匿名単方向リングネットワークにおけるモバイルエージェント集合問題に対するトークンを用いた乱択アルゴリズム

難波 瑛次郎[1] 大下 福仁[2] 角川 裕次[1] 増澤 利光[1]

1 大阪大学 大学院情報科学研究科

2 奈良先端科学技術大学院大学

概要　本稿では，匿名単方向リングネットワークにおいて，トークンを用いてモバイルエージェント集合問題を解くアルゴリズムについて考察する．集合問題では，初期状況において任意のノードに分散している複数のエージェントが有限時間内に同一のノードに集合することを目的とする．本稿では，匿名単方向リングネットワークにおいて，エージェントがノード数やエージェント数を知らないモデルを扱う．本モデルに対する既存手法では，各ノードに白板と呼ばれる十分な量のメモリを仮定することで，集合問題を高確率で解いている．それに対して，本稿では，各ノードに 1 ビットのトークンを置けるという仮定のもとで，集合問題に対する乱択アルゴリズムを提案する．シミュレーションにより，提案アルゴリズムはランダムな初期配置から高確率で集合問題を解くことを示す．また，決定性 アルゴリズムで集合問題を解けない周期的な初期配置に対しても，提案アルゴリズムにより確率的に集合問題が解けることを示す．

1　はじめに

分散システムとは，多数の計算機（以下，**ノード**）とそれらを繋ぐ通信リンク（以下，**リンク**）から構成されたシステムである．近年，分散システムは大規模化が進んでおり，分散システムの設計はますます困難になっている．そのため，大規模化・複雑化する分散システムを設計するための有効な手法として，現在，モバイルエージェント (以下，**エージェント**) を利用した設計法が注目を集めている．エージェントとは，ネットワーク中のノード間を移動しながら，自律的に動作するソフトウェアである．

エージェントシステムを用いた代表的な問題に集合問題がある [1]．集合問題とは，初期状況においてネットワーク中に分散している複数のエージェントが，有限時間内にひとつのノードで集合することを目的とする．ひとつのノードに集合することで，各エージェントが保持している局所情報を共有したり，あるいは，エージェントの動作を同期させたりすることができる．ネットワーク中の各ノードに一意な ID が存在するとき，集合問題は容易に解くことができる．例えば，特定の ID を持つノードで集合するようにアルゴリズムを設計すればよい．しかし，ネットワークによってはセキュリティ上の問題から，ノードの ID を利用できない場合がある．よって，各ノードに ID がない匿名ネットワークにおいて，集合問題を解くことも重要である．しかし，匿名ネットワークにおいて決定性アルゴリズムで集合問題を解くことは不可能である [2]．そのため，ノードに情報を残せる [3, 4, 5]，乱数を利用できる [6, 7, 8]，ネットワークの辺ラベリングを制限する [5, 9]，ネットワークのトポロジを制限する [10] などの仮定を加えることで集合問題を解くアルゴリズムが提案されている．本稿では，匿名単方向リングネットワークにおいて，ノードに情報を残し，乱数を利用することができる場合の集合問題について考察する．

匿名単方向リングネットワークにおける集合問題に対する乱択アルゴリズムとして，文献 [2] のアルゴリズムが挙げられる．このアルゴリズムは，リングネットワーク上の各ノードに，**白板**と呼ばれる読み書き可能なメモリが存在することを仮定している．その白板上に乱数を用いて生成したラベルを書き込むことで，ノードに一意な ID が存在するネットワークを高確率で作成し，その ID を用いて集合を実現している．そのため，各ノードはラベルを保存するために十分な量の白板を管理する必要がある．

本稿では，匿名単方向リングネットワークにおいて，**トークン**を用いて集合問題を解く乱択アルゴリズムを提案する．トークンとは，ノードに置くことができる 1 ビットの目印である．各ノードは 1 ビットのメモリでトークンの情報を管理することができ，白板を用いる場合に比べて，各ノードが管理するメモリの量を削減することができる．本稿では，シミュレーションにより提案アルゴリズムが，ランダムな初期配置から高確率で集合問題を解くことができることを示す．また，決定性アルゴリズムでは集合不可能である周期的な初期配置からも確率的に集合問題を解くことができることを示す．

本稿の構成は以下の通りである．2 章では，本稿で用いるモデルと問題の定義を行う．3 章では，本モデルにおいて，集合問題を解くアルゴリズムを提案し 4 章でその評価を行う．5 章では本稿の結果をまとめる．

2　諸定義

2.1　ネットワークモデル

単方向リングネットワーク R は，2 項組 $R = (V, L)$ で定義される．ここで V はノードの集合であり，L は通信リンクの集合である．ネットワークのノード数を $n = |V|$ と定義する．また，本稿ではリングネットワークを次のように定義する．

- $V = \{v_0, v_1, ..., v_{n-1}\}$
- $L = \{\{v_i, v_{(i+1) \bmod n}\} | 0 \leq i \leq n-1\}$

リング R において，ノード $v_0, v_1, ..., v_{n-1}$ は時計回りにこの順で並んでいるとする．以下，簡単のため，ノードのインデックスにおける計算は n の法のもとに計算されるものとして，$v_{i \bmod n}$ を単に v_i と表す．さらに，v_i と v_{i+1} はリンクで結ばれているため，互いに**隣接ノード**であるという．また，v_i にとって，v_{i+1} は**前方ノード**であるといい，それに対して，v_{i+1} にとって，v_i は**後方ノード**であるという．また，v_i から v_{i+1} への方向を**前方向**，v_{i+1} から v_i への方向を**後方向**であるという．

ネットワークには，ノードに一意な ID が存在する**ラベル付きネットワーク**と，一意な ID が存在しない**匿名ネットワーク**が存在する．本稿では，匿名ネットワークの上で議論を進めていく．また，ネットワーク中の各ノード v_i にはトークンと呼ばれる 1 ビットの目印を置くことができる．以下，v_i 上に存在するトークンを $t_i = \{true, false\}$ で表す．$t_i = true$ のとき，ノード v_i にトークンが存在することを表し，$t_i = false$ のとき，ノード v_i にトークンが存在しないことを表す．

2.2 エージェントモデル

ネットワーク中のエージェントの集合を $A = \{a_0, a_1, ..., a_{k-1}\}$ とする．エージェントは有限ムーア型状態機械 $(S, \delta, s_{initial})$ で定義される．状態集合 S はエージェントの状態の集合であり，エージェントの状態はエージェントが持つ全変数の割り当てにより定義される．状態集合 S の中には，**初期状態** $s_{initial}$ と**終了状態** s_{final} という特別な状態が含まれる．本稿では，全てのエージェントは同じ状態機械であると仮定し，全てのエージェントは同じ初期状態 $s_{initial}$ から実行を開始する．また，エージェントはトークンを各々1 つずつ有しており，ノード v_i にトークンが置かれていなければ 1 つ置くことができる．

状態遷移関数 δ は $\delta : S \times T \times RN \to S \times T \times M$ であり，エージェントの状態，ノード上のトークンの状態，乱数値から，次のエージェントの状態，次のトークンの状態，次の移動を決める関数である．$T = \{true, false\}$ はエージェントが滞在するノード上のトークンの状態を表し，$true$ はトークンが存在すること，$false$ はトークンが存在しないことを表す．RN は乱数の取りうる値の集合を表す．また，$M = \{move, stay\}$ はエージェントの移動を表す．$move$ はエージェントが移動することを表し，$stay$ はエージェントが待機することを表す．このとき，エージェントの移動は瞬間的であるする．つまり，エージェントはリンク上に存在することはなく，常にノードに存在する．さらに，単方向リングネットワークと仮定しているため，v_i に存在するエージェントは v_{i+1} にしか移動することができない．本稿では，各エージェントが同一の状態機械の場合を想定するため，全てのエージェントは同一の状態遷移関数に従って動作する．

2.3 システムの状況

エージェントシステムの**状況** c は，システム内にいる各エージェント a_i の状態 $(s_i \in S)$ と各ノード v_j の状態 $(t_j \in T)$，各エージェントの位置 $(l_i \in L)$ で表す．つまり，各エージェントの全変数の割り当てと各ノードのトークンの状態，エージェントの位置によって状況が決定される．ここで，エージェントの位置は整数の列 $(l_0, l_1, ..., l_{k-1})$ で表す．$L = \{0, 1, ..., n-1\}$ としたとき，$l_i (\in L)$ はエージェント $a_i (\in A)$ がいるノードの位置 (インデックス) を表す．C をエージェントシステムにおいて，取り得る全状況の集合とする．**初期状況** $c_0 (\in C)$ では，各エージェントは同一の状態機械であることを想定しているため，初期状態は同じ $s_{initial}$ であり，各ノードのメモリにはトークンはない状態 $t_i = false$ である．よって初期状況はエージェントの初期位置にのみ依存する．ただし初期状況において，複数のエージェントが同一のノードに存在することはないものとする．

A_i を空ではない任意のエージェントの集合とする．状況 c_i において，A_i に属する全ての実行可能なエージェントが状態遷移関数に従い動作を行うことを $c_i \xrightarrow{A_i} c_{i+1}$ と表記し，**ステップ**と呼ぶ．このとき，A_i に同一のノードにいる複数のエージェントが選択されたとき，それらのエージェントの動作する順序は任意とする．任意の i について，$A_i = A$ のとき，全てのエージェントが同時に動作を行い，このモデルを**同期**モデルと呼ぶ．それに対して，この条件が成り立たない場合，このモデルを**非同期**モデルと呼ぶ．初期状況から i ステップ後の状況を c_i と表す．**実行** E は状況の列であり，$E = c_0, c_1, ...$ と表す．全てのエージェントが終了状態 s_{final} へ遷移した状況を**終了状況** c_{final} と呼ぶ．終了状況以降どのエージェントも動作を行わない．なお，本稿で提案するアルゴリズムは同期モデルを仮定している．

2.4 集合問題

集合問題は，初期状況において任意のノードに分散している複数のエージェントが有限時間内に同一のノードに集合することを目的とする．

休止状態 $(rest)$ をトークンの内容が変わらない限り動作しない終了状態とする．

定義 1 実行 E が以下の条件を満たすとき，E は集合問題を解くという．

- E は有限長である．
- 終了状況において，全てのエージェントが同一のノードに存在し，全てのエージェントの状態が**休止状態**である．

3　アルゴリズム

まず初めに，アルゴリズムの方針を簡潔に述べる．各エージェントは初期状況における初期ノードにそれぞれが有するトークンを置き，同時にアルゴリズムの実行を開始する．エージェントは移動しながら，トークン間の間隔，すなわち，トークンを有するノード間のリンク数を記憶していく．次に，トークン間隔の列が周期的になったとき，すなわち，同じ間隔の列が 2 回繰り返されたときリングネットワーク上を 2 周したものとみなし，リング上の 1 ノードを集合ノードとして選んで集合しようとする．この時，常にエージェントは暫定のノード数 n' を計算している．誤って休止状態となった場合，後方からきたエージェントにより n' を比較した後，n' の大きい方のエージェントと以後同様に振る舞う．

次に, 提案アルゴリズムの変数を Algorithm 1、擬似コードを Algorithm 2 に示す．エージェントはトークンが置かれているノード v_i と，その前方向にはじめて存在するトークンが置かれているノード $v_{i+\Delta_{i_0}}$ との間のリンクの数 Δ_i を記憶する．その後，同様にして Δ のリスト（以下，パターン）$P_i = (\Delta_{i_0}, \Delta_{i_1}, ..., \Delta_{i_j})$ を作成する．作成されたパターンに 2 度同じ繰り返しを発見した時，つまり $\{\Delta_{i_0}, ..., \Delta_{i_x}\} = \{\Delta_{i_{x+1}}, ..., \Delta_{i_{2x+1}}\}$ なった時，パターンを二等分する．そしてそのパターン $P = \{\Delta_{i_0} ..., \Delta_{i_x}\}$ を**辞書式最小順**に並べ替る．リスト $\{x_y, x_{y+1}, ..., x_n, x_0, x_1, ..., x_{y-1}\}$ を辞書式最小順 $\{x_0, x_1, ..., x_n\}$ に並べ替え，その差の合計 $x_y + x_{y+1} + ... + x_n$ から目的ノードまでの距離を算出する．目的ノードへ到着した時，エージェントは休止状態へと移行する．また，同じノードにエージェントがいる場合はエージェント間で通信を行い，n' が大きい方のエージェントと以降同様に振る舞う．ここで暫定ノード数 n' には常にパターン P の要素の合計を 2 で割った値が格納されている．各エージェントは同一のアルゴリズムにしたがって処理を行うため，通信した際にエージェントの変数を全て変更するだけで，それ以降変数を変更されたエージェントは変更したエージェントと同様の動作を行う．

Algorithm 1 変数

Variables in Agent a_i

int $state = 0$ //エージェントの状態変数

ArrayList<Integer> $pattern$ //パターンを記憶するリスト

int $goal = 0$ //集合地点までの距離

int $n' = 0$ // エージェントが推測するネットワークのノード数

int $round = 0$ //同期用変数

int count=0 //リンク数のカウンタ

Variable in Node v_j

boolean $node = false$ //ノード上のトークンの有無を表す

Algorithm 2 決定性アルゴリズム

Main Routine

状態 0：$state == 0$
　$node \leftarrow true$
　$count++$
　$n'++$
　$state \leftarrow 1$

状態 1：$state == 1$
　if $node$ **then**
　　$pattern.add(count)$
　　$count \leftarrow 1$
　　if $pattern$ に繰り返しを発見 **then**
　　　$state \leftarrow 2$
　　　目的ノード$\leftarrow DictionaryMinimum(pattern)$//
　　　目的ノードまでの移動数を算出
　　end if
　end if

状態 2：$state == 2$
　if 目的ノードへ到着 **then**
　　$state \leftarrow 3$
　　次のノードへ
　end if

状態 3：$state == 3$
　休止状態

default：
　$round++$
　if $state$ が 2 のエージェントが同じノードに存在する **then**
　　各変数を共有
　end if

しかしこの決定性アルゴリズムでは，集合が不可能である場合が存在する．それは，リングネットワーク全体に周期性が存在する場合である．リングネットワークにおいて全体として周期性が存在することは，リング R の全てのトークンの間の間隔を記憶したリスト I が 2 回以上の同一のリスト I' の連続で構成されていることを表す．つまり $P = (I' + ... + I')$ と表される．このような場合でも確率的に集合可能にするため，下述の乱択アルゴリズムを上記の決定性アルゴリズムに追加する．

Algorithm 3 に上記の決定性アルゴリズムの状態 3 以降の部分を変更した乱択アルゴリズムを示す．上記の決定性アルゴリズムで，各エージェントが目的のノードまで集合するまでに，最初にエージェントが目的ノードへ到着し休止状態となった時点から高々n' ステップの内に目的ノードへと集合する．そのため，n' ステップの間最初に目的ノードへと到着したエージェントは休止状態のまま待機し，$n'+1$ ステップ目で休止状態を解き，同ノードへと集合しているエージェントを起動する．その後，以下の 2 処理を 50% の確率でどちらか一方のみ行う．

- 暫定のリング R' を 1 周，つまり n' 回ノードを移動し確認を行う．暫定のエージェント数は n' の場合と同様にパターンのリストから推測でき，トークンの数がエージェント数であるため，リスト $pattern$ の大きさとなる．
- その場で休止状態となる

そのため．リング R 全体に周期性が存在し，その周期が $\frac{n}{2}$ であるようなリング R の場合，1 回の確認作業で 50% の確率正しく集合することが可能となる．

Algorithm 3 乱択アルゴリズム

Main Routine
状態 3：$state == 3$
  n' ラウンド待機
  if n' の待機が終了 **then**
    if 50% の確率 **then**
      $state \leftarrow 4$
      $goal \leftarrow n'$
      同一ノード存在する休止状態のエージェント群を起動
      起動したエージェント群に全ての変数を共有
    else
      $state \leftarrow 5$
      同一ノードに存在するエージェント群に共有
    end if
  end if
状態 4：$state == 4$
  if $goal > 0$ **then**
    次のノードへ
  else
    $state \leftarrow 5$
  end if
状態 5：$state == 5$
  休止状態
default：
  $round++$
  n' を推測
  if ($state == 2$ または $state == 4$ であり)，かつ (n' が自分より小さい) エージェントが同じノードに存在 **then**
    各変数を共有
  end if

4 アルゴリズムの評価

4.1 シミュレーション環境

3 章で提案したアルゴリズムのランダムな初期配置におけるエージェント集合確率を評価するためにシミュレーションを行った．シミュレーション環境は表 1 の通りである．

開発環境	:	eclipse Luna 4.4.1
仕様言語	:	java
JDK Version	:	JDK 1.7
試行回数	:	10000 回
ノード数上限	:	500
エージェント数	:	$2 \leq k \leq n$

表 1: シミュレーション環境

4.2　シミュレーション結果と考察

提案するアルゴリズムの集合成功率は表 2 のとおりである．この結果から，乱択アルゴリズムの追加により決定性アルゴリズムに悪影響を与えずに，集合可能な初期配置を増加させることが可能であるといえる．下記のシミュレーション結果において，エージェントの初期配置はランダムとなっている．表 2 に、エージェント数ノード数の両方をランダムに設定し、初期配置もランダムに配置したシミュレーション結果を示す．

アルゴリズム	失敗回数	成功確率
決定性アルゴリズムのみ	99 回	99.01%
乱択アルゴリズム追加後	91 回	99.09%

表 2: シミュレーション結果:$n \leq 500, 2 \leq k < n$ のとき (10000 回)

また，入力としてランダムではなく，全体として周期性の出来やすいノード数 4 エージェント数 2 の場合を 5000 回実行した結果を表 3 1 に掲載する．

アルゴリズム	失敗回数	成功確率
決定性アルゴリズムのみ	1580 回	68.4%
乱択アルゴリズム追加後	795 回	84.1%

表 3: シミュレーション結果:$n = 4, k = 2$ のとき (5000 回)

これらの結果から以下のことがいえる．

- 追加した乱択アルゴリズムは，元の決定性アルゴリズムの動作に悪影響を与えない．決定性アルゴリズムのみのとき，短いリングと勘違いしたエージェントは止まっているだけであるため，正しいリング長を認識したエージェントがいれば必ず追いつく．しかし，乱択部分を追加すると，勘違いしたエージェントが再移動するため，最終的に追いつかれることなく集合に失敗する可能性がある．しかし，このようなことがあまり起こらないことを確認した．
- 全体として周期的な初期位置の場合，確率的に集合確率を向上させることができる．ノード数 4 エージェント数 2 のとき，約 $\frac{1}{3}$ の確率で全体として周期的である配置となる．乱択アルゴリズムを追加して実行した場合，その失敗回数が約半分となっている．そのため，周期的な初期配置の場合でも，確率的に集合問題を解く事ができるといえる．

またこの結果から，乱択部分を複数回実行するように設計すれば，さらに集合確率を上げられることが考えられる．具体的には，$state = 3$ となり n' 回移動して確認処理を行った後，移動した先のノードに暫定エージェント数 k' 以上のエージェントが存在した場合は，さらにアルゴリズムを再実行する．次回以降は，パターンの繰り返しを判定する部分において，前回のパターン長よりも長い場合のみ判定するようにすれば良い．また，移動した先のエージェント数が想定していたエージェント数である状態が複数回続いた場合は，休止状態となれば良い．

5　おわりに

本稿では，匿名単方向リングネットワークにおいて，エージェントがノード数やエージェント数を知らないとき，トークンを用いて集合問題を解く乱択アルゴリズムを提案した．シミュレーションにより，ランダムな初期配置に対して，集合が実現できることを示した．また決定性アルゴリズムでは集合不可能である周期的な初期配置からの集合も確率的に実現できることを示し，アルゴリズムの有効性を確認した．

今後の課題として，乱択アルゴリズムで行っているランダムな移動を繰り返し適用することにより，集合成功確率や，移動回数を理論的に評価するアルゴリズムの提案が挙げられる．また，エージェントをノードに配置する際に周期的な初期配置とならないように配置する乱択アルゴリズムの検討も今後の課題である．

参考文献

[1] E.Kranakis, D.Krizanc, and E.Markou. *The Mobile Agent Rendezvous Problem in the Ring*. Synthesis Lectures on Distributed Computing Theory. Morgan and Claypool Publishers, 2010.

[2] T. Masuzawa, H. Kakugawa, F. Ooshita, and S. Kawai. Randomized gathering of mobile agents in anonymous unidirectional ring networks. Vol. 25, No. 5, pp. 1289–1296, May 2014.

[3] J. Chalopin, S. Das, and P. Widmayer. Rendezvous of mobile agents in directed graphs. In *DISC 2010 – 24th International Conference on Distributed Computing*, Vol. 6343 of *Lecture Notes in Computer Science*, pp. 282–296. Springer, 2010.

[4] S. Das, R. Srámek M. Mihalák, E. Vicari, and P. Widmayer. Rendezvous of mobile agents when tokens fail anytime. Vol. 5401 of *LNCS*, pp. 463–480, Berlin, 2008. Springer.

[5] E. Kranakis, D. Krizanc, and E. Markou. Mobile agent rendezvous in a synchronous torus. In J. Correa, A. Hevia, and M. Kiwi, editors, *Proceedings of the 5th Latin American Symposium on Theoretical Informatics (LATIN 2006)*, Vol. 3887 of *LNCS*, pp. 653–664, Berlin, 2006. Springer.

[6] S. Alpern, V.J. Baston, and S. Essegaier. Rendezvous search on a graph. *Journal of Applied Probability*, Vol. 36, No. 1, pp. 223–231, 1999.

[7] E. Kranakis and D. Krizanc. An algorithmic theory of mobile agents. In R. Bruni and U. Montanari, editors, *2nd Symposium on Trustworthy Global Computing*, No. 4661 in LNCS, pp. 89–97, Berlin, 2007. Springer.

[8] S. Ghosh. Distributed systems: an algorithmic approach. 2007.

[9] L. Barrière, P. Flocchini, P. Fraigniaud, and N. Santoro. Rendezvous and election of mobile agents: Impact of sense of direction. *Theory Comput. Syst.*, Vol. 40, No. 2, pp. 143–162, 2007.

[10] D. Baba, T. Izumi, F. Ooshita, H. Kakugawa, and T. Masuzawa. Space-optimal rendezvous of mobile agents in asynchronous trees. *Structural Information and Communication Complexity*, pp. 86–100, 2010.