

2.2 分散システム

分散システムは通信リンクを介して通信しながら動作するプロセスの集合である [4]．ここで，**プロセス**とは分散システムにおける計算主体である．プロセスの集合を U，通信リンクの集合を E として分散システムをグラフ $G(U, E)$ として表すことができる．

各プロセス P_i は個別の識別子 $id(P_i)$ を持ち，各プロセスの局所変数を管理している．あるプロセスがほかのプロセスの持つ情報を得るためには通信リンクを介してメッセージの送信，受信を行う．2 つのプロセス間の直接通信は通信リンクで接続されている場合だけ可能である．各プロセスのアルゴリズムの集合を**分散アルゴリズム**と言い，各プロセスは分散アルゴリズムに従ってメッセージの送受信と内部計算，つまり自身の局所変数の値の更新を行う．

分散システムは**同期システム**と**非同期システム**に分類できる [4]．同期システムではメッセージの受信，内部計算，メッセージの送信を行う**ラウンド**をすべてのプロセスが同期して実行する．非同期システムではメッセージの通信遅延や内部計算の速度に一切の仮定を置かない．

2.3 コータリー

本節ではコータリー (coterie) の定義を与え，代表的なコータリーを紹介する．

定義 1. *[4]* U をプロセスの集合として，C が以下の *2* つの条件を満たす U の部分集合 Q_i の集合であるとき，C をコータリーという．

(i) すべての $1 \leq i, j \leq N$ に対して $Q_i \cap Q_j \neq \phi$

(ii) すべての $1 \leq i, j \leq N (i \neq j)$ に対して $Q_i \not\subseteq Q_j$

コータリーの定義における各 Q_i は**コーラム** (quorum)

と呼ばれる．コータリーに含まれるコーラムの数について次の定理が成り立つことが知られている．

定理 1. *[2]* コーラムに含まれるコータリーの数の上界はプロセス数が n のとき， 2^{n-1} である．

本研究では以下のコータリーを扱う [4].

マジョリティコータリー $|U| = n$ としてサイズが $k = \lfloor n/2 \rfloor + 1$ となる U のすべての部分集合からなる集合.

シングルトンコータリー $P \in U$ を任意のプロセスとして，P だけを含むコーラムからなる集合 Singleton=$\{\{P\}\}$.

グリッドコータリー プロセスの集合が論理的に 2 次元格子を形成しているとする．すなわち，$U = \{P_{i,j} | 1 \leq i \leq l, 1 \leq j \leq m\}$ とする．ただし $l, m \geq 2$ とする．このとき任意の $1 \leq i \leq l, 1 \leq j \leq m$ に対して $Q_{i,j} = \{P_{h,k} | h = i$ または $k = j\}$ として，$Grid = \{Q_{i,j} | 1 \leq i \leq l, 1 \leq j \leq m\}$ とする．このようなコータリーを l **行** m **列のグリッドコータリー**と呼ぶ.

3. コータリーの分散遷移問題

コータリーの分散遷移問題を以下のように定義する．

定義 2. プロセスの集合 U と U 上の *2* つのコータリー C_α, C_β が与えられたとき，遷移列 $\langle C_\alpha = C_0, C_1, ..., C_k = C_\beta \rangle$ が存在するか判定する．実行可能解の条件はコータリーであることとする．隣接関係について，C_i は C_{i-1} から *1* つのプロセス $P \in U$ を $P \notin Q$ を満たす *1* つのコーラム Q に追加する，または $P \in Q$ を満たす *1* つのコーラム Q から除去することで得られるものとする．

このような隣接関係に設定するのが妥当と考えた理由について説明する．1 回の遷移で 1 つのプロセスだけでの変更のみを許すこととしたのは，非同期分散システムでは異なる 2 つのプロセスが属するコーラムを同時に変更するのが困難だからである．1 つのコーラムからの除去，または 1 つのコーラムへの追加に限定したのは，隣接関係を単純にするためと，そのように限定しても，証明は省略するが以下の定理から遷移可能性に影響がないことが示せるからである．

定理 2. プロセス P を含むコーラムの集合がある 1 つのプロセス P だけについて異なる 2 つのコータリー C_α, C_β がそれぞれ，初期解，目標解として与えられたとき，常に C_α から C_β に遷移可能である．

定理 2 より，一般のコータリーの遷移問題に対して 1 つのプロセスを複数のコーラムへ追加または複数のコーラムから除去を許した遷移列が存在したとき，各遷移をプロセスが属するコーラムの差分を 1 つずつ除去して追加を行う遷移に置き換えることで，1 つのコーラムへの追加または 1 つのコーラムからの除去に限定しても，常に遷移列が存在する．したがって，除去と追加を 1 つのコーラムに限定することは一般の場合で遷移可能性に影響がない．

本章では，本研究での問題設定においてコーラム数が遷移によって変化しないことを示す．次にその性質を用いてコーラム数に着目して遷移可能性を考察した結果と，代表的なコータリーついて遷移可能性を考察した結果を示す．

3.1 コーラム数の不変性

一般に，あるプロセスの集合に対してコーラム数が異なる複数のコータリーを構成することができる．しかし，本研究の問題設定では隣接関係の設定より，1 回の遷移で 1 つのプロセスについてしか変更できないことから，コーラム数が異なるコータリー間の遷移が不可能であることが示せる．

定理 3. コーラム数が異なる *2* つのコータリーは隣接しない.

証明. コーラム数が異なる 2 つのコータリー C, C' を考え，C と C' は隣接していると仮定する．$|C| \neq |C'|$ であり，一般性を失わず $|C| < |C'|$ とする．この時，$Q \notin C$ かつ $Q \in C'$ を満たすコーラム Q が存在する．Q と異なるコーラム $Q' \in C'$ を選ぶと，コータリーの定義より $Q' \cap Q \neq \phi$ を満たすから，$P \in Q \cap Q'$ となるプロセス P が少なくとも 1 つ存在する．同様にコータリーの定義より $Q \not\subseteq Q'$ を満たすから $P' \in Q \setminus Q'$ を満たすプロセス P' が少なくとも 1 つ存在する．$P \in Q \cap Q'$ より $P \in Q'$ が成り立つ．また，$P' \in Q \setminus Q'$ より $P' \notin Q'$ が成り立つ．したがって $P \neq P'$ より $|Q| \geq 2$ となる．以上より，C と C' が隣接するにはコーラム Q に少なくとも 2 つ以上のプロセスを同時に追加または除去する必要があるが，C と C' が隣接するのは 1 つのプロセスを $Q \in C$ に追加または除去して C' が得られる場合だけだから矛盾する．したがって，C と C' は隣接しない． □

定理 3 より，コータリーの任意の遷移列 $\langle C_\alpha = C_0, C_1, ..., C_k = C_\beta \rangle$ において，C_{i-1} と C_i $(i = 1, 2, ...k)$ は隣接しているから $|C_{i-1}| = |C_i|$ を満たしており，$|C_\alpha| = |C_\beta|$ である．したがって遷移可能になるのは遷移前と遷移後でコーラム数が等しいときだけになる．以降は実行可能解のコーラム数を q とおいて，$q = |C_\alpha| = |C_\beta|$ とする．

3.2 限定的なコータリーの考察

初期解と目標解のコータリーを限定したときの遷移可能性について考察した結果を示す.

定理 4. $|U| \geq 3$ のとき，マジョリティコータリーは隣接する実行可能解が存在しない.

証明. 定義より，マジョリティコータリーはプロセス数が $k = \lfloor n/2 \rfloor + 1$ となるすべてのコーラムを含む．任意のコー

ラム Q から任意のプロセス $P \in Q$ を 1 つ除去して得られる集合 $Q \setminus \{P\}$ は，$P' \in U \setminus Q$ を追加した $Q \setminus \{P\} \cup \{P'\}$ という集合を考えると，この集合はプロセス数が $k = \lfloor n/2 \rfloor + 1$ であるからマジョリティコータリーに含まれている．この集合は $(Q \setminus \{P\}) \subset (Q \setminus \{P\} \cup \{P'\})$ を満たすからコータリーの条件を満たさなくなる．任意のコーラム Q に任意のプロセス $P \in U \setminus Q$ を 1 つ追加して得られる集合 $Q \cup \{P\}$ は，P' を $P' \in Q$ として $Q \cup \{P\} \setminus \{P'\}$ という集合を考えると，この集合はプロセス数が $k = \lfloor n/2 \rfloor + 1$ であるからマジョリティコータリーに含まれていて，$(Q \cup \{P\} \setminus P') \subset (Q \cup \{P\})$ となるからコータリーの条件を満たさなくなる．以上より任意のコーラムに対して追加または除去すると必ずコータリーの条件を満たさなくなるからマジョリティコータリーは隣接する実行可能解を持たない． □

シングルトンコータリーについて以下が成り立つ

定理 5. コータリーの分散遷移問題における初期解と目標解がシングルトンコータリーであるとき，常に遷移可能である．

証明. 2 つの異なるシングルトンコータリーを $C_\alpha = \{\{P_i\}\}, C_\beta = \{\{P_j\}\}$ $(i \neq j)$ とする．$\{\{P_i, P_j\}\}$ は $\{\{P_i\}\}$ のコーラム $\{P_i\}$ が含むプロセス P_j を追加して得られるから $\{\{P_i, P_j\}\}$ と $\{\{P_i\}\}$ は隣接している．また，$\{\{P_j\}\}$ は $\{\{P_i, P_j\}\}$ が含むコーラム $\{P_i, P_j\}$ からプロセス P_j を除去して得られるので，$\{\{P_j\}\}$ と $\{\{P_i, P_j\}\}$ は隣接している．以上より $C_\alpha = \{\{P_i\}\}$，$\{\{P_i, P_j\}\}$，$C_\beta = \{\{P_j\}\}$ の順に遷移すればいい． □

シングルトンコータリーの場合を一般化して $q = 1$ のときを考える．

定理 6. コータリーの分散遷移問題における初期解と目標解のコーラム数が $q = 1$ であるとき，常に遷移可能である．

証明. 初期解を $C_0 = C_\alpha = \{Q_\alpha\}$，目標解を $C_\beta = \{Q_\beta\}$ とする．$P \in Q_i \setminus Q_\alpha$ を満たすプロセス P を Q_α に 1 つ 追加して Q_{i+1} とする．この操作を，追加するプロセスがなくなるまで繰り返すと，$a = |Q_\beta \setminus Q_\alpha|$ として $C_a = \{Q_\alpha \cup Q_\beta\}$ となる．ここから $P \in Q_\alpha \setminus Q_\beta$ を満たすプロセス P の除去を繰り返して $b = |Q_\alpha \setminus Q_\beta|$ とすれば $C_{a+b} = C_\beta$ に遷移する． □

定理 6 と定理 4 の結果から，コーラム数が 1 と極端に少ない場合と，コーラム数が $\binom{n}{\lfloor n+1/2 \rfloor}$ と極端に多いマジョリティコータリーの場合についての遷移可能性を示せたが，その間のコーラム数での遷移可能性を示すのは困難だった．本研究では $n-1$ のとき遷移不可能な組が構成できることを示したが，残りについて予想と計算機実験を行った．

定理 7. コータリーの分散遷移問題において，プロセス数が n，コーラム数が $q = n-1$ のとき，遷移不可能な初期解と目標解の組が存在する．

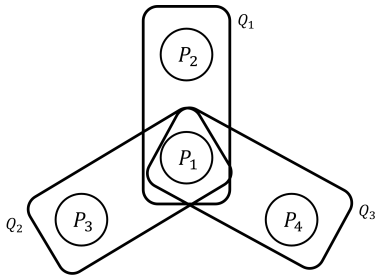

図 2 定理 7 のコータリーの例

証明. コーラム数が $n-1$ のコータリーについて各コーラムを $Q_i = \{P_1, P_{i+1}\} (i = 1, 2, ..., n-1)$ とするコータリー C を考える．コータリーの条件は $P_1 \in Q_i \cap Q_j$ より条件 (i) を満たす．$P_i \in Q_i \setminus Q_j$ であるから，$Q_i \not\subset Q_j$ となって，条件 (ii) を満たす．初期解を $C_\alpha = \{\{P_1, P_{i+1}\}\} (i = 1, 2, ..., n-1)$，目標解を C_α 以外の任意のコータリーとする．C_α が含むコーラム Q_i にプロセス $P_j (j \neq 1, i+1)$ を追加すると，$Q_i \subset Q_j$ となるから，条件 (ii) を満たさなくなる．Q_i から P_1 を除去すると $Q_i = \{P_{i+1}\}$ になるが $Q_i \cap Q_j = \phi$ となるから条件 (i) を満たさなくなる．P_{i+1} を除去すると $Q_i = \{P_1\}$ は $Q_i \subset Q_j$ になるから条件 (ii) を満たさなくなる．以上より C_α から他の実行可能解に遷移することができない． □

3.3 予想と計算機実験

3.2 の結果から遷移可能性についてコーラム数が少ないほうが遷移可能になる可能性が高いと考えた．$1 < q < n-1$ については常に遷移可能であると予想し，$n-1 \leq q$ については遷移不可能な初期解と目標解の組が存在すると予想した (**表 1**)．定理 1 より，$|C| \leq 2^{n-1}$ とした．

表 1 コーラム数と遷移可能性

コーラム数	遷移可能性
1	常に可能
$1 < q < n-1$	常に可能と予想
$n-1$	遷移不可能な場合がある
$n \leq q \leq 2^{n-1}$	遷移不可能な場合があると予想

$1 < q < n-1$ のときの予想に対して表 2 に示した環境で計算機実験を行った．計算機実験に用いたプログラムの概要を説明する．まず，与えられたプロセス数に対してすべてのプロセスの部分集合族を列挙し，それらのうちコータリーの条件を満たすものを抽出する．次に，求めたコータリーの集合に含まれるペアに対して，隣接関係を満たせばそのコータリー間に辺を追加する．結果として生成されるグラフが連結グラフであれば任意の初期解と目標解は遷移可能，そうでなければ遷移不可能な組が存在すると判定する．プロセス数を入力して $1 < q < n-1$ の範囲で常に遷移可能と判定されたかどうかを出力する．コータリー

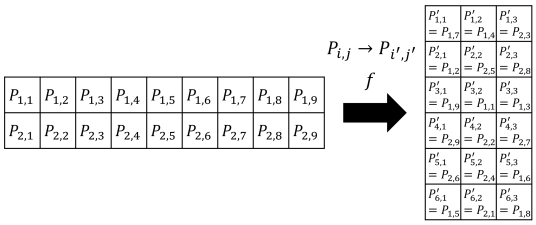

図 3 グリッドコータリーの遷移問題の例

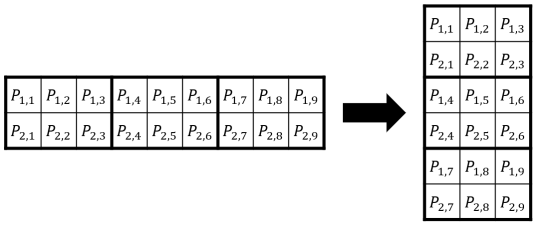

図 4 遷移 A

の通り数が膨大になったことからプロセス数が 5 以下の非常に小さい場合についてでしか結果が得られなかったが，結果は予想通りだった．表 3 に各プロセス数において，$1 < q < n-1$ の遷移可能性の結果とその計算時間を示した．

表 2 実験環境

OS	Windows 10 Home
CPU	Intel(R) Core(TM) i5-8250U
メモリ	8.00GB
使用言語	C++

表 3 計算機実験の結果

プロセス数	$1 < q < n-1$ の遷移可能性	計算時間 (sec)
3	常に可能	<0.0000
4	常に可能	0.0160
5	常に可能	17.922

4. グリッドコータリーの分散遷移問題

本章ではグリッドコータリーの遷移問題を考える．グリッドコータリーはコーラム数とプロセス数が等しいので，3 章で遷移不可能な組が存在すると予想した範囲に含まれる．本章では，コーラム数とプロセス数が一致する場合に，遷移列を与えるアルゴリズムを示す．

初期解 C_α を l_α 行 m_α 列のグリッドコータリー，目標解 C_β を l_β 行 m_β 列のグリッドコータリーとする．ただし，グリッドコータリーの定義より，$l_\alpha, m_\alpha, l_\beta, m_\beta \geq 2$ が成り立つ．定理 3 より，$|C_\alpha| = |C_\beta|$ となる必要があることから $l_\alpha m_\alpha = l_\beta m_\beta$ が成り立つ．つまり，ある有理数 x が存在し，$l_\beta = l_\alpha x, m_\beta = m_\alpha / x$，かつ，$l_\alpha, m_\alpha, l_\beta, m_\beta$ が全て自然数にとなる．初期解でのプロセスの集合を $P_\alpha = \{P_{i,j} | 1 \leq i \leq l_\alpha, 1 \leq j \leq m_\alpha\}$，目標解でのプロセスの集合を $P_\beta = \{P'_{i,j} | 1 \leq i \leq l_\beta, 1 \leq j \leq m_\beta\}$ と表す．P_α と P_β の間には関数 f で一対一対応が与えられているとする．初期解は各コーラムを $Q_{i,j} = \{P_{h,k} | h = i \text{ または } k = j\}$ として

$$C_\alpha = \{Q_{i,j} | 1 \leq i \leq l_\alpha, 1 \leq j \leq l_\beta\},$$

目標解は各コーラムを $Q'_{i,j} = \{P'_{h,k} | h = i \text{ または } k = j\}$ として

$$C_\beta = \{Q'_{i,j} | 1 \leq i \leq l_\beta, 1 \leq j \leq m_\beta\}$$

とする．**図 3** に，初期解と目標解の例を示す．

提案アルゴリズムでは 2 つの遷移列によってこの問題を解く．最初の遷移列は，整数 x に対して，l 行 m 列のグリッドコータリーを lx 行 m/x 列のグリッドコータリーに遷移させる系列である．次の遷移列は，グリッドコータリーコータリーの行数，列数は保ったまま，2 つのプロセスの位置を入れ替える操作を目標解に一致するまで行う遷移列である．提案アルゴリズムでは以上の 2 つの遷移列を順に行うことで，初期解から目標解へ到達する遷移列を生成する．最初の遷移列を**遷移** A，次の遷移列を**遷移** B と呼ぶ．以降では，遷移 A と遷移 B を説明し，2 つの遷移列を組み合わせて初期解から目標解までの遷移列を構成する方法を説明する．

4.1 遷移 A

x を自然数として l_α 行 m_α 列のグリッドコータリー C を $l_\beta = x l_\alpha x$ 行 $m_\beta = m_\alpha / x$ 列のグリッドコータリー C^* に遷移させる遷移 A を与える．**図 4** の例を用いて概要を説明する．遷移 A では，図 4 の太線で示したように，グリッドコータリーを横に x 個に分割して縦に積み重ねることで，l_β 行 m_β 列のグリッドコータリーに変形する．図 4 の左のグリッドコータリーから右のグリッドコータリーに遷移させるとき，例えば，$P_{1,4}$ を中心として 1 行目，4 列目のプロセスの集合から成るコーラム $Q_{1,4}$

$$\{P_{1,1}, P_{1,2}, P_{1,3}, P_{1,4}, P_{1,5}, P_{1,6}, P_{1,7}, P_{1,8}, P_{2,4}\}$$

は右のグリッドコータリー上では $P_{1,4}$ を中心として 3 行目 1 列目のプロセスの集合から成るコーラム

$$\{P_{1,1}, P_{2,1}, P_{1,4}, P_{2,4}, P_{1,7}, P_{2,7}, P_{1,5}, P_{1,6}\}$$

に更新する．初期解のすべてのコーラムに対して，コータリーの条件を満たしながら同様の更新を施す．

遷移 A の適用前のグリッドコータリーで $P_{h,k}$ に対応するコーラム，つまり，h 行もしくは k 列に含まれるプロセスの集合から成るコーラムを $Q_{h,k}$ とし，適用前のグリッドコータリーの分割を以下のように表す．

$$sub_p = \{Q_{h,k} | 1 \leq h \leq l_\alpha \wedge (p-1)m_\beta < k \leq p m_\beta\}$$

ただし p の範囲は $p = 1, 2, ..., x$ である．遷移 A の適用後

$P_{1,1}$	$P_{1,2}$	$P_{1,3}$	$P_{1,4}$	$P_{1,5}$	$P_{1,6}$
$P_{2,1}$	$P_{2,2}$	$P_{2,3}$	$P_{2,4}$	$P_{2,5}$	$P_{2,6}$
$P_{3,1}$	$P_{3,2}$	$P_{3,3}$	$P_{3,4}$	$P_{3,5}$	$P_{3,6}$

図 5 $Q_{1,4}$ の遷移 (1)

$P_{1,1}$	$P_{1,2}$	$P_{1,3}$	$P_{1,4}$	$P_{1,5}$	$P_{1,6}$
$P_{2,1}$	$P_{2,2}$	$P_{2,3}$	$P_{2,4}$	$P_{2,5}$	$P_{2,6}$
$P_{3,1}$	$P_{3,2}$	$P_{3,3}$	$P_{3,4}$	$P_{3,5}$	$P_{3,6}$

図 6 $Q_{1,4}$ の遷移 (2)

$P_{1,1}$	$P_{1,2}$	$P_{1,3}$	$P_{1,4}$	$P_{1,5}$	$P_{1,6}$
$P_{2,1}$	$P_{2,2}$	$P_{2,3}$	$P_{2,4}$	$P_{2,5}$	$P_{2,6}$
$P_{3,1}$	$P_{3,2}$	$P_{3,3}$	$P_{3,4}$	$P_{3,5}$	$P_{3,6}$

図 7 $Q_{1,4}$ の遷移 (3)

Algorithm 1 遷移 A

```
グリッドコータリーの行数と列数を更新するアルゴリズム
C_{i,j} ← 各プロセス
Q_{i,j} ← C に含まれるコーラム
Q*_{i,j} ← C* に含まれるコーラム
for h ← 1 to l_α do
    for k ← 1 to m_α do
        for j = 1 to m_α do
            if P_{h,j} ∉ Q*_{i,j} then
                Remove(P_{h,j}, Q_{i,j})
            end if
        end for
        for p ← 1 to x do
            for i = 1 to l_α do
                if P_{i,k+(p−1)m_α} ∈ Q*_{i,j} then
                    Add(P_{i,k+(p−1)m_α}, Q_{i,j})
                end if
            end for
        end for
    end for
end for
```

のグリッドコータリーで $P_{h,k}$ を中心に $P_{h,k}$ と同じ行，同じ列のプロセスからなるコーラムを $Q^*_{h,k}$ とする．すべてのコーラムを $Q_{h,k}$ から $Q^*_{h,k}$ に更新する手順を説明する．$Q_{h,k}$ と $Q^*_{h,k}$ を比較して，初めに，$Q_{h,k}$ 含まれるが $Q^*_{h,k}$ に含まれないプロセスをすべて除去する．次に $Q^*_{h,k}$ 含まれるが $Q_{h,k}$ に含まれないプロセスの追加を行う．**図 5** から**図 7** はコーラムの更新の例である．$l_\alpha = 3, m_\alpha = 6$ から $l_\beta = 6, l_\beta = 3$ に遷移 A を適用したとき，$Q_{1,4}$ を更新する場合を示す．網掛け部分が $Q_{1,4}$ に含まれている．図 5 は更新前，**図 6** は除去を行った後， 図 7 は追加を行った後である．

提案アルゴリズムの疑似コードを Algorithm 1 に示す．$\mathrm{Remove}(P_{i,j}, Q_{h,k})$ によって，プロセス $P_{i,j}$ 除去したコーラム $Q_{h,k}$ から除去したに更新する．$\mathrm{Add}(P_{i,j}, Q_{h,k})$ によってプロセス $P_{i,j}$ を追加したコーラム $Q_{h,k}$ に更新する．

定理 8. *Algorithm 1* は，与えられた l_α 行 m_α 列のグリッドコータリーを l_β 行 m_β 列のグリッドへ遷移する遷移列を生成する．ただし，l_α，m_α，l_β，m_β はいずれも自然数で，$l_\alpha m_\alpha = l_\beta m_\beta$，自然数 x について $l_\beta = x l_\alpha$，$m_\beta = m_\alpha / m$ を満たす．

証明． 遷移 A が実効可能解の系列であることを示すために，sub_q に含まれるコーラム $Q_{h,k}$ に除去と追加を行っている途中の状態を考える．$Q_{h,k}$ にプロセスの除去と追加を行っている間，他のコーラムには除去も追加も行わないため，Q_i，Q_j $(i \neq h, j \neq k)$ について，$Q_i \cap Q_j \neq \phi$，$Q_i \not\subseteq Q_j$，$Q_j \not\subseteq Q_i$ が常に成り立つ．$Q_{h,k}$ と他のコーラム Q について $Q_{h,k} \cap Q \neq \phi$，$Q_{h,k} \not\subseteq Q$，$Q \not\subseteq Q_{h,k}$ が示せばよい．遷移 A はコーラムを 1 つずつ更新するため，遷移途中には各コーラムからの除去と追加が行われる前のプロセス，その途中のプロセス，その後のプロセスが存在する．

最初に, 更新前のコーラムを $Q_{i,j}$ と置いて, $Q_{h,k} \cap Q_{i,j} \neq \phi$ を満たしていることを示す．$Q_{i,j}$ は更新前であるから，遷移前のグリッドコータリー上で i 行目のプロセスと j 列目のプロセスをすべて含む．$Q_{h,k}$ は提案アルゴリズムより，更新の過程においても遷移前のグリッドコータリー上で k 列目に含まれるプロセスが除去されないから，常に k 列目のプロセスをすべて含む．行と列は一つのプロセスで必ず交わるため，遷移前のグリッドコータリー上で $Q_{i,j}$ は常に i 行目のプロセスをすべて含み，$Q_{h,k}$ は k 列目のプロセスをすべて含むことから，$Q_{h,k} \cap Q_{i,j} \neq \phi$ を満たす．

次に，更新前のコーラムを $Q_{i,j}$ と置いて，$Q_{h,k} \not\subseteq Q_{i,j}$ かつ $Q_{i,j} \not\subseteq Q_{h,k}$ を満たしていることを示す．プロセスを除去する過程と追加する過程に分けて考える．$Q_{h,k}$ からプロセスを除去する過程において，明らかに $Q_{i,j} \not\subseteq Q_{h,k}$ が成り立つ．$l_\alpha, m_\beta \geq 2$ であるから，プロセスを除去する過程においても，遷移前のグリッドコータリー上で $Q_{h,k}$ は常に h 行目のプロセスと k 列目のプロセスをそれぞれ 2 つ以上含む．グリッドコータリー上でのコーラムの定義より，$Q_{i,j}$ について，$i' \neq i$ 行目または $j' \neq j$ 列目に着目した時，それらの行または列に含まれるプロセスは $Q_{i,j}$ には 1 つしか含まれていない．$i \neq h$ または $j \neq k$ であるから，$Q_{h,k}$ がプロセスを 2 つ以上含み，$Q_{i,j}$ がプロセスを 1 つしか含まないような行または列が存在する．したがって，$Q_{h,k} \not\subseteq Q_{i,j}$ が成り立つ．$Q_{h,k}$ への追加の過程において，明らかに $Q_{h,k} \not\subseteq Q_{i,j}$．$Q_{h,k}$ は h 行目に対して除去を既に行ったため，遷移前のグリッドコータリー上ですべてのプロセスを含むような行はこの時点で存在しない．一方で $Q_{i,j}$ は遷移前であるから，遷移前のグリッドコータリー上ですべてのプロセスを含むような行が存在する．したがって，$Q_{i,j} \not\subseteq Q_{h,k}$ が成り立つ．

次に, 更新後のコーラムを $Q_{i,j}$ と置いて, $Q_{h,k} \cap Q_{i,j} \neq \phi$ を満たしていることを示す．プロセス $P_{i,j}$ の遷移後のグリッドコータリーコータリー上での位置を i^* 行，j^* 列目とおく．また，プロセス $P_{h,k}$ の遷移後のグリッドコータリー

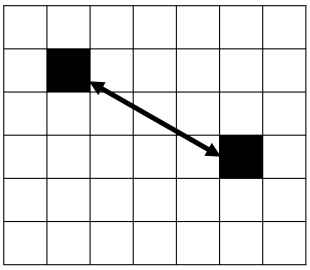

図 8 遷移 B

上での位置を h^* 行，k^* 列目とおく．$Q_{i,j}$ は更新後であるから，遷移後のグリッド上で i^* 行目のプロセスと j^* 列目のプロセスをすべて含む．$Q_{h,k}$ は提案アルゴリズムより，更新の過程においても遷移後のグリッドコータリー上で h^* 行目に含まれるプロセスが除去されないから，常に h^* 列目のプロセスをすべて含む．行と列は一つのプロセスで必ず交わるため，遷移後のグリッドコータリー上で $Q_{i,j}$ は常に i^* 行目のプロセスをすべて含み，$Q_{h,k}$ は k^* 列目のプロセスをすべて含むことから，$Q_{h,k} \cap Q_{i,j} \neq \phi$ を満たす．

最後に，更新後のコーラムを $Q_{i,j}$ と置いて，$Q_{h,k} \not\subseteq Q_{i,j}$ かつ $Q_{i,j} \not\subseteq Q_{h,k}$ を満たしていることを示す．プロセスを除去する過程と追加する過程に分けて考える．$Q_{h,k}$ からプロセスを除去する過程において，明らかに $Q_{i,j} \not\subseteq Q_{h,k}$ が成り立つ．$l_\alpha, m_\beta \geq 2$ であるから，除去の過程においても遷移後のグリッドコータリー上で $Q_{h,k}$ は常に h^* 行目のプロセスと k^* 列目のプロセスをそれぞれ 2 つ以上含む．グリッドコータリー上でのコーラムの定義より，$Q_{i,j}$ について，i^* 行目以外の行または j^* 列目以外の列に着目した時，プロセスが 1 つしか含まれていない．$i^* \neq h^*$ または $j^* \neq k^*$ であるから，$Q_{h,k}$ がプロセスを 2 つ以上含み，$Q_{i,j}$ がプロセスを 1 つしか含まないような行または列が存在する．したがって，$Q_{h,k} \not\subseteq Q_{i,j}$ が成り立つ．$Q_{h,k}$ への追加の過程において，明らかに $Q_{h,k} \not\subseteq Q_{i,j}$．$Q_{h,k}$ は h 行目に対して除去を既に行ったため，遷移後のグリッドコータリー上ですべてのプロセスを含むような行は 1 つだけ存在する．一方で $Q_{i,j}$ は遷移後であるから，遷移後のグリッドコータリー上ですべてのプロセスを含むような行と列が 1 つずつ存在する．$i^* \neq h^*$ または $j^* \neq k^*$ であるから $Q_{h,k}$ への追加を終えても，$Q_{h,k}$ が $Q_{i,j}$ を包含することはない．したがって，$Q_{i,j} \not\subseteq Q_{h,k}$ が成り立つ．

以上より，$Q_{h,k}$ は他のコーラムとの間で常に $Q_i \cap Q_j \neq \phi$ と $Q_i \not\subseteq Q_j$ の条件を満たす． □

4.2 遷移 B

遷移 B は 2 つのプロセスの位置を入れ替える遷移列である．2 つのプロセスの位置を入れ替えるためには，2 つの異なるプロセスを $P_{i,j}, P_{i^*,j^*} (i \neq i^* \text{ または } j \neq j^*)$ として，$P_{i,j}$ を含むコーラムの集合 $\{Q | P_{i,j} \in Q\}$ と P_{i^*,j^*} を含むコーラムの集合 $\{Q | P_{i^*,j^*} \in Q\}$ を入れ替える．入れ替える前のグリッドコータリーを C，入れ替えたあとのグリッドコータリーを C^* と置く．提案手法の方針は，プロセス $P_{i,j}$ を含むが遷移後のコータリーにおいてプロセス $P_{i,j}$ を含まないようなコーラムすべてから $P_{i,j}$ を順に除去し，同様の操作を P_{i^*,j^*} に対しても行う．次に，遷移後のコータリー C^* においてプロセス $P_{i,j}$ を含むが，遷移前のコータリー C においてプロセス $P_{i,j}$ を含まないようなコーラムすべてに $P_{i,j}$ を追加し，同様の操作を $P_{i',j'}$ に対しても行う．以上の操作で，$P_{i,j}$ を含むコーラムの集合と $P_{i',j'}$ を含むコーラムの集合が入れかわる．提案アルゴリズムの疑似コードを Algorithm 2 に示す．

Algorithm 2 遷移 B

```
2つのプロセスの位置を入れ替えするアルゴリズム
C_{i,j} ← 各プロセス
i, j ← 座標 1
i*, j* ← 座標 2
for all q ∈ {Q_{h,k} | h = i ∨ k = j} do
    if P_{i*,j*} ∉ q then
        Remove(P_{i,j}, q)
    end if
end for
for all q ∈ {Q_{h,k} | h = i* ∨ k = j*} do
    if P_{i,j} ∉ q then
        Remove(P_{i*,j*}, q)
    end if
end for
for all q ∈ {Q_{h,k} | h = i ∨ k = j} do
    if P_{i*,j*} ∈ q then
        Add(P_{i,j}, q)
    end if
end for
for all q ∈ {Q_{h,k} | h = i* ∨ k = j*} do
    if P_{i,j} ∈ q then
        Add(P_{i*,j*}, q)
    end if
end for
```

グリッドコータリーが l 行 2 列のときの同じ行に存在する 2 つのプロセスの入れ替え，2 行 m 列のときの同じ列に存在する 2 つのプロセスの入れ替えには提案手法は直接適用できない．この場合，別のプロセスとの入れ替えを経由し，遷移 B を 3 回行うことでプロセスの入れ替えを実現できる．**図 9** に例を示す．

定理 9. *Algorithm 2* は与えられたグリッドコータリーから，2 つのプロセスの位置を入れ替えたグリッドコータリーへの遷移列を生成する．ただし，2 つのプロセスはグリッドコータリーが l 行 2 列のときの同じ行のプロセスの組または 2 行 m 列のときの同じ列のプロセスの組ではないとする．

証明. 遷移 B が常に実効可能解を経由していることを示す．初めに，プロセスの位置を入れ替えた結果，そのプロセスが不要になるコーラムから除去する過程において常に実効可能解を経由しているを示す．グリッドコータリーの

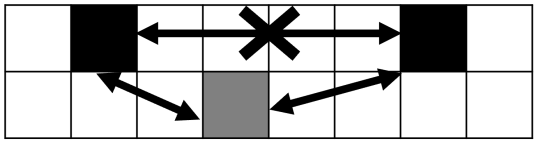

図 9 例外的なプロセスの入れ替え

定義より $|Q_{u,v} \cap Q_{u^*,v^*}| \geq 2$ であり，例外にした場合を除けば $|Q_{u,v} \setminus Q_{u^*,v^*}| \geq 2$ かつ $|Q_{u^*,v^*} \setminus Q_{u,v}| \geq 2$ である．$P_{i,j}$ を除去し終えた時点でどちらも高々 1 しか減らないからコータリーの条件 $Q_{u,v} \cap Q_{u^*,v^*} \neq \phi$，$Q_{u,v} \not\subseteq Q_{u^*,v^*}$，$Q_{u^*,v^*} \not\subseteq Q_{u,v}$ を満たす．

次に，プロセスの位置を入れ替えた結果，そのプロセスが必要になるコーラムに追加する間について $|Q_{u,v} \cap Q_{u^*,v^*}|$ は単調に増加するから明らかに $Q_{u,v} \cap Q_{u^*,v^*} \neq \phi$．グリッドコータリーの定義より，遷移後は $|Q_{u,v} \setminus Q_{u^*,v^*}| \geq 2$ かつ $|Q_{u^*,v^*} \setminus Q_{u,v}| \geq 2$ を満たすから，$Q_{u,v} \not\subseteq Q_{u^*,v^*}$，$Q_{u^*,v^*} \not\subseteq Q_{u,v}$ を満たす． □

遷移 A はグリッドコータリーの行数と列数を整数倍で変換し，遷移 B は行数や列数を変えずにプロセスの配置を変更する．Algorithm 3 はこれらの遷移を用いて，初期解から目標解に到達するような遷移列を生成する．$l_\beta = l_\alpha x, m_\beta = m_\alpha / x$ の時，x は有理数になる可能性がある．遷移 A は列数を整数倍にする遷移しか与えていないから，このような場合には $x = a/b$ とおくと，まず $m_\alpha b$ 行 m_α / b 列のグリッドコータリーに遷移してから 90 度回転させて，m_α / b 行 $m_\alpha b$ 列のグリッドコータリーとみなす．次に，$l_\alpha a / b = l_\beta$ 行 $(m_\alpha / a)b = m_\beta$ 列のグリッドコータリーに遷移する．例えば，4 行 9 列から 6 行 6 列に遷移する場合，$x = 3$ として遷移 A を行って，12 行 3 列に遷移し，90 度回転させて 3 行 12 列とみなす．次に $x = 2$ として遷移 A を行い，6 行 6 列に遷移する．このようにして行数と列数をを目標解に一致させた後，遷移 B によりプロセスの位置を関数 f で与えられた移動先に一致させる．

Algorithm 3 グリッドコータリーの分散遷移問題を解くアルゴリズム

```
x ← a/b
Algorithm1(b)
if b ≠ 1 then
    グリッドコータリーを 90 度回転
    Algorithm1(a)
end if
for i = 1 to l_β do
    for j = 1 to m_β do
        i′, j′ ← 関数 f で i 行 j 列目のプロセスと対応する
        if (i, j) ≠ (i′, j′) then
            if l_β = 2 かつ j = j′ then
                u, v ← i ≠ u かつ j ≠ v を満たす座標
                Algorithm2((i, j), (u, v))
                Algorithm2((u, v), (i′, j′))
                Algorithm2((i, j), (u, v))
            end if
        else
            if m_β = 2 かつ i = i′ then
                u, v ← i ≠ u かつ j ≠ v を満たす座標
                Algorithm2((i, j), (u, v))
                Algorithm2((u, v), (i′, j′))
                Algorithm2((i, j), (u, v))
            else
                Algorithm2((i, j), (i′, j′))
            end if
        end if
    end for
end for
```

5. おわりに

本研究ではコータリーの分散遷移問題について，コーラム数とプロセス数の関係に制約を与えて遷移可能性について解析した結果と，グリッドコータリーに制限したときの遷移の手続きとその正当性を示した．グリッドコータリーの分散遷移問題を考察した結果から，コーラム数がプロセス数以上であっても，追加で制約を与えられたコータリー間であれば遷移可能になる可能性を示した．今回提案したアルゴリズムは分散アルゴリズムではないため，分散アルゴリズムとしての実現方法を検討している．遷移可能性の解析で得られた予想の証明も検討している．

参考文献

[1] Bonamy, M., Ouvrard, P., Rabie, M., Suomela, J. and Uitto, J.: Distributed Recoloring, *In Proceedings of the 32nd International Symposium on Distributed Computing*, pp. 12:1–12:17 (2018).

[2] Gracia-Molina, H. and Barbara, D.: How to Assign Votes in a Distributed System, *Journal of the Association for Computing Machinery*, Vol. 32, No. 4, pp. 841–860 (1985).

[3] Ito, T., Demaine, E. D., Harvey, N. J., Papadimitriou, C. H., Sideri, M., Uehara, R. and Uno, Y.: On the complexity of reconfiguration problems, *Theoretical Computer Science*, Vol. 412, pp. 1054–1065 (2011).

[4] 亀田恒彦, 山下雅史：分散アルゴリズム, 近代科学社 (1994).

[5] Censor-Hillel, K. and Rabie, M.: Distributed Reconfiguration of Maximal Independent Sets, *In Proceedings of the 46th International Colloquium on Automata, Languages, and Programming*, pp. 135:1–135:14 (2019).