

キラリティを持たない モバイルエージェントによる 動的リング上での g -部分集合問題

九州工業大学 大手 陸史

指導教員：柴田 将拡

概要

本論文では、同期的動的リングネットワークにおいてモバイルエージェント間にキラリティが無い場合の g -部分集合問題を考察する。モバイルエージェントの協調動作に関する問題の一つ、全集合問題は、 k 体のエージェント全てが一つのノードに集合することを要求する。そして、部分集合問題はその全集合問題を一般化したものであり、与えられた自然数 $g(< k)$ に対して、各ノードに少なくとも g 体のエージェントが存在するか、エージェントが存在しないかの状態でエージェントが終了することを要求する。また、本論文では部分集合を行うネットワークに、1-インターバル連結リングつまり、 n 個のノードがリンクで連結され、各時間ステップ(ラウンド)毎にリンクの高々一つが欠落する可能性のあるリングを考える。先行研究ではこの g -部分集合問題について、1-インターバル連結リング上でエージェント間にキラリティ(エージェント同士の進行方向の同意)がある場合のアルゴリズムが提案されている。これに対して、エージェント間にキラリティが無い場合のアルゴリズムを k と g の関係ごとに設計を行い、問題の解決に必要な総ラウンド数と、エージェントの総移動数について比較を行った。第一に、 $2g + 1 \leq k \leq 3g - 2$ の時、ラウンド数 $O(n \log g)$ と総移動数 $O(gn \log g)$ で解けることを示す。第二に、 $3g - 1 \leq k \leq 8g - 4$ の時、ラウンド数 $O(n)$ と総移動数 $O(kn)$ で解けることを示す。最後に、 $k \geq 8g - 3$ の時、ラウンド数 $O(kn/g)$ と総移動数 $O(gn)$ で解けることを示す。これらの結果は、 $k \geq 2g + 1$ のとき動的リング上でエージェント間にキラリティが無くとも g -部分集合問題を解けることを意味する。加えて、 $k < 8g - 3$ の時はキラリティ有の時と同等の性能で、 $k \geq 8g - 3$ の時はラウンド数は増加するが総移動数は同等の性能で g -部分集合問題を解くことができている。

1 はじめに

1.1 研究背景・関連研究

分散システムとは、通信リンクで接続された計算機(ノード)の集合として構成される。分散システムの有望な設計手法として、モバイルエージェントが注目されている [1][2]。エージェントは、訪問したノードで収集した情報を持ちながらシステム上を行き来し、各ノードでアクションを実行しタスクを達成する。言い換えると、エージェントはプロセスコードとデータをカプセル化することができ、分散システムの設計を簡素化することができる [3][4]。エージェントの協調動作に関する基本的な問題に、全集合問題がある。この問題は、 k 体のエージェントが任意のノードに配置されたとき、 k 体全てのエージェントが一つのノードに集合して終了することを要求する。一つのノードに集合することでエージェントは情報の共有や、行動の同期が可能である [5][6][7]。

そして、 g -部分集合問題 [8] と呼ばれる、全集合問題を変形したものが検討されている。この問題では、全てのエージェントが一つのノードに集合するのではなく、いくつかのノードに別々に集合する。具体的には、与えられた自然数 $g(< k)$ に対して、各ノードに少なくとも g 体以上のエージェントが存在するか、1 体も存在しないかのいずれかの状態で終了することを要求する。この問題は、実用的な観点からは特に大規模なネットワークで有効であると考えられる。つまり、 g -部分集合が達成されると、エージェントは少なくとも g 体以上ごとのグループに分割され、同じグループのエージェントと情報やタスクを共有することができる。また、各グループはネットワークを分割し、割り当てられた領域を効率的にパトロールすることも期待できる。 $k < 2g$ が成立する場合、 g -部分集合問題は、全集合問題と等価である。一方、 $k \geq 2g$ が成立する場合、 g -部分集合問題に対する要求は全集合問題に対するものよりも易しくなる。したがって、 g -部分集合問題は、全集合問題よりも少ない移動回数で(すなわち、より少ないコスト)で解ける可能性がある。

関連研究として、柴田らはリングネットワーク [8][9][10][11] における g -部分集合問題を検討している。特に、[11] では、システムの実行中にトポロジーが変化する、1-インターバル連結リングにおける g -部分集合問題を解くアルゴリズムを提案している。その研究においてはエージェント間にキラリティを有する、すなわち進行方向についての同意があると仮定されているが、本研究では、エージェント間にキラリティが無いと仮定した場合のアルゴリズムの設計と評価を行った。

1.2 本論文の成果

本論文では1-インターバル連結リング [11][12][13][14][15] における、モバイルエージェント間にキラリティが無い場合の g -部分集合問題について考察し、アルゴリズム

ムを提案する．図 1.1 に例を示す．ネットワークはエージェントが双方向に移動できるリングであり，各ノードは情報を書き込み読み出しできるホワイトボードを持つとする．そして，各エージェントは個別の ID，ノード数 n とエージェント数 k の知識を持ち，完全に同期的に行動する．このような設定において，[11] にて n と k の関係ごとに g -部分集合を達成するアルゴリズムが提案されている．この設定に，エージェント間にキラリティが無いという条件を加えアルゴリズムを設計し，[11] にて提案されているキラリティの有るアルゴリズムと，時間複雑度 (ラウンド数) とエージェントの総移動数を比較した．エージェント間にキラリティが無い場合のアルゴリズムの結果と，既存の結果 [11] を表 1 にまとめた．まず， $k \leq 2g$ の時， g -部分集合問題は解けないことが [11] にて示されている． $2g + 1 \leq k \leq 3g - 2$ の時，ラウンド数 $O(n \log g)$ と総移動数 $O(gn \log g)$ で解けることを示す．次に $3g - 1 \leq k \leq 8g - 4$ の時，ラウンド数 $O(n)$ と総移動数 $O(kn)$ で解けることを示す．そして， $k \geq 8g - 3$ の時，ラウンド数 $O(kn/g)$ と総移動数 $O(gn)$ で解けることを示す．これらの結果は， $k \geq 2g + 1$ の時 g -部分集合問題が動的リング上でエージェント間にキラリティが無くとも解けることを意味している．加えて， $k < 8g - 3$ の時はキラリティ有の時と同等の性能で， $k \geq 8g - 3$ の時はラウンド数は増加するが総移動数は同等の性能で g -部分集合問題を解くことができる．

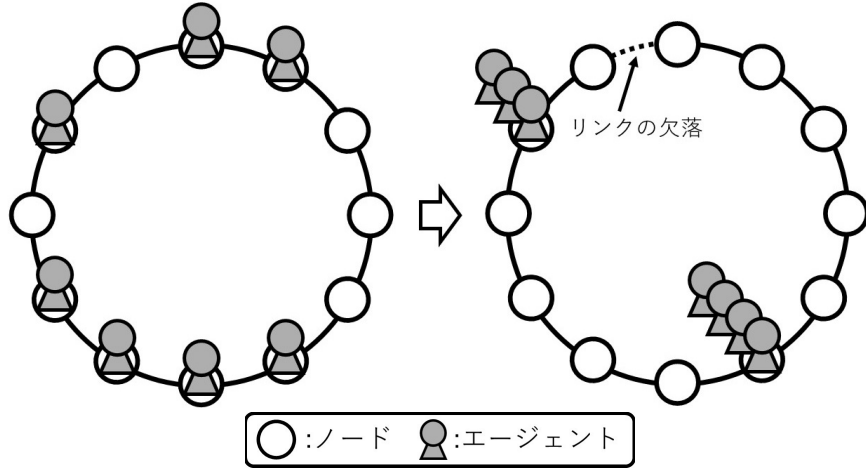


図 1 動的リングにおける， g -部分集合問題 ($g = 3$) の例

2 予備知識

2.1 システムモデル

本論文で扱うシステムモデルは，基本的に [11] で定義されたものに従う．動的双方向リング R は 2-tuple $R = (V, E)$ と定義され， $V = \{v_0, v_1, \dots, v_{n-1}\}$ を n 個のノード

表 1 キラリティの有無による g -部分集合の結果

	本研究	先行研究 [11]	本研究	先行研究 [11]
キラリティ	無し	有り	無し	有り
k と g の関係	$2g + 1 \leq k \leq 3g - 2$		$3g - 1 \leq k \leq 8g - 4$	
総ラウンド数	$O(n \log g)$	$O(n \log g)$	$O(n)$	$O(n)$
総移動数	$O(gn \log g)$	$O(gn \log g)$	$O(kn)$	$O(kn)$
	本研究	先行研究 [11]		
キラリティ	無し	有り		
k と g の関係	$k \geq 8g - 3$			
総ラウンド数	$O(kn/g)$	$O(n)$		
総移動数	$O(gn)$	$O(gn)$		

の集合, $E = \{e_0, e_1, \dots, e_{n-1}\} (e_i = \{v_i, v_{(i+1) \bmod n}\})$ をリンクの集合とする. 簡単にするため, $v_{(i+j) \bmod n}$ (resp., $e_{(i+j) \bmod n}$) を v_{i+j} (resp., e_{i+j}) と表記する. そして, v_i から v_{i+1} (resp., v_i から v_{i-1}) への方向を時計回り (resp., 反時計回り) と定義する. リング内のリンクの一つが各時間ステップで欠落することがあり, 欠落するリンクは, 敵対的スケジューラによって選択される. このような動的リングは 1 インターバル連結リングと呼ばれる. ノード v_i からノード v_j への距離は, $\min\{(j-i) \bmod n, (i-j) \bmod n\}$ と定義される. ノードは匿名で ID を持たないと仮定する. 加えて, 各ノード $v_i \in V$ は v_i に滞在するエージェントが読み書きできるホワイトボードを持つ.

$A = \{a_0, a_1, \dots, a_{k-1}\}$ を $k (\leq n)$ 体のエージェントの集合とする. エージェントはリンクを介してノード間を移動することができる. つまり任意の i について, v_i から v_{i+1} へ, または v_{i-1} へ移動することができる. エージェントは固有の ID を持ち, n と k についての前提知識を持っていると仮定する. また, エージェントは現在の滞在するノードに他のエージェントが滞在しているかを直接検知することはできない. そして, エージェント間にキラリティを持たない. つまり, 各エージェントが前進方向と認識する移動方向が時計回りか反時計回りのどちらかであり, エージェント間で統一されていない. エージェント a_i は決定性有限オートマトン $(S, W, \delta, s_{\text{initial}}, s_{\text{final}}, w_{\text{initial}}, w'_{\text{initial}})$ として定義される. 最初の要素 S はエージェントの全ての状態の集合であり, 二つの特別な状態, 初期状態 s_{initial} と最終状態 s_{final} を含む. 二番目の要素 W はホワイトボードの全ての状態 (内容) の集合であり, 二つの特別な状態 w_{initial} と w'_{initial} を含む. w_{initial} と w'_{initial} については次の段落で説明する. 三番目の要素 $\delta: S \times W \mapsto S \times W \times M$ は a_i とホワイトボードの現在の状態から, a_i とホワイトボードの次の状態と, a_i が隣のノードに移動するか否かを決定する状態遷移関数である. δ の最後の要素である $M = \{-1, 0, 1\}$ は a_i が移動を行うかどうかを表す. 値 1 (resp., -1) は前進 (resp., 後進), 値 0 はその場に留まることを意味する. そして, 任意の $w_i \in W$ にて $\delta(s_{\text{final}}, w_j) = (s_{\text{final}}, w_j, 0)$ が成り立つ時,

つまり a_i が状態 s_{final} に達した時、以降決して状態を変更したり、ホワイトボードを更新したり、現在のノードを離れたりしなくなる。 $S, \delta, s_{initial}, s_{final}$ はエージェントの ID と前進方向 (時計回りか反時計回りか) に依存する。

エージェントシステムでは、(大域) 状況 c は全エージェントの状態、全ノードの状態 (ホワイトボードの内容)、全エージェントの位置 (どのノードに滞在しているか) の積として定義される。そして C を全ての状況の集合と定義する。初期状況 $c_0 \in C$ において、エージェントはそれぞれ異なるノードに任意に配置され (同じノードに複数のエージェントは配置されない)、各ホワイトボードの状態はエージェントがいるか否かに応じて、 $w_{initial}$ または $w'_{initial}$ であるとする。つまり、初期状況においてノード v_j にエージェントが配置されている場合 $w_{initial}$ 、配置されていない場合 $w'_{initial}$ となる。

アルゴリズムの実行中、エージェントのノード間の移動は一瞬で行われるものとし、エージェントは常にノードに存在する (リンク上にエージェントは存在しない)。各エージェントは、アトミックアクションとして以下の4つの操作を実行する: 1) 現在のノードのホワイトボードの内容を読む、2) ローカル計算を実行する (または状態を変更する)、3) 現在のノードのホワイトボードの内容を更新する、4) 隣のノードに移動するか現在のノードに留まる。同じノードに複数のエージェントが存在する場合、それらのエージェントは任意の順番でアトミックアクションをインターリーブする。また、あるエージェントが隣のノードに移動しようとしたとき (例えばノード v_j から v_{j+1}) 対応するリンクが (リンク e_j) が欠落している場合、そのエージェントはブロックされたとし、次のアトミックアクション開始時にはまだ v_j に留まったままになる。

本システムモデルは完全同期リングを仮定する。つまり、ラウンドと呼ばれる各時間ステップにおいて、全てのエージェントがアトミックアクションを一度実行する。そして、 c_0 から始まる実行は $E = c_0, c_1, \dots$ と定義される。ここで、各 $c_i (i \leq 1)$ は全てのエージェントのアトミックアクションによって c_{i+1} から到達した状況である。実行は、無限であるか、または全てのエージェントの状態が s_{final} の状況で終了する。

2.2 g -部分集合問題

本論文での g -部分集合問題の要件は、与えられた自然数 g に対して各ノードに、少なくとも g 個以上のエージェントが存在するか、エージェントが存在しないかのどちらかの状態で終了することとする。

定義 1. 以下の条件が成立するとき、アルゴリズムは動的リングにおける g -部分集合問題を解く

- 実行 E は有限である。 (すべてのエージェントは s_{final} で終了する)

- 最終的に、エージェントが存在するノードには、少なくとも g 個以上のエージェントが存在する。

本論文では、提案アルゴリズムを時間複雑度 (問題を解くのに必要なラウンド数) と全エージェントの総移動回数で評価する。[11] では静的リングにおけるエージェントの総移動回数の下界が $\Omega(gn)$ であることが示されている。この定理は動的リングにおいても成り立つ。

定理 1. 動的リングにおける g -部分集合を解くのに必要なエージェントの総移動回数の下界は $g \geq 2$ の時 $\Omega(gn)$ である。

また、時間複雑度については [11] にて以下の定理が成り立つことが示されている。

定理 2. 動的リングにおける g -部分集合問題を解くのに必要な時間複雑度の下界は $\Omega(n)$ である。

3 $2g + 1 \leq k \leq 3g - 2$ の場合

本章では、 $2g + 1 \leq k \leq 3g - 2$ の時、動的リングにおける g -部分集合問題をラウンド数 $O(n \log g)$ 、総移動回数 $O(gn \log g)$ で解くアルゴリズムを提案する。このアルゴリズムでは、全てのエージェントはリングを一周して全エージェントの ID を取得し、集合する一つの共通のノードを決定しようとする。しかし、リンクの欠落によって ID が取得できない可能性がある。これを解決するためにエージェントは後述する追加の動作を行い、最終的に g -部分集合を達成する。アルゴリズムは選択フェーズ、集合フェーズの二つのフェーズによって構成される。選択フェーズでは、エージェントはリング内を移動し集合するノードを決定する。そして集合フェーズでは、エージェントは決定したノードを目指し移動する。

3.1 選択フェーズ

このフェーズの目的は、各エージェントがリングを周回し ID を収集し、集合ノードを決定することである。まず、各エージェント a_i は現在のノードのホワイトボードに自身の ID を書きこみ、 $3n$ ラウンドの間前進しようとする。この時エージェント間にキラリティが無いとため、各エージェントの前進方向は v_j から v_{j+1} か v_j から v_{j-1} の二通り、つまり時計回りと反時計回りのいずれかである。 $3n$ ラウンドの移動の間、 a_i は訪問したノードのホワイトボードを調べ ID を配列 $a_i.ids[]$ に記録し、訪問したノードの数を記録する変数 $a_i.nVisited$ を 1 増加させる。 $3n$ ラウンド後、 $a_i.nVisited$ の値は、(a) 少なくとも n 以上、または (b) リンクの欠落によって n 未満である。(a) の場合 a_i はリングを一周したことになり、したがって全エージェント k 個の ID を

取得している．この時，全てのエージェントは記録した中から最小の値の ID が書かれたノードを集合ノード v_{gather} に決定する．(b) の場合 [11] にて，エージェント間にキラリティを有するアルゴリズムでは， k 個のエージェント全てが同じノードに留まることが示されている．従って，このアルゴリズムではエージェントのうち，前進方向が同じ，つまり時計回りか反時計回りかが一致するエージェントは，全て同じノードに留まっている．この時， a_i は収集した ID が k 未満であることを現在のノードに記録し，そのノードを準集合ノードとして $v_{semi-gather}$ に決定する．

選択フェーズの疑似コードを Algorithm 1 に示す．グローバル変数は表 2 に示す(いくつかの変数は他の章で使用する)．

選択フェーズにおいて，エージェントが $3n$ ラウンドの間前進した後，訪問したノードの数が n 未満の場合，現在のノードにそのことを書き込む (14 行目)．そして，現在のノードに訪問したノードの数が n 未満のエージェントがいるならば，進行方向が同じエージェントは同じノードに集まっているので，現在のノードを $v_{semi-gather}$ に決定する．自身のノード訪問数が n 以上のエージェントは，現在のノード上に，訪問したノード数が n 未満であるようなエージェントが存在しない場合は，記録した ID の最初の k 個の内，最小の ID を持つノードを v_{gather} に決定する (18 行目)．

Algorithm 1 選択フェーズにおけるエージェント a_i の振る舞い (v_j は a_i が現在滞在しているノード)

```

1:  $v_j.id \leftarrow a_i.id$ ,  $a_i.ids[a_i.nIDs] \leftarrow v_j.id$ 
2:  $a_i.nIDs \leftarrow a_i.nIDs + 1$ ,  $v_j.nAgents \leftarrow v_j.nAgents + 1$ 
3: while  $a_i.rounds \leq 3n$  do
4:    $v_j.nAgents \leftarrow v_j.nAgents - 1$ 
5:   ノード  $v_j$  から前進方向のノードへ移動を試みる
6:   if  $a_i$  が前進方向の新しいノード  $v_j$  へ移動した then
7:      $a_i.nVisited \leftarrow a_i.nVisited + 1$ 
8:     if  $(v_j.id \neq \perp) \wedge (a_i.nIDs < k)$  then
9:        $a_i.ids[a_i.nIDs] \leftarrow v_j.id$ ,  $a_i.nIDs \leftarrow a_i.nIDs + 1$ 
10:    end if
11:  end if
12:   $v_j.nAgents \leftarrow v_j.nAgents + 1$ ,  $a_i.rounds \leftarrow a_i.rounds + 1$ 
13: end while
14: if  $a_i.nVisited < n$  then
15:    $v_j.missing \leftarrow true$ 
16: end if
17: if  $v_j.missing = false$  then
18:    $a_i.gatID \leftarrow$  収集した ID  $a_i.ids[]$  のなかで最小の ID
19: end if

```

表 2 提案アルゴリズムに使用するグローバル変数

エージェント a_i の変数

型	変数名	意味	初期値
int	$a_i.rounds$	現在のラウンド数	1
int	$a_i.nIDs$	観測した異なる ID の数	0
int	$a_i.nVisited$	訪問したノードの数	0
int	$a_i.rank$	同じノード内で ID が何番目に小さいか	0
int	$a_i.dir$	移動しようとする方向 (1: 前進, -1: 後進)	0
int	$a_i.gatID$	集合ノードの ID	null
array	$a_i.ids[]$	観測した ID の配列	\perp
boolean	$a_i.crs$	他のエージェントと交差したか	false
int	$a_i.minID$	交差した, あるいは同じノードに集まった エージェントの中で最小の ID	null

ノード v_j の変数

型	変数名	意味	初期値
int	$v_j.id$	ノードが保存している ID	\perp
int	$v_j.ids[]$	ノードが保存している ID の配列	\perp
int	$v_j.nAgents$	ノードに滞在しているエージェント数	0
boolean	$v_j.Missing$	ID 収集数が n 未満のエージェントがいるかどうか	false
int	$v_j.dir$	初めて v_j を訪れたエージェントグループが移動しようとする方向 (1: 前方, -1: 後方)	0
boolean	$v_j.waiting$	v_j に留まるエージェントがいるかどうか	false
boolean	$v_j.fMarked$	v_j が前進グループに訪問されたかどうか	false
boolean	$v_j.bMarked$	v_j が後進グループに訪問されたかどうか	false
boolean	$v_j.candi$	v_j が集合ノードかどうか	false
int	$v_j.rID$	v_{j+1} への移動を表す変数	0
int	$v_j.lID$	v_{j-1} への移動を表す変数	0
boolean	$v_j.uni$	前進方向が統一されているかどうか	false

選択フェーズにて、以下の補題 1 がある。

補題 1. アルゴリズム 1 が終了した時、各エージェントは次のいずれかを達成する：
(i) リングを一周してエージェントの ID を収集し、集合ノード v_{gather} を決定する。
(ii) リングを一周できなかった同じ前進方向の全エージェントは一つのノードに集まり、 $v_{semi-gather}$ を決定する。

Proof. アルゴリズム 1 を実行した時の $a_i.nVisited$ の値が、(a) n 以上 (20 行目)、(b) n 未満 (14 行目) の順に考える。まず、 $a_i.nVisited \geq n$ が成立する場合、 a_i は少なくともリングを一周し、 k 体全てのエージェントの ID を収集している。従って全てのエージェントの中で最小の ID を保存するノードを v_{gather} を決定し (i) を達成する。次に (b) $a_i.nVisited < n$ が成立する場合を考える。この場合、リングを一周できなかったエージェントと同じ前進方向のエージェントは一つのノードに集まることを示す。アルゴリズム 1 を実行した後、前進方向が同じエージェント a_i と a_j が異なるノードに存在すると仮定し、 a_i から a_j の距離 d_{ji} を考える。あるラウンドで a_i がブロックされ、 a_j が前進すると d_{ji} は 1 減少し、その逆も同様である。 a_i があるラウンドでブロックされなかった場合、すべてのエージェントが前進しようとするので、 d_{ji} の値は最大でも 1 だけ増加する。そして、 $a_i.nVisited < n$ であるため、 a_i がブロックされ d_{ji} は 1 減少する (あるいはすでに 0 である) ことは少なくとも $2n + 1$ ラウンド起こる。これは、 a_i と a_j が同じノードに留まることを示し、仮定と矛盾する。よって、 $a_i.nVisited < n$ となりリングを一周できなかった同じ前進方向の全エージェントは一つのノードに集まり、そのノードを $v_{semi-gather}$ に決定し (ii) を達成する。全てのエージェントについて、(i) か (ii) のいずれかを満たすため、補題が成立する。 \square

3.2 集合フェーズ

このフェーズの目的は、エージェントが集合ノード v_{gather} を訪問し、 g -部分集合 (または全集合) 達成することである。このフェーズの開始から $3n$ ラウンドの間、選択フェーズにて v_{gather} を決定したエージェントは、 v_{gather} に到達するまで前進しようとし、到達後そのノードに留まる。 $v_{semi-gather}$ を決定したエージェントはそのノードに留まり続ける。しかし、 v_{gather} に到達するまで前進を試みたが、リンクの欠落によって到達できなかったエージェントが存在する可能性もある。よって、エージェントの配置によって、次のように動作を分岐する。 $3n$ ラウンド前進した後、(a) v_{gather} に留まっているエージェントは、現在のノードが集合ノードであることをノードに書き込み、 $2n$ ラウンドの間留まる。(b) $v_{semi-gather}$ に留まっているエージェントは、一つのグループとして n ラウンドの間前進し、その後 n ラウンドの間後進する。その間に集合ノードであると記録されているノード (すなわち v_{gather}) に到達すると留まる。(c) v_{gather} に到達できなかったエージェントは、補題 1 の議論と同様に一か所

のノードに集合しているため、一つのグループとして n ラウンドの間前進し、その後 n ラウンドの間後進する．その間に v_{gather} に到達するとそのノードに留まる．この時点で全てのエージェントが v_{gather} に留まっていれば全集合を達成しているため、アルゴリズムを終了する．まだ v_{gather} に到達していないエージェントが存在する場合や、選択フェーズにて二つの $v_{semi-gather}$ が決定され v_{gather} が存在しなかった場合は、全てのエージェントは合わせて二つのノードに留まっている．この時点でどちらのノードにも g 体以上のエージェントが存在すれば、 g -部分集合を達成しているため、アルゴリズムを終了する．しかし、どちらかのノードに存在するエージェントが g 体未満の場合、もう片方のノードには少なくとも $2+g$ 体のエージェントが存在し、そのノードからスプリッティング [11] と呼ばれる手法を用いて g -部分集合を達成する．直感的にはこの手法は、あるノードに $g+c(c \geq 2)$ 体のエージェントが存在するとき、そこから $c/2$ 体のエージェントを前進、さらに $c/2$ 体のエージェントを後進させて g 体以下のエージェントが存在するノードに移動し、残りの g 体のエージェントは現在のノードに留まる．この行動と、後述する追加の行動を繰り返すことで、エージェントが g 体以下のノードのエージェントを補充し最終的に g -部分集合を達成する．

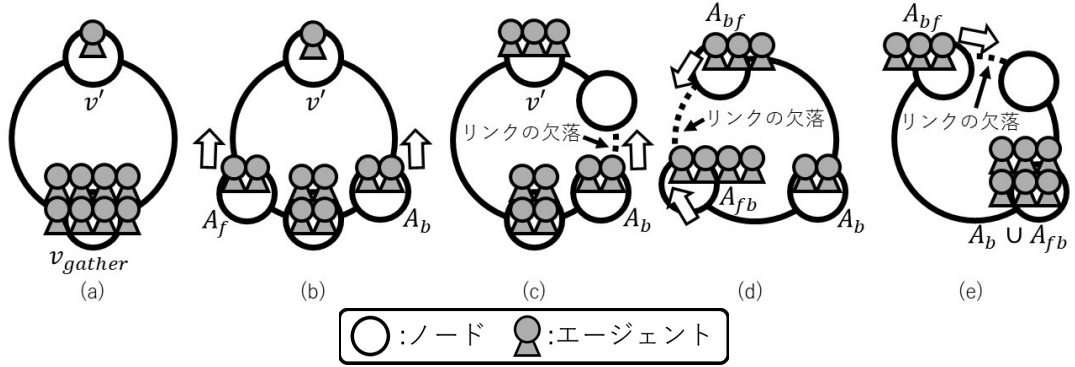


図 2 $2g+1 \leq k \leq 3g-2$ の場合の集合フェーズの実行例

具体的には、エージェントの $3n$ ラウンドとその後の $2n$ ラウンドの移動後、 v_{gather} に到達しなかったエージェントは一つのノードに集まっているため、そのノードを v' とする．よって、エージェントは v_{gather} か v' のどちらかのノードに存在する．選択フェーズにて二つの $v_{semi-gather}$ が決定され v_{gather} が存在しない場合もエージェントの動作は同様なので、以降の説明やでは二つのエージェントグループが存在するノードをそれぞれ v_{gather} と v' に置き換える． $g=4$ の場合の例を (図 2(a)) に示す (説明に不要なノードは省略している)．エージェントは k と g の知識を持つため、 v_{gather} と v' の両方に少なくとも g 体のエージェントが存在すれば、 g -部分集合を達成しているため、アルゴリズムを終了する．そうでない場合、つまり、 v_{gather} か v' に g 体未満のエージェントしかない場合、スプリッティングを用いて g 体未満のノードを

訪問しようとする。この手法は、最大で $\lceil \log g \rceil$ のサブフェーズからなる。スプリッティングの最初に v' に $g - c'(1 \leq c' \leq g - 1)$ 体のエージェントが存在すると仮定する。そして、 $c = k - 2g + c'$ とすると、 v_{gather} には、 $k - (g - c') = g + c$ のエージェントが存在する。ここで、 $c > c'$ と $c \geq 2$ が成立することに注意。まず、 v_{gather} に存在するエージェントの間で進行方向の同意をとる。つまり、キラリティを獲得する。次に、 v_{gather} の各エージェントは、 $g + c$ 体のエージェントの中で、自分の ID が何番目に小さいかを記録する。その序数を $a_i.rank$ とする。そして、 $1 \leq a_i.rank \leq g$ であれば、 a_i は v_{gather} に留まる。 $g + 1 \leq a_i.rank \leq \lfloor c/2 \rfloor$ の場合、エージェントは前進グループ A_f 、 $\lfloor c/2 \rfloor + 1 \leq a_i.rank \leq g + c$ の場合、後進グループ A_b に属する。その後、前進グループ A_f (resp., 後進グループ A_b) は V' を訪れるまで前進 (resp., 後進) を試みる (図 2(b))。このとき、それぞれのグループの前進 (後進) 方向は、各エージェントのそれまでの前進方向に影響されず、 v_{gather} に滞在していたエージェント間では統一されている。そして、各ラウンドで欠落しているリンクは最大一つなので、少なくとも A_f と A_b のどちらかは次のノードに移動でき、 v' に到達することができる (図 2(c))。

A_f は v' に到達できたが、 A_b はブロックされて v' に到達できなかったと仮定する。その後、 v' にいる全てのエージェントがリングを移動して、 A_b が滞在しているノードを訪問しようとする。具体的には、前の n ラウンドで、 v_{gather} に滞在した全てのエージェントからなるグループを A_{fb} とし、 A_f に含まれるエージェントと前の n ラウンドで v' に滞在したエージェントからなるグループを A_{bf} とする。そして、 A_{fb} (resp., A_{bf}) は n ラウンドの間 A_b のいるノードに到達するまで前進 (resp., 後進) を試みる (図 2(d))。その後、 A_{fb} (resp., A_{bf}) が A_b がいるノードに到達できなかった場合、 A_{fb} (resp., A_{bf}) は方向を切り替え、 n ラウンドの間 A_b のいるノードに到達するまで後進 (resp., 前進) を試みる。 A_b のいるノードで留まらなないと仮定した時、この動作によって全てのノードが A_{fb} か A_{bf} に訪問されることになる。したがって、 A_{fb} または A_{bf} は A_b のいるノードを訪問することができる。 (図 2(e))。もし、 A_{bf} がノードに到達できれば、 A_{fb} がいるノードには、 g 体のエージェントが存在し、 A_{bf} と A_b を持つノードには、 $g - c' + c \ (\geq g)$ 体のエージェントが存在する。従って、 g -部分集合を達成し、アルゴリズムを終了する。そうでない場合、つまり A_{fb} だけが A_b のいるノードに到達した場合、 A_{fb} と A_b のノードには、 $g + \lfloor c/2 \rfloor$ 体のエージェントが存在し、 A_{bf} のノードには、 $g - c' + \lceil c/2 \rceil$ 体のエージェントが存在する。そうすると、エージェント数が g 未満のノードにおけるエージェント数と g との差は半分になる。したがって、このようなサブフェーズを最大 $\lceil \log g \rceil$ 回実行することで、エージェントが存在する全てのノードに、少なくとも g 体のエージェントが存在するような構成を作ることができる (g -部分集合が達成される)。

集合フェーズの疑似コードをアルゴリズム 2 に示す。このフェーズの最初の $3n$ ラウンドと $2n$ ラウンドを実行した後、各エージェントは $v_j.nAgents$ によって現在のノードのエージェント数をカウントする。 $v_j.nAgents \geq g + 2$ で、エージェントが

存在する他のノードのエージェント数が g 未満の場合、 a_i は手順 $More()$ を実行し、 $a_i.rank$ に応じて、 A_f か A_b に属するか、 v_j に留まり続けるかを決定する。 $More()$ の疑似コードをアルゴリズム 3 に示す。もし、 $v_j.nAgents < g$ であれば、 a_i は手順 $Less()$ を実行し、 v_j に n ラウンド滞在し続け、その後移動を再開する。 $Less()$ の疑似コードをアルゴリズム 4 に示す。アルゴリズム 3 と 4 では、簡単のために、エージェント間のキラリティの獲得と、 a_i が $a_i.rank$ をどのように計算するかは省略し、エージェントはリング内を移動するために、手順 $Moving()$ と $LatterMove()$ を使用する。それぞれの疑似コードはアルゴリズム 5 と 6 に示す。

集合フェーズにて、以下の補題 2 がある。

補題 2. アルゴリズム 2 が終了した時、エージェントは g -部分集合を達成している

Proof. まず、集合フェーズの開始から、 $3n$ ラウンドの間、各エージェントは v_{gather} に到達するために前進しようとする。そして、補題 1 と同様の議論により、 v_{gather} に到達しなかった前進方向が同じエージェントは一つのノードに集まる。その後、 v_{gather} に到達しなかったエージェントのグループが v_{gather} に到達するまで n ラウンドの間前進、 n ラウンドの間後進することで、全てのエージェントは v_{gather} か v' に留まっている。選択フェーズにて二つの $v_{semi-gather}$ が決定され v_{gather} が存在しない場合もエージェントの動作は同様なので、二つのエージェントグループが存在するノードをそれぞれ v_{gather} と v' とみなす。少なくとも g 体のエージェントが v_{gather} と v' に存在するとき、すでに g -部分集合を達成しているのでアルゴリズムを終了する。一方、 v' (resp., v_{gather}) に g 体未満のエージェントしか存在しないとき、 $2g+1 \leq k \leq 3g-2$ の場合を考えるので、 v_{gather} (resp., v') には少なくとも $g+2$ 体のエージェントが存在する。 v' に $g-c'(1 \leq c' \leq g-1)$ 体のエージェントが存在すると仮定する。そして、 $c = k - 2g + c'$ とすると、 v_{gather} には、 $k - (g - c') = g + c$ 体のエージェントが存在する。この時 $c > c'$ 、 $c \geq 2$ が成立することに注意。このような状態から、 v_{gather} のエージェントは $More()$ を実行し、次に $Moving()$ を実行する。手順 $More()$ では、 $\lfloor c/2 \rfloor$ 体のエージェントは前進グループ A_f に属し前進しようとし、 $\lceil c/2 \rceil$ 体のエージェントは後進グループ A_b に属し後進しようとする。(5 から 7 行目)。リンクの欠落は最大一つのため、 A_f または A_b は、既に v' を訪問していない限り、次のノードに移動することができ、従ってどちらかが n ラウンド以内に v' を訪問することができる。一般性を損なわず、 A_f は v' に到達できたが、 A_b はリンクの欠落によって到達できないと仮定する。その後、 A_b が存在するノードを訪問するために、手順 $LatterMove()$ により、 v_{gather} (resp., v') に滞在したエージェントはエージェントグループ A_{fb} (resp., A_{bf}) に属し、 n ラウンドの間前進 (resp., 後進) しようとする。 A_b が存在しないと仮定した場合、この $2n$ ラウンドの間で A_{fb} と A_{bf} で全てのノードを訪問することができる。 A_{bf} がそのノードに到達した時、エージェントのいる両ノードには少なくとも g 体のエージェントが存在し、そこでアルゴリズムを終了することができる。 A_{fb} がノードに到達すると、 A_{fb} が存在するノードには $g + \lceil c/2 \rceil$

体のエージェントが存在し、 A_{bf} が存在するノードには、 $g - c' + \lfloor c/2 \rfloor$ 体が存在する (図 2(e)). $c > c'$ が成り立つので、エージェントが g 体未満のノードでは、上記の振る舞いによってエージェント数と g とのさが半分になる. 従って、このような振る舞いを最大 $\lceil \log g \rceil$ 回繰り返すことで、エージェントは g -部分集合を達成することができる. よって補題が成立する. \square

提案アルゴリズムについて、以下の定理 3 を示す.

定理 3. $2g + 1 \leq k \leq 3g - 2$ が成立するとき、提案アルゴリズムは動的リングにおける g -部分集合問題を $O(n \log g)$ ラウンド、総移動数 $O(gn \log g)$ で解く.

Proof. 補題 1 と補題 2 により、提案アルゴリズムは g -部分集合問題を解くことができる. 以下では、問題を解くまでの総ラウンド数とエージェントの総移動数を解析する.

まず、選択フェーズでは、各エージェントは集合ノード v_{gather} を決定するために $3n$ ラウンド前進しようとする. ここでは $k = O(g)$ が成り立つので、 $O(n)$ ラウンド、総移動数 $O(gn)$ が必要となる. 次に、集合フェーズでは、始めに $3n$ ラウンドとその後の $2n$ ラウンドの間、各エージェントは v_{gather} に到達するために前進または後進しようとするが、これには $O(n)$ ラウンドと総移動数 $O(gn)$ が必要となる. その後、(i) 手順 *More()* により n ラウンドの間、少なくとも $g + 2$ 体のエージェントを持つノードから、複数のエージェントが前進または後進して、 g 体未満のエージェントをもつノードを訪問する. そして、(ii) 手順 *LatterMove()* により、 $2n$ ラウンドの間、エージェントグループ A_{fb} と A_{bf} が前進または後進して、エージェント数が g 体未満のノードを訪問する. 以上の動作は $O(n)$ ラウンドを必要とし、総移動数は $O(gn)$ 回である. エージェントは g -部分集合を達成するまで、このような振る舞いを最大で $\lceil \log g \rceil$ 回繰り返すので、総ラウンド数は $O(n \log g)$ 、総移動数は $O(gn \log g)$ 回である. 従って定理は成立する. \square

Algorithm 2 集合フェーズにおけるエージェント a_i の振る舞い (v_j は a_i が現在滞在しているノード)

```
1:  $a_i.rounds \leftarrow 1$ 
2: while  $(a_i.rounds \leq 3n) \wedge (a_i.gatID \neq null)$  do
3:   if  $v_j.id \neq a_i.gatID$  then
4:      $v_j.nAgents \leftarrow v_j.nAgents - 1$ 
5:     ノード  $v_j$  から前進方向のノードへ移動を試みる { リンクの欠落によって移動できない場合がある }
6:      $v_j.nAgents \leftarrow v_j.nAgents + 1$ 
7:   end if
8:    $a_i.rounds \leftarrow a_i.rounds + 1$ 
9: end while
10: if  $v_j.id = a_i.gatID$  then
11:    $v_j.candi = true$ 
12: end if
13:  $a_i.rounds \leftarrow 1$ 
14: while  $(a_i.rounds \leq n)$  do
15:   if  $v_j.candi \neq true$  then
16:      $v_j.nAgents \leftarrow v_j.nAgents - 1$ 
17:     ノード  $v_j$  から前進方向のノードへ移動を試みる
18:      $v_j.nAgents \leftarrow v_j.nAgents + 1$ 
19:   end if
20:    $a_i.rounds \leftarrow a_i.rounds + 1$ 
21: end while
22:  $a_i.rounds \leftarrow 1$ 
23: while  $(a_i.rounds \leq n)$  do
24:   if  $v_j.candi \neq true$  then
25:      $v_j.nAgents \leftarrow v_j.nAgents - 1$ 
26:     ノード  $v_j$  から後進方向のノードへ移動を試みる
27:      $v_j.nAgents \leftarrow v_j.nAgents + 1$ 
28:   end if
29:    $a_i.rounds \leftarrow a_i.rounds + 1$ 
30: end while
31: キラリティの獲得を行う ( $v_j$  に滞在しているエージェントの前進方向を統一する)
32: if  $v_j.nAgents = k$  then
33:   全集合を達成しているため、アルゴリズムを終了
34: else if  $(v_j.nAgents \geq g) \wedge (k - v_j.nAgents \geq g)$  then
35:    $g$ -部分集合を達成しているため、アルゴリズムを終了
36: else if  $v_j.nAgents \geq g + 2$  then
37:    $More()$ 
38: else
39:    $Less()$ 
40: end if
```

Algorithm 3 手順 $More()$ (v_j は a_i が現在滞在しているノード)

```
1:  $a_i.rank$  の計算を行う
2: if  $1 \leq a_i.rank \leq g$  then
3:    $a_i.dir \leftarrow 0$ 
4: else if  $g + 1 \leq a_i.rank \leq \lfloor (v_j.nAgents - g)/2 \rfloor$  then
5:    $a_i.dir \leftarrow 1$ 
6: else
7:    $a_i.dir \leftarrow -1$ 
8: end if
9:  $Moving(a_i.dir)$ 
10: if  $v_j.nAgents < g$  then
11:    $a_i.dir \leftarrow 0, v_j.waiting \leftarrow true$ 
12: else if  $v_j.waiting = false$  then
13:    $a_i.dir \leftarrow 1$ 
14: else
15:    $a_i.dir \leftarrow -1, v_j.waiting \leftarrow false$ 
16: end if
17:  $LatterMove(a_i.dir)$ 
```

Algorithm 4 手順 $Less()$ (v_j は a_i が現在滞在しているノード)

```
1:  $a_i.rounds \leftarrow 1, v_j.waiting \leftarrow true$ 
2: while  $a_i.rounds \leq n$  do
3:   現在のノード  $v_j$  に留まる
4:    $a_i.rounds \leftarrow a_i.rounds + 1$ 
5: end while
6:  $a_i.dir \leftarrow -1, v_j.waiting \leftarrow false$ 
7:  $LatterMove(a_i.dir)$ 
```

4 $3g - 1 \leq k \leq 8g - 4$ の場合

本章では、 $3g - 1 \leq k \leq 8g - 4$ の時、動的リングにおける g -部分集合問題をラウンド数 $O(n)$ 、総移動数 $O(kn)$ で解くアルゴリズムを提案する。このアルゴリズムでは、全てのエージェントはリングを一周して全てのエージェントの ID を取得し、集合する共通のノード v_{gather} に集合しようとする。従って、第3章と同様にアルゴリズムは、選択フェーズ、集合フェーズの二つのフェーズから構成される。選

Algorithm 5 手順 $Moving(a_i.dir)$ (v_j は a_i が現在滞在しているノード)

```
1:  $a_i.rounds \leftarrow 1$ 
2: while  $a_i.rounds \leq n$  do
3:   if  $a_i.dir = 0$  then
4:     現在のノード  $v_j$  に留まる
5:   else
6:     while  $v_j.waiting = false$  do
7:        $v_j.nAgents \leftarrow v_j.nAgents - 1$ 
8:       ノード  $v_j$  から  $a_i.dir$  (1: 前進, -1: 後進) によって決定した方向のノードへ
       移動を試みる
9:        $v_j.nAgents \leftarrow v_j.nAgents + 1$ 
10:    end while
11:  end if
12:   $a_i.rounds \leftarrow a_i.rounds + 1$ 
13: end while
```

Algorithm 6 手順 $LatterMove$ (v_j は a_i が現在滞在しているノード)

```
1:  $Moving(a_i.dir)$ 
2:  $a_i.dir \leftarrow a_i.dir * (-1)$ 
3: if  $(v_j.nAgents \geq g) \wedge (k - v_j.nAgents \geq g)$  then
4:    $g$ -部分集合を達成しているため, アルゴリズムを終了
5: else if  $v_j.nAgents \geq g + 2$  then
6:    $More()$ 
7: else
8:    $Less()$ 
9: end if
```

択フェーズは, 第3章と全く同じであり, エージェントの総移動数を $O(gn \log g)$ から $O(gn)$ に減らすために, 集合フェーズを修正する.

エージェントが選択フェーズを終えた後, エージェントの配置によって, 次のように動作を分岐する. $3n$ ラウンド前進した後, (a) v_{gather} に留まっているエージェントは, 現在のノードが集合ノードであることをノードに書き込み, $2n$ ラウンドの間留まる. (b) $v_{semi-gather}$ に留まっているエージェントは, 一つのグループとして n ラウンドの間前進し, その後 n ラウンドの間後進する. その間に集合ノードであると記録されているノード (すなわち v_{gather}) に到達すると留まる. (c) v_{gather} に到達できなかったエージェントは, 補題1の議論と同様に一か所のノードに集合しているため, 一つのグループとして n ラウンドの間前進し, その後 n ラウンドの間後進する. その間に v_{gather} に到達するとそのノードに留まる. この時点で全てのエージェントが

v_{gather} に留まっていれば全集合を達成しているため、アルゴリズムを終了する。リンクの欠落によって、 v_{gather} に到達できなかったエーエージェントが存在する場合、それらが滞在するノードを v' とする。ここまでの処理は3.2節の集合フェーズと同様である。ただし、 v' (resp., v_{gather}) に g 体以下のエーエージェントのみ存在する場合、本章では $3g - 1 \leq k \leq 8g - 4$ の場合を考えるので、 v_{gather} (resp., v') には少なくとも $2g$ 体以上のエーエージェントが存在する。少なくとも $2g$ 体以上のエーエージェントが存在する。ノードを v_{more} と呼ぶ。そしてこの関係を利用し、スプリッティングを修正(単純化)する。直感的には、 v_{more} から少なくとも g 体以上ずつの二つのエーエージェントグループがそれぞれ前進、後進しようとする。さらに、それらのグループが g 体未満のエーエージェントが存在するノードを訪問した時、 g 体未満のエーエージェントはグループに合流し同じ方向に移動しようとする。この動作によって、エーエージェントが存在する各ノードに少なくとも g 体以上のエーエージェントが存在する状態で、アルゴリズムを終了することができる。

具体的には、 v_{more} に存在するエーエージェントの数を $k'(\geq 2g)$ とする。そして、 V_{more} にいる各エーエージェントは $a_i.rank$ を計算する。 $1 \leq a_i.rank \leq g$ が成り立つ場合、そのエーエージェントは前進グループ A_f に属し、前進しようとする。また、 $(k' < 3g) \vee (g+1 \geq a_i.rank \geq 2g)$ が成り立つ場合、後進グループ A_b に属し、後進しようとする。両方の条件を満たさない場合、 A_f と A_b が v_{more} から移動しても、少なくとも g 体のエーエージェントが存在するため、 v_{more} に残ったエーエージェントはアルゴリズムを終了する。 A_f と A_b がリング内を移動している間、 A_f (resp., A_b) が新しいノード v_j に訪問すると、 v_j が A_f (resp., A_b) によって訪問されたことを表すフラグ $v_j.fMarked$ (resp., $v_j.bMarked$) を $true$ にセットする。これらのフラグは、エーエージェントグループ A が現在のノードが他のグループによって訪問されたかどうかを確認するために使用され、すでにフラグがセットされていた場合、 A は移動を停止することができる。さらに、 A_f (resp., A_b) が g 未満のエーエージェントの存在するノードを訪問した時、 g 未満のエーエージェントは A_f (resp., A_b) に合流し、前進 (resp., 後進) を試みる。しかし、合流することによってグループ内のエーエージェントの数が $2g$ を超える可能性がある。この場合、ID を用いて g 体のエーエージェントだけが移動を試み続け、それ以外のエーエージェントは現在のノードでアルゴリズムを終了する。この動作により、各リンクを通過するエーエージェントは最大で $2g$ 体となり、エーエージェントグループの総移動回数を $O(gn)$ に減らすことができる。そして、 A_f (resp., A_b) が n ラウンドの間、 $v_j.fMarked = true$ (resp., $v_j.bMarked = true$) のノードを訪問するまで移動を繰り返すと、 A_f と A_b を合わせてリング内の全てのノードを訪問することができ、 g -部分集合を達成する。

図3に例を示す(説明に関係のないノードは省略している)。図3(a)から(b)にかけて、後進グループ A_b が $2(< g)$ 体以上のエーエージェントがいるノードを訪問し、2体のエーエージェントが A_b に加わる。その後、更新された A_b のエーエージェントの数は $7(> 2g)$ であるため、3体のエーエージェントのみが A_b として移動を続け、残りの4体のエーエージェントは、そのノードでアルゴリズムを終了する(図3(b)から(c))。図3(c)から(d)に

かけて、前進グループ A_f がリンクの欠落によってブロックされ続けていると仮定する。この場合でも、各ラウンドでリンクの欠落は一つまでなので、 A_b は移動を続けることができる。図 3(e) のように、 A_f (resp., A_b) がフラグを立てたノードを訪問するか、スプリッティングの動作を始めてから n ラウンド経過すると、エージェントは g -部分集合を達成する。

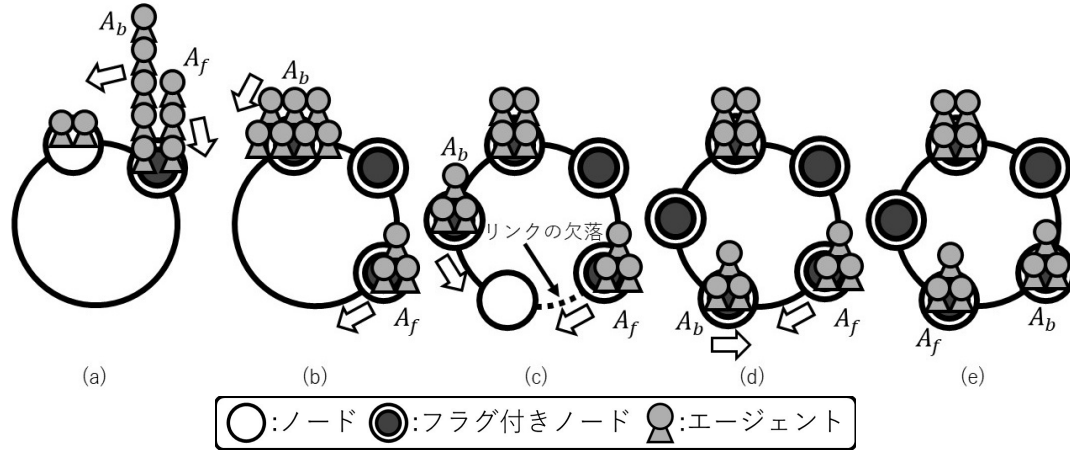


図 3 $3g - 1 \leq k \leq 8g - 4$ の場合の集合フェーズの実行例

集合フェーズの疑似コードをアルゴリズム 7 に示す。このフェーズの最初の $3n$ ラウンドとその後の n ラウンドの移動を終えた後、各エージェント a_i は $v_j.nAgents$ によって現在のノードのエージェント数をカウントする。その数が、少なくとも $2g$ (すなわち、 v_j が $vmore$) であれば、 a_i は手順 $More()$ を実行するし、 $a_i.rank$ に応じて、 A_f か A_b に属するか、アルゴリズムを終了するかを決定する。 $More()$ の疑似コードをアルゴリズム 8 に示す。 v_j のエージェント数が g 未満の場合、 a_i は手順 $Less2()$ を実行し、グループ A_f もしくは A_b が v_j を訪問するまで v_j に留まり続ける。 $Less2()$ の疑似コードをアルゴリズム 9 に示す。アルゴリズム 8, 9 において、エージェントはリング内を移動するために $Moving()$ を使用する (アルゴリズム 10)。

また、前進グループと後進グループが、あるノードを同時に訪問する可能性がある。これは、そのノード以外の全てのノードは二つのグループのどちらかが訪問済みであること意味するので、その時点でアルゴリズムを終了する (アルゴリズム 9 の 5 行目, アルゴリズム 10 の 12 行目)。さらに、 A_f と A_b が移動しようとするとき、ある一つのリンクが欠落し続けている可能性がある。 A_f と A_b は同じリンク e' によってブロックされ続ける場合、それまでの n ラウンドの間の移動によって A_f と A_b で合わせて n 個のノードを訪問することができる。したがって、エージェントがフラグを観測していなくても、移動を開始してから n ラウンド後にアルゴリズムを終了することができる (アルゴリズム 10 の 17 行目)。

Algorithm 7 集合フェーズにおけるエージェント a_i の振る舞い (v_j は a_i が現在滞在しているノード)

```
1:  $a_i.rounds \leftarrow 1$ 
2: while  $(a_i.rounds \leq 3n) \wedge (a_i.gatID \neq null)$  do
3:   if  $v_j.id \neq a_i.gatID$  then
4:      $v_j.nAgents \leftarrow v_j.nAgents - 1$ 
5:     ノード  $v_j$  から前進方向のノードへ移動を試みる { リンクの欠落によって移動できない場合がある }
6:      $v_j.nAgents \leftarrow v_j.nAgents + 1$ 
7:   end if
8:    $a_i.rounds \leftarrow a_i.rounds + 1$ 
9: end while
10: if  $v_j.id = a_i.gatID$  then
11:    $v_j.candi = true$ 
12: end if
13:  $a_i.rounds \leftarrow 1$ 
14: while  $(a_i.rounds \leq n)$  do
15:   if  $v_j.candi \neq true$  then
16:      $v_j.nAgents \leftarrow v_j.nAgents - 1$ 
17:     ノード  $v_j$  から前進方向のノードへ移動を試みる
18:      $v_j.nAgents \leftarrow v_j.nAgents + 1$ 
19:   end if
20:    $a_i.rounds \leftarrow a_i.rounds + 1$ 
21: end while
22:  $a_i.rounds \leftarrow 1$ 
23: while  $(a_i.rounds \leq n)$  do
24:   if  $v_j.candi \neq true$  then
25:      $v_j.nAgents \leftarrow v_j.nAgents - 1$ 
26:     ノード  $v_j$  から後進方向のノードへ移動を試みる
27:      $v_j.nAgents \leftarrow v_j.nAgents + 1$ 
28:   end if
29:    $a_i.rounds \leftarrow a_i.rounds + 1$ 
30: end while
31: キラリティの獲得を行う ( $v_j$  に滞在しているエージェントの前進方向を統一する)
32: if  $g \leq v_j.nAgents \leq 2g - 1$  then
33:    $g$ -部分集合を達成しているため, アルゴリズムを終了
34: else if  $v_j.nAgents \geq 2g$  then
35:    $More2()$ 
36: else
37:    $Less2()$ 
38: end if
```

Algorithm 8 手順 *More2()* (v_j は a_i が現在滞在しているノード)

```
1:  $a_i.\text{rounds} \leftarrow 1$ 
2:  $a_i.\text{rank}$  の計算を行う
3: if  $1 \leq a_i.\text{rank} \leq g$  then
4:    $a_i.\text{dir} \leftarrow 1$ 
5: else if  $(v_j.n\text{Agents} < 3g) \vee (g+1 \leq a_i.\text{rank} \leq 2g)$  then
6:    $a_i.\text{dir} \leftarrow -1$ 
7: else
8:    $A_f$  と  $A_b$  が離れた後も  $g$  体以上のエージェントが残るため、アルゴリズムを
     終了
9: end if
10: Moving2()
```

集合フェーズにて以下の補題3がある.

補題 3. アルゴリズム 7が終了した時, エージェントは g -部分集合を達成している

Proof. まず, 集合フェーズの開始から, $3n$ ラウンドの間, 各エージェントは v_{gather} に到達するために前進しようとする. そして, 補題1と同様の議論により, v_{gather} に到達しなかった前進方向が同じエージェントは一つのノードに集まる. その後, v_{gather} に到達しなかったエージェントのグループが v_{gather} に到達するまで n ラウンドの間前進, n ラウンドの間後進することで, 全てのエージェントは v_{gather} か v' に留まっている. 選択フェーズにて二つの $v_{semi-gather}$ が決定され v_{gather} が存在しない場合もエージェントの動作は同様なので, 二つのエージェントグループが存在するノードをそれぞれ v_{gather} と v' とみなす. 少なくとも g 体のエージェントが v_{gather} と v' に存在するとき, すでに g -部分集合を達成しているのでアルゴリズムを終了する. 一方, v' (resp., v_{gather}) に g 体未満のエージェントしか存在しないとき, $3g-1 \leq k \leq 8g-4$ の場合の場合を考えるので, v_{gather} (resp., v') には少なくとも $2g$ 体のエージェントが存在する. 従って, v_{gather} または v' にいるエージェントは *More2()* を実行した後 *Moving2()* を実行する. また, *Moving2()* を実行した前進グループ A_f または後進グループ A_b が g 体以下のエージェントのいるノードを訪問した場合, g 体以下のエージェントはグループに合流する. 更新されたグループのエージェント数が $2g$ 以上であれば, ID を用いて g 体のエージェントだけが移動を続け, 残りのエージェントはアルゴリズムを終了する. 従って, A_f または A_b が離れた後でも, そのノードには g 体以上のエージェントが残される. その後さらに A_f は前進, A_b は後進を続けようとするため, 少なくともどちらかは各ラウンドで次のノードを訪問することができる. 従って, n ラウンド以内にリング内の全てのノードを A_f または A_b が訪問し, エージェントが存在するノードには g 体以上のエージェントが存在することになる. よって, 補題が成立する. \square

Algorithm 9 手順 $Less2()$ (v_j は a_i が現在滞在しているノード)

```
1:  $a_i.rounds \leftarrow 1, v_j.waiting \leftarrow true$ 
2: while  $true$  do
3:    $a_i.rounds \leftarrow a_i.rounds + 1$ 
4:   if  $(v_j.fMarked = true) \wedge (v_j.bMarked = true)$  then
5:      $A_f$  と  $A_b$  のどちらもが訪問しているため、アルゴリズムを終了
6:   end if
7:   if  $(v_j.fMarked = true) \vee (v_j.bMarked = true)$  then
8:      $a_i.rank$  の計算を行う
9:     if  $(v_j.nAgents \geq 2g) \wedge (a_i.rank \geq g + 1)$  then
10:      アルゴリズムを終了
11:   else
12:      $a_i.dir \leftarrow v_j.dir, Moving2()$ 
13:   end if
14: end if
15: end while
```

提案アルゴリズムについて、以下の定理 4 を示す.

定理 4. $3g - 1 \leq k \leq 8g - 4$ が成立するとき、提案アルゴリズムは動的リングにおける g -部分集合問題を $O(n)$ ラウンド、総移動数 $O(kn)$ で解く.

Proof. 補題 1 と補題 3 により、提案アルゴリズムは g -部分集合問題を解くことができる. 以下では、問題を解くまでの総ラウンド数とエージェントの総移動数を解析する.

まず、選択フェーズでは、各エージェントは集合ノード v_{gather} を決定するために $3n$ ラウンドの間前進しようとする. ここでは $k = O(n)$ が成り立つので、 $O(n)$ ラウンド、総移動数 $O(gn)$ が必要となる. 次に、集合フェーズでは、始めに $3n$ ラウンドとその後の $2n$ ラウンドの間、各エージェントは v_{gather} に到達するために前進または後進しようとするが、これには $O(n)$ ラウンドと総移動数 $O(gn)$ が必要となる. その後、あるノードに少なくとも $2g$ 以上のエージェントが存在すると、そこから前進エージェントグループ A_f と後進エージェントグループ A_b がリング内を移動し、 g -部分集合を達成する. 補題 3 の証明で述べたように、 A_f と A_b の少なくともどちらかは各ラウンドで次のノードに移動することができるので、これらのグループの移動は n ラウンド以内に終了する. また、 A_f と A_b のエージェント数は最大で $2g$ であり、 A_f と A_b は合計で n , $n + 1$, または $n + 2$ 個のノードを訪問するため、エージェントグループの移動中に各リンクを通過する回数は最大 $4g$ である. したがって、エージェントは総ラウンド数は $O(n)$ 、総移動数は $O(gn)$ で g -部分集合を達成し、定理は成立する. \square

5 $k \geq 8g - 3$ の場合

本章では、 $k \geq 8g - 3$ の時、動的リングにおける g -部分集合問題をラウンド数 $O(kn/g)$ 、総移動数 $O(gn)$ で解くアルゴリズムを提案する。総移動数が $O(gn)$ であるということを考えると、全てのエージェントがリングを一周することなく g -部分集合を達成する必要がある。従って、この章ではエージェントが ID を持つことと $k \geq 8g - 3$ であることを用いて、総移動数を減らすことを目指す。アルゴリズムは、四つのフェーズからなる: キラリティ統一フェーズ、準選択フェーズ、準集合フェーズ、達成フェーズ。キラリティ統一フェーズでは、エージェント間のキラリティの統一を行う。準選択フェーズでは、エージェントは少なくとも $2g$ 体のエージェントが集まる可能性のある集合候補ノードの集合を選択する。準集合フェーズでは、エージェントは集合候補ノードに移動しようとし、結果的に少なくとも一つのノードには $2g$ 体以上のエージェントが留まる。達成フェーズでは、第4章の集合フェーズと同様の方法で、 g -部分集合を達成する。

5.1 キラリティ統一フェーズ

このフェーズの目的は、リング上の全てのエージェントの進行方向が同じになる、つまりエージェント間にキラリティを獲得することである。以降、エージェントの ID は $1 \sim ck (c \geq 1)$ の中から定められると仮定する。このフェーズは前後半の二つのパートで構成される。前半パートでは、少なくとも 1 組 (2 体) のエージェントが交差する、あるいは同じノードに集まる状況を作る。後半パートでは、交差もしくは同じノードに集まったエージェントの組がそれぞれ別方向に移動し自身のキラリティを伝えることで、エージェント全体のキラリティを統一する。より具体的に、まず前半パートでは $3n$ と $3n$ で合計 $6n$ ラウンドをかけてエージェントの組を作成する。最初の $3n$ ラウンドの間、エージェントは自身のものを含め ID を 3 個収集するまで前進しようとする。そしてこの移動の途中で他のエージェントと交差する、あるいは同じノードに留まっていることを検知した場合、前進を中止し現在のノードに留まり続ける。エージェントはノード v_j から移動する際に、移動先のノードが v_{j+1} (resp., v_{j-1}) ならば $v_j.rID$ (resp., $v_j.lID$) に自身の ID を上書きし、 $v_j.lID$ (resp., $v_j.rID$) に書き込まれている ID を消去する。そして、移動先のノード v_{j+1} (resp., v_{j-1}) の $v_{j+1}.lID$ (resp., $v_{j-1}.rID$) に他のエージェントの ID が記録されていれば、その ID のエージェントと交差していることを検知する。そして、自身の他のエージェントとの交差を示すフラグ $a_i.crs$ を初期化する。ID が 3 個収集できた場合、移動を停止して 3 個のうち 1 番目と 2 番目の ID を比較する。1 番目の ID の方が大きい場合は前進方向はそのまま、2 番目の ID の方が大きい場合は自身の前進方向を反転させる。そして、2 回目の $3n$ ラウンドの間、もう一度、エージェントは自身のものを含め ID を 3 個収集するまで前進しようとする。そしてこの移動の途中で

他のエージェントと交差する，あるいは同じノードに留まっていることを検知した場合，前進を中止し現在のノードに留まり続ける．合計 $6n$ ラウンドの移動が終了した後，他のエージェントとの交差を検知したエージェントは交差した相手と自身の ID を比較し，小さい方の ID を $a_i.minID$ に記録する．他のエージェントと同じノードに留まっているエージェントは，同一ノード内のエージェントと ID を比較して最小の ID を $a_i.minID$ に記録し，前進方向を最小の ID のエージェントの前進方向と反対向きにする．キラリティ統一フェーズの前半パートの疑似コードを Algorithm11 に示す．

Algorithm 11 キラリティ統一フェーズの前半パートにおけるエージェント a_i の振る舞い (v_j は a_i が現在滞在しているノード)

```

1:  $v_j.id \leftarrow a_i.id, a_i.ids[a_i.nIDs] \leftarrow v_j.id$ 
2:  $a_i.nIDs \leftarrow a_i.nIDs + 1, v_j.nAgents \leftarrow v_j.nAgents + 1$ 
3: while  $a_i.rounds \leq 6n$  do
4:   if  $a_i.crs = false$  then
5:     if (前進方向が  $v_{j+1} \wedge v_j.lID \neq 0 \wedge v_j.lID \neq a_i.id$ ) then  $a_i.crs \leftarrow true$ 
6:     else if (前進方向が  $v_{j-1} \wedge v_j.rID \neq 0 \wedge v_j.rID \neq a_i.id$ ) then  $a_i.crs \leftarrow true$ 
7:     else if ( $(a_i.nIDs < 3) \wedge (v_j.nAgents = 1)$ ) then
8:        $v_j.nAgents \leftarrow v_j.nAgents - 1$ 
9:       if (前進方向が  $v_{j+1}$ ) then  $v_j.rID \leftarrow a_i.id, v_j.lID \leftarrow 0$ 
10:      else if (前進方向が  $v_{j-1}$ ) then  $v_j.lID \leftarrow a_i.id, v_j.rID \leftarrow 0$ 
11:      ノード  $v_j$  から前進方向のノードへ移動を試みる
12:      if  $a_i$  が前進方向の新しいノード  $v_j$  へ移動した then
13:         $a_i.nVisited \leftarrow a_i.nVisited + 1$ 
14:        if  $v_j.id \neq \perp$  then
15:           $a_i.ids[a_i.nIDs] \leftarrow v_j.id, a_i.nIDs \leftarrow a_i.nIDs + 1$ 
16:        end if
17:      end if
18:    end if
19:  end if
20:   $a_i.rounds \leftarrow a_i.rounds + 1$ 
21:  if  $a_i.nIDs = 3 \wedge a_i.rounds = 3n$  then
22:    if  $a_i.ids[1] < a_i.ids[2]$  then
23:      前進方向を反転する,  $a_i.nIDs \leftarrow 0$ 
24:    end if
25:  end if
26: end while
27: if  $a_i.crs = true$  then
28:   if (前進方向が  $v_{j+1}$ ) then  $a_i.minID \leftarrow \min(a_i.id, v_j.lID)$ 
29:   else if (前進方向が  $v_{j-1}$ ) then  $a_i.minID \leftarrow \min(a_i.id, v_j.rID)$  end if
30: end if
31: if  $v_j.nAgents > 1$  then
32:    $a_i.minID \leftarrow \min(\text{同一ノードに滞在するエージェントの中で最小の } ID)$ 
33:   if (最小の ID のエージェントと前進方向が同じ) then 前進方向を反転する
34:   end if
35: end if

```

前半パートの動作によって、次の補題4が成り立つ(証明は省略する).

補題 4. キラリティ統一フェーズの前半パートを実行後、少なくとも1組のエージェントが交差する、あるいは同じノードに集まる

後半パートでは、前半パートでできたエージェントの組ごとに移動を行う。組の中で最小のIDを持つエージェントと、もう1体のエージェントがそれぞれ反対の方向に前進し、最小のエージェントの前進方向を訪問したノードに書き込んでいくことで、最終的に全体のキラリティを統一する。具体的にまず、後半パートの開始時に、 $v_j.lID$ と $v_j.rID$ の初期化し、 $a_i.minID$ が自身のIDではないエージェントのいずれかが $v_j.id$ に自身のIDを書き込み、前進方向が統一されていることを示すフラグ $v_j.uni$ を設定する。そして、自身の他のエージェントとの交差を示すフラグ $a_i.crs$ を初期化する。次に $\lceil ckn/g \rceil$ ラウンドの間、 $a_i.minID$ が自身のIDと同じエージェントと、 $v_j.id$ にIDを書き込んだエージェントのみが動作する。自身に記録した $a_i.minID$ が、 $(i-1) \times g < a_i.minID \leq i \times g$ ($1 \leq i \leq k$) の範囲であれば、それらのエージェントは、 $(i-1)n+1$ から in ラウンド目までの間のみ移動を行う。移動するエージェントは、ノードを離れようとする時、記録した $a_i.minID$ を書き込んでいく。IDを書き込む時、 v_j から v_{j+1} (resp., v_{j-1}) に移動する場合、書き込むIDが自身のIDであれば $v_j.rID$ (resp., $v_j.lID$) に、そうでなければ $v_j.lID$ (resp., $v_j.rID$) に上書きする。同時に、前進方向が統一されていることを示すフラグ $v_j.uni$ を設定する。そして新しいノードに移動した時、 $v_j.uni$ がすでに設定されており、かつ $v_j.lID$ か $v_j.rID$ に $a_i.minID$ 以下のIDが書き込まれているならば、交差を検知したとしてフラグ $a_i.crs$ を設定する。そうでなければ、 $v_j.id$ に自身のIDを書き込む。移動しているエージェントは、移動を始めて n ラウンド経過する、または交差の検知を示すフラグが設定されている場合、移動を終了する。そして、 $\lceil ckn/g \rceil$ ラウンドの内 n ラウンドが経過するたびに、移動していないエージェントを含める全てのエージェントは、現在のノードの $v_j.lID$ か $v_j.rID$ にIDが書き込まれている場合、それ以降の移動を中止し、準選択フェーズへ移行する。 $\lceil ckn/g \rceil$ ラウンド経過するか、移動が中止された時点で、現在のノードの $v_j.lID$ と $v_j.rID$ のどちらにIDが書き込まれているかによって前進方向を決定する。この時点で、全てのエージェント間でキラリティの統一が完了する。

キラリティ統一フェーズの疑似コードを Algorithm12 に示す。

Algorithm 12 キラリティ統一フェーズの後半パートにおけるエージェント a_i の振る舞い (v_j は a_i が現在滞在しているノード)

```

1: int  $i \leftarrow 1$ 
2:  $a_i.rounds \leftarrow 1, a_i.crs \leftarrow false$ 
3:  $v_j.lID \leftarrow 0, v_j.rID \leftarrow 0$ 
4: if ( $a_i.id \neq a_i.minID$ ) then  $v_j.id \leftarrow a_i.id$  end if
5: while  $a_i.rounds \leq \lceil ckn/g \rceil$  do
6:   if  $a_i.minID > (i-1)g \wedge a_i.minID \leq ig$  then
7:     if ( $v_j.id = a_i.id \vee a_i.minID = a_i.id$ )  $\wedge a_i.crs = false$  then
8:        $v_j.nAgents \leftarrow v_j.nAgents - 1$ 
9:       if (前進方向が  $v_{j+1} \wedge a_i.id = a_i.minID$ )  $\vee$  (前進方向が  $v_{j-1} \wedge a_i.id \neq$ 
         $a_i.minID$ ) then
10:         $v_j.rID \leftarrow a_i.id, v_j.lID \leftarrow 0, v_j.uni \leftarrow true$ 
11:      else if (前進方向が  $v_{j-1} \wedge a_i.id = a_i.minID$ )  $\vee$  (前進方向が  $v_{j+1} \wedge a_i.id \neq$ 
         $a_i.minID$ ) then
12:         $v_j.lID \leftarrow a_i.id, v_j.rID \leftarrow 0, v_j.uni \leftarrow true$ 
13:      end if
14:      ノード  $v_j$  から前進方向のノードへ移動を試みる
15:      if  $a_i$  が前進方向の新しいノード  $v_j$  へ移動した then
16:         $a_i.nVisited \leftarrow a_i.nVisited + 1$ 
17:        if ( $v_j.uni = true$ )  $\wedge$  ( $v_j.lID \leq a_i.minID \vee v_j.rID \leq a_i.minID$ ) then
18:           $a_i.crs \leftarrow true$ 
19:        else
20:           $v_j.id \leftarrow a_i.id$ 
21:        end if
22:      end if
23:    end if
24:  end if
25:  if  $a_i.rounds \bmod n = 0$  then
26:     $i \leftarrow i + 1$ 
27:    if ( $v_j.uni = true$ ) then break end if
28:  end if
29:   $a_i.rounds \leftarrow a_i.rounds + 1$ 
30: end while
31: if ( $v_j.lID \neq 0$ ) then 前進方向を  $v_{j-1}$  に設定
32: else if ( $v_j.rID \neq 0$ ) then 前進方向を  $v_{j+1}$  に設定
33: end if

```

後半パートの実行後、次の補題5が成り立つ(証明は省略する)。

補題 5. キラリティ統一フェーズの後半パートを実行後、全てのエージェント間でキラリティが統一されている。

5.2 準選択フェーズ

補題5によりエージェント間のキラリティが統一されたため、このフェーズでは先行研究[11]の手法を応用して、少なくとも $2g$ 以上のエージェントが集まる可能性のある集合候補ノードの集合を選択することである。考えられる素朴なアプローチは、各エージェント a_i が前方 $1, 2, \dots, 2g-1$ 番目までのエージェントと後方 $1, 2, \dots, 2g-1$ 番目までのエージェントのIDを収集して初期のノードに戻る。ここで、エージェント a の $i(i \neq 0)$ 番目に前方(後方)のエージェント a' は、初期状況において a の前方(後方)方向に、 a と a' の間に $i-1$ 体のエージェントが存在するようなエージェントを表す。そして、収集した自身のIDをふくめ $4g-1$ 個のIDと比較して自身のIDが最小であれば、初期のノードを集合候補ノード v_{candi} に設定する。すると、 a_i の後方に存在する $2g-1$ 体のエージェントは v_{candi} に集まろうと前進し、最終的に $2g$ 体のエージェントが v_{gather} に留まる可能性がある。しかし、1-インターバル連結リングであることから、次の問題がある:(1) リンクの欠落によって $4g-1$ 個のIDを収集できず、 v_{candi} が選択されない可能性がある。(2) v_{candi} が選択されたとしても、リンクの欠落によって v_{candi} に到達できず、各ノードに $2g$ 体未満のエージェントしか集まらない可能性がある。これらの問題を処理するために、このフェーズでは、各エージェント a_i は前進を続け、 $4g-1$ 個より多くのIDを観測しようとし、必要な数のIDを観測したら、観測したIDを用いて v_{candi} を選択し、現在のノード v_j で $v_j.candi = true$ に変更する。

具体的には、まず移動を始める前に先行研究[11]と異なり、統一フェーズによって準選択フェーズ開始時に同じノードに複数のエージェントが滞在している可能性がある。それらのエージェントは、滞在している中で自身が何番目に小さいIDかを $a_i.rank$ として計算し、自身のIDを現在のノードの $v_j.ids[a_i.rank]$ に書き込んでいく。同じのエージェント全員のIDが $v_j.ids[]$ に書き込まれた後、収集したID列を保存する $a_i.ids[]$ に、 $v_j.ids[a_i.rank]$ の自身のIDから始まりそれより大きいIDを順に記録していく。この時、 $v_j.ids[]$ に書き込まれたIDを準選択フェーズにおける v_j の初期ID列とし、エージェントが移動し訪問先のノードでIDを収集する際は、 $v_j.ids[]$ の先頭から順にIDを収集していく。初期ID列の決定についての例を図4に示す。

初期ID列の設定が終わった後、 $3n$ ラウンドの間、各エージェント a_i は $10g-4$ 個のIDを観測するか、現在のノードに $2g$ 体以上のエージェントが存在するまで前進しようとする。その後、 a_i は少なくとも $8g-3$ 個のIDを観測したかどうかによっ

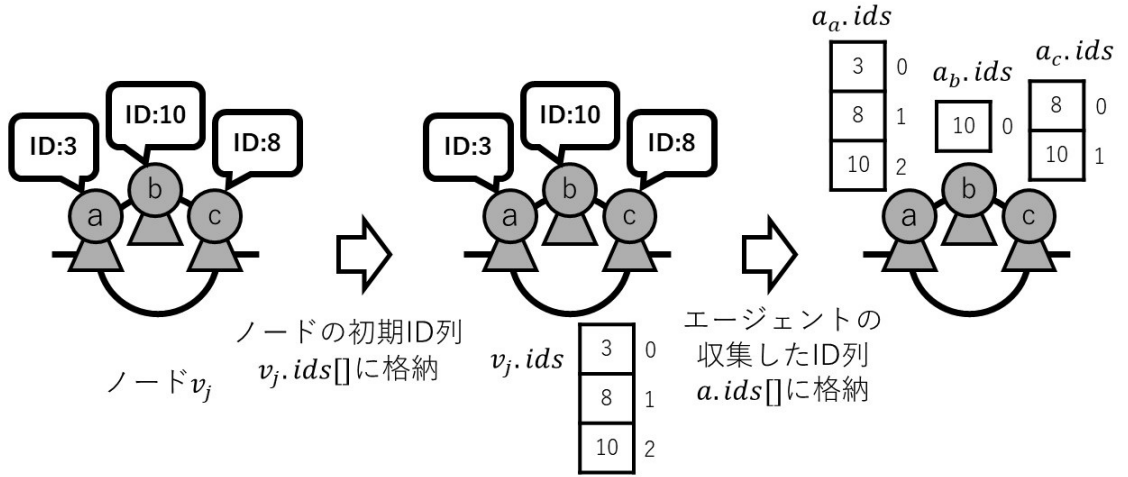


図 4 準選択フェーズにおけるノードの初期 ID 列の格納の例

て行動を決定する．もし a_i が $8g - 3$ 個の ID を観測しなかった場合，あるノード v_j に $2g$ 体以上のエージェントが存在することを補題 6 で示し，フラグ $v_j.candi$ を $true$ に設定して v_j が集合候補ノード v_{candi} であることを表す (問題 (1) が解決する)．これは， a_i が少なくとも $(10g - 3) - (bg - 3) = 2g$ の ID を観測していないため， a_i の後方エージェントも必要な ID を観測しておらず， a_i と同じノードに留まっているからである．一方，少なくとも $8g - 3$ 個の ID を観測した場合，最初の $8g - 3$ 個の ID を比較に使用し， $4g - 1$ 番目の ID を自身の ID とみなす．この時， a_i は $4g - 2$ 個の前方の ID と $4g - 2$ 個の後方の ID と自身の ID を比較することになる．そして， $4g - 1$ 番目の ID が最小である場合， a_i は現在のノードで $v_j.candi$ を $true$ に設定する． $k \geq 8g - 3$ が成り立ち，観測した ID の内少なくとも $8g - 3$ 番目までの ID はすべて異なるので， a_i の後方 $4g - 2$ 個のエージェントは a_i の滞在ノードを前進方向の最も近い集合候補ノードとして認識することができる．従って， $4g - 1$ 体のエージェント ($a_i + 4g - 2$ 体) のエージェントは v_{candi} を訪問するまで前進しようとする．その途中でリンクが欠落し続けた場合，エージェントは二つのグループに分割されるが，少なくとも一つのグループには $2g$ 体のエージェントが存在する (問題 (2) が解決する)．

準選択フェーズの疑似コードをアルゴリズム 13 に示す．準選択フェーズにて，以下の補題 6 がある．

補題 6. アルゴリズム 13 の終了後， $v_j.candi = true$ のノードが少なくとも一つ存在する．

Proof. 全エージェントの中で ID が最小のエージェントを a_{min} ， a_{min} の $(4g - 2)$ 番目の後方エージェントを a_i とする．アルゴリズム 13 を実行した後の $a_i.nIDs$ の値が (a) $8g - 3$ 未満，(b) $8g - 3$ 以上の値になる場合を順に考える．まず，(a) $a_i.nIDs < 8g - 3$

が成立する場合, a_i が観測できなかった ID の数を $g0 = (10g - 4) - a_i.nIDs$ とし, $a_{i-1}, a_{i-2}, \dots, a_{i-(g'-1)}$ を 1-st, 2-nd, ..., $(g'-1)$ -st とする. この時, $(g'-1) + a_i.nIDs = ((10g - 4) - a_i.nIDs) - 1 + a_i.nIDs = 10g - 5 < 10g - 4$ であるから, エージェント $a_{i-(g'-1)}$ は必要数 $10g - 4$ 個の ID を観測していない. 従って, $a_{i-1}, a_{i-2}, \dots, a_{i-(g'-1)}$ もまた同様に $10g - 4$ 未満の ID を観測しており, 補題 1 と同様の議論により同じノードに留まっている. $g' - 1 \geq (10g - 4) - (8g - 4) - 1 = 2g - 1$ が成り立つので, 少なくとも $2g$ 体のエージェントが同じノードに滞在し, 従って $v_j.candi$ が *true* に設定される. 次に, (b) $a_i.nIDs \geq 8g - 3$ が成立する場合, a_i は a_{min} の ID が自身の ID となり, その ID が $8g - 3$ 個の ID の中で最小であることを認識する. 従って, a_i は現在のノード v_j で $v_j.candi = true$ を設定する. よって, 補題は成立する. \square

Algorithm 13 準選択フェーズにおけるエージェント a_i の振る舞い (v_j は a_i が現在滞在しているノード)

```

1:  $a_i.rank$  の計算を行う,  $v_j.ids[a_i.rank] \leftarrow a_i.id, a_i.nIDs \leftarrow 0$ 
2: for  $i = 0; i < v_j.nAgents - a_i.rank; i++$  do
3:    $a_i.ids[i] \leftarrow v_j.ids[a_i.rank + i], a_i.nIDs \leftarrow a_i.nIDs + 1$ 
4: end for
5:  $a_i.rounds \leftarrow 0$ 
6: while  $a_i.rounds \leq 3n$  do
7:   if  $(a_i.nIDs < 10g - 4) \wedge (v_j.nAgents < 2g)$  then
8:      $v_j.nAgents \leftarrow v_j.nAgents - 1$ 
9:     ノード  $v_j$  から前進方向のノードへ移動を試みる
10:    if  $(a_i$  が前進方向の新しいノードへ移動した)  $\wedge (v_j.id \neq \perp)$  then
11:      for  $i = 0; i < v_j.nAgents; i++$  do
12:         $a_i.ids[a_i.nIDs] \leftarrow v_j.ids[i], a_i.nIDs \leftarrow a_i.nIDs + 1$ 
13:      end for
14:    end if
15:     $v_j.nAgents \leftarrow v_j.nAgents + 1, a_i.rounds \leftarrow a_i.rounds + 1$ 
16:  end if
17: end while
18: if  $(v_j.nAgents \geq 2g) \vee ((a_i.nIDs \geq 8g - 3) \wedge (\forall h \in [0, 8g - 2] \setminus 4g - 2; a_i.ids[4g - 2] < a_i.ud[h]))$  then
19:    $v_j.candi \leftarrow true$ 
20: end if

```

5.3 準集合フェーズ

このフェーズでは、エージェントはあるノードに少なくとも $2g$ 体のエージェントが存在するような状況を作ることを目指す。補題 6 より、準選択フェーズの終了時に、 $v_j.candi = true$ である集合候補ノード v_j が少なくとも一つ存在する。以下では、このような候補ノードを v_{candi} と呼ぶ。そして、各エージェントは前進方向の最も近い v_{candi} まで移動し、到達すると留まる。具体的には、このフェーズでは $3n$ ラウンドの間、各エージェントは v_{candi} に留まるか、少なくとも $2g$ 体のエージェントが存在するまで前進しようとする。リンクの欠落によって、 v_{candi} に到達できないエージェントが存在する可能性があるが、1 と同様に、 v_{candi} に到達できなかった前進方向が同じエージェントは、一つのノードに留まっている。この時点で、少なくとも一つのノードには $2g$ 体以上のエージェントが存在することを補題 7 にて示す。従って、エージェントはあるノードに少なくとも $2g$ 体のエージェントが存在するような状況を作ることができる。準集合フェーズの疑似コードをアルゴリズム 14 に示す。

準集合フェーズにて、以下の補題 7 がある。

補題 7. 準集合フェーズを終えた時、 $v_j.nAgents \geq 2g$ を持つノード v_j が少なくとも一つ存在する。

Proof. 準集合フェーズの開始時、準選択フェーズにて条件 (i) 満たして決定された v_{candi} が存在する場合、すでにその v_{candi} には $2g$ 体以上のエージェントが存在する。また、条件 (ii) 決定された v_{candi} のみ存在する場合、 v_{candi} が一つだけのとき補題 1 と同様の議論で、 $8g - 3$ 体以上のエージェントは、 v_{candi} が欠落したリンクの両端の二つのノードの合わせて三つのノードいずれかに留まる。 $8g - 3$ 体のエージェントを三つのノードに分割したとしても、そのうちの一つには $2g$ 体以上のエージェントが存在する。 v_{candi} が複数のとき、それぞれの v_{candi} の間には $4g - 2$ 体以上のエージェントが存在する。その区間において、二つの v_{candi} に最初から滞在するエージェントとその間に存在するエージェントを合わせて $4g$ 体以上のエージェントが存在する。欠落が起こったとしても、 v_{candi} 間には $4g$ 体以上いることから、時計回りのグループか反時計回りのグループのどちらかは $2g$ 体以上なので、どちらかの v_{candi} には $2g$ 体以上のエージェントが存在する。よって全ての場合で $v_j.nAgents \geq 2g$ を持つノード v_j が少なくとも一つ存在するため、補題が成立する。 \square

Algorithm 14 準集合フェーズにおけるエージェント a_i の振る舞い (v_j は a_i が現在滞在しているノード)

```

1:  $a_i.\text{rounds} \leftarrow 1$ 
2: while  $a_i.\text{rounds} \leq 3n$  do
3:   if  $v_j.\text{candi} = \text{false}$  then
4:      $v_j.n\text{Agents} \leftarrow v_j.n\text{Agents} - 1$ 
5:     ノード  $v_j$  から前進方向のノードへ移動を試みる
6:     if  $a_i$  が前進方向の新しいノードへ移動した then
7:        $v_j.n\text{Agents} \leftarrow v_j.n\text{Agents} + 1$ 
8:     end if
9:     if  $v_j.n\text{Agents} \geq 2g$  then
10:       $v_j.\text{candi} \leftarrow \text{true}$ 
11:    end if
12:  end if
13:   $a_i.\text{rounds} \leftarrow a_i.\text{rounds} + 1$ 
14: end while

```

5.4 達成フェーズ

このフェーズでは、エージェントが g 体の部分集合を目指す。補題 7 によって、第 4 章と同様に、 $2g$ 体以上のエージェントをもつノードが少なくとも一つ存在する。第 4 章と異なるのは、エージェントをもつノードが二つ以上存在し、それぞれが $2g$ 体以上のエージェントを持つ可能性があることである。すなわち、(1) $2g$ 体以上のエージェントがいるノードに滞在するエージェントは、前進グループと後進グループに分割され、それぞれ前進と後進しようとする。(2) 前進グループ (resp., 後進グループ) が g 未満のエージェントはのいるノードを訪れた場合、 g 未満のエージェントは前進グループ (resp., 後進グループ) に合流する。

例を図 5 に示す。図 5(a) において、二つのノード v_p と v_q が存在し、それぞれ $6(\geq 2g)$ 体のエージェントが存在する。従って、前進グループ A_{f_p} と後進グループ A_{b_p} (resp., A_{f_q} と A_{b_q}) はそれぞれ v_p (resp., v_q) から移動を開始する。(a) から (b) にかけて A_{b_p} は g 体未満のエージェントをもつ v_l に到達し、 g 体未満のエージェントは合流し後進を続ける。(b) から (d) にかけて A_{f_q} はリンクの欠落によってブロックされ続けている。(b) から (c) にかけて、 A_{f_p} と A_{b_q} は交差し、フラグによって交差したことを検知してアルゴリズムを終了する。(c) から (d) にかけて A_{b_p} は g 体未満のエージェントをもつ v_m に到達し、 g 体未満のエージェントは合流し後進を続けようとする。そして、更新されたエージェント数が $7(\geq 2g)$ であるため、ID を用いて g 体のエージェントのみ後進を続け、残りのエージェントはアルゴリズムを終了する。

((d) から (e)). この動作により、達成フェーズは各リンクを最大 $2g$ のエージェントが通過し、 $O(gn)$ の移動回数で完了できる．そして、(e) から (f) にかけて A_{f_q} と A_{b_p} は同時にあるノードに到達し、フラグによって合流したことを検知してアルゴリズムを終了する．よって、エージェントは g -部分集合を達成する．

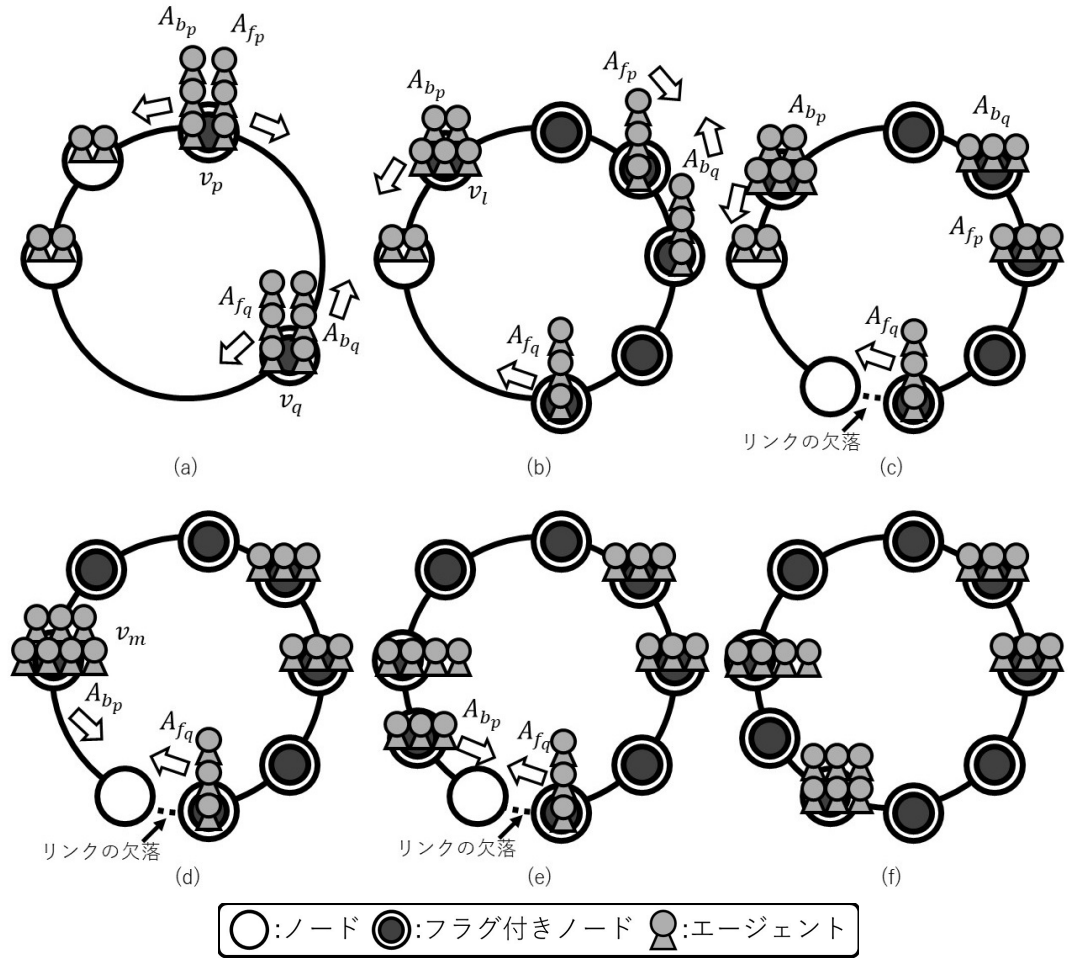


図 5 $k \geq 8g - 3$ の場合の達成フェーズの実行例

Algorithm 10 手順 $Moving2(a_i.dir)$ (v_j は a_i が現在滞在しているノード)

```
1: while  $a_i.rounds \leq n$  do
2:    $a_i.rounds \leftarrow a_i.rounds + 1$ 
3:   if  $(a_i.dir = 1) \wedge (v_j.fMarked = false)$  then
4:      $v_j.fMarked = true$ 
5:   end if
6:   if  $(a_i.dir = -1) \wedge (v_j.bMarked = false)$  then
7:      $v_j.bMarked = true$ 
8:   end if
9:    $v_j.nAgents \leftarrow v_j.nAgents - 1$ 
10:  ノード  $v_j$  から  $a_i.dir$ (1: 前進, -1: 後進) によって決定した方向のノードへ移動を試みる
11:   $v_j.nAgents \leftarrow v_j.nAgents + 1$ 
12:  if  $a_i$  が次のノードに移動できた then
13:    if  $((a_i.dir = 1) \wedge (v_j.bMarked = true)) \vee ((a_i.dir = -1) \wedge (v_j.fMarked = true))$  then
14:      アルゴリズムを終了
15:    end if
16:    if  $a_i.dir = 1$  then
17:       $v_j.fMarked = true$ 
18:    end if
19:    if  $a_i.dir = -1$  then
20:       $v_j.bMarked = true$ 
21:    end if
22:    if  $(v_j.bMarked = true) \wedge (v_j.fMarked = true)$  then
23:       $A_f$  と  $A_b$  のどちらもが訪問しているため, アルゴリズムを終了
24:    end if
25:     $a_i.dir \leftarrow v_j.dir$ 
26:    if  $v_j.waiting = true$  then
27:       $a_i.rank$  の計算を行う
28:      if  $(v_j.nAgents \geq 2g) \wedge (a_i.rank \geq g + 1)$  then
29:        アルゴリズムを終了
30:      end if
31:    end if
32:  end if
33: end while
```

達成フェーズにて、以下の補題 8 がある。

補題 8. 達成フェーズの実行後、エージェントは g -部分集合を達成する

Proof. 補題 7 により、達成フェーズの最初に $2g$ 体以上のエージェントを持つノードが少なくとも一つ存在する。従って、 $2g$ 体以上のエージェントは、手順 *More2()* (アルゴリズム 8) を実行し前進または後進を開始することができる。以降は、補題 3 と同様の議論によって、 g -部分集合を達成することができる。よって補題は成立する。□

提案アルゴリズムについて以下の定理 5 がある。

定理 5. $k \geq 8g - 3$ が成立するとき、提案アルゴリズムは動的リングにおける g -部分集合問題を総ラウンド数 $O(kn)$ 、総移動数 $O(gn)$ で解く。

Proof. まず、キラリティ統一フェーズの前半パートでは、二回の $3n$ ラウンド合計 $6n$ ラウンドの間移動することで、少なくとも一組のエージェントが交差あるいは同じノードに留まることを目指す。ここでのラウンド数は $O(n)$ 、この移動の際、 k 体のエージェントが ID を三つ観測するまで移動を続けるため、総移動数は $O(k)$ とみなすことができる。キラリティ統一フェーズの後半パートでは、前半パートによってできたエージェントの組がそれぞれ別方向に移動し自身のキラリティを伝えていく。この時、全てのエージェントの ID が $1 \sim ck (c \geq 1)$ の範囲であると仮定し、エージェント自身の組の ID が $(i-1) \times g < a_i.minID \leq i \times g (1 \leq i \leq k)$ の範囲であれば、それらのエージェントは、 $(i-1)n + 1$ から in ラウンド目までの間のみ移動を行う。よって全てのエージェントの組が移動を行うとすると、最大で $\lceil ckn/g \rceil$ ラウンドかかり、ラウンド数は $O(kn/g)$ とみなせる。そして、同じ n ラウンドの間に移動を行うエージェントの組は最大で g 組であり、エージェント数は $2g$ 体となる。移動を行った中で最小の ID を持つ組が一周した時点で全体のキラリティが統一されるため、それ以降のエージェントの組は移動を行わない。そのため、後半パートの総移動数は多くとも $2gn$ 回であるため、総移動数は $O(gn)$ とみなせる。従って、キラリティ統一フェーズ全体では、ラウンド数 $O(kn/g)$ 、総移動数 $O(gn)$ である。

次に、準選択フェーズでは、 $3n$ ラウンドの間、各エージェントは a_i は現在のノードに $2g$ 体以上のエージェントが存在するか、少なくとも $10g - 4$ 個の ID を収集し、集合候補ノードを決定するまで前進を試みる。これには、総ラウンド数 $O(n)$ が必要である。さらに、 a_i は $10g - 4$ 個の ID を観測すると移動を止めるので、このフェーズでは各リンクは最大でも $10g - 3$ 回通過する。したがって、準選択フェーズの移動複雑度は $O(gn)$ である。

次に、準集合フェーズでは、各エージェントは $3n$ ラウンドの間前進し、集合候補ノードか、エージェントが $2g$ 体以上存在するノードに到達し留まろうとする。これには、総ラウンド数 $O(n)$ が必要である。各リンクは最大で $4g - 1$ 回エージェントが通過するので、準集合フェーズの移動複雑度は $O(gn)$ である。

最後に、達成フェーズでは、 $2g$ 体以上のエージェントをもつノードに存在するエージェントは、 g -部分集合を達成するために移動を行う。補題 3, および補題 8 で述べたように前進グループと後進グループの少なくともいずれかは各ラウンドで次のリンクを訪問できるため、 n ラウンド以内に終了することができる。さらに各前進グループと後進グループは最大 $2g$ 体のエージェントから構成されるため、各ノードを通過するエージェント数は最大 $4g$ 体となる。したがって、総移動数は $O(gn)$ である。

従って、エージェントは総ラウンド数 $O(kn/g)$ と、総移動数 $O(gn)$ で g -部分集合問題を解くことができ、定理は成立する。□

6 結論

本論文では、動的リングにおけるエージェント間にキラリティの無い状況下の g -部分集合問題を考え、キラリティの有る場合のアルゴリズム [11] と比較して、総ラウンド数と総移動数を用いて評価を行った。第一に、 $2g + 1 \leq k \leq 3g - 2$ の時、ラウンド数 $O(n \log g)$ と総移動数 $O(gn \log g)$ で解けることを示した。第二に、 $3g - 1 \leq k \leq 8g - 4$ の時、ラウンド数 $O(n)$ と総移動数 $O(kn)$ で解けることを示した。最後に、 $k \geq 8g - 3$ の時、ラウンド数 $O(kn/g)$ と総移動数 $O(gn)$ で解けることを示した。今後の課題は、より一般的なトポロジーにおける問題や、半同期または非同期で行動するエージェントに対する問題についての検討、モバイルエージェントを用いた分散システムの現実的な実装用途について検討を行っていく。

参考文献

- [1] Robert S Gray, George Cybenko, David Kotz, Ronald A Peterson, and Daniela Rus. D’agents: Applications and performance of a mobile-agent system. *Software: Practice and Experience*, Vol. 32, No. 6, pp. 543–573, 2002.
- [2] Joachim Baumann, Fritz Hohl, Kurt Rothermel, and Markus Straßer. Mole—concepts of a mobile agent system. *world wide web*, Vol. 1, pp. 123–137, 1998.
- [3] Danny B Lange and Mitsuru Oshima. Seven good reasons for mobile agents. *Communications of the ACM*, Vol. 42, No. 3, pp. 88–89, 1999.
- [4] Giacomo Cabri, Letizia Leonardi, and Franco Zambonelli. Mobile agent coordination for distributed network management. *Journal of Network and Systems Management*, Vol. 9, pp. 435–456, 2001.
- [5] Evangelos Kranakis, Nicola Santoro, Cindy Sawchuk, and Danny Krizanc. Mobile agent rendezvous in a ring. In *23rd International Conference on Distributed Computing Systems, 2003. Proceedings.*, pp. 592–599. IEEE, 2003.
- [6] Paola Flocchini, Evangelos Kranakis, Danny Krizanc, Nicola Santoro, and Cindy Sawchuk. Multiple mobile agent rendezvous in a ring. In *LATIN 2004: Theoretical Informatics: 6th Latin American Symposium, Buenos Aires, Argentina, April 5-8, 2004. Proceedings 6*, pp. 599–608. Springer, 2004.
- [7] Fukuhito Ooshita, Shinji Kawai, Hirotugu Kakugawa, and Toshimitsu Masuzawa. Randomized gathering of mobile agents in anonymous unidirectional ring networks. *IEEE Transactions on Parallel and Distributed Systems*, Vol. 25, No. 5, pp. 1289–1296, 2013.
- [8] Masahiro Shibata, Shinji Kawai, Fukuhito Ooshita, Hirotugu Kakugawa, and Toshimitsu Masuzawa. Partial gathering of mobile agents in asynchronous unidirectional rings. *Theoretical Computer Science*, Vol. 617, pp. 1–11, 2016.
- [9] Masahiro Shibata, Norikazu Kawata, Yuichi Sudo, Fukuhito Ooshita, Hirotugu Kakugawa, and Toshimitsu Masuzawa. Move-optimal partial gathering of mobile agents without identifiers or global knowledge in asynchronous unidirectional rings. *Theoretical Computer Science*, Vol. 822, pp. 92–109, 2020.
- [10] Masahiro Shibata and Sébastien Tixeuil. Partial gathering of mobile robots from multiplicity-allowed configurations in rings. In *International Symposium on Stabilizing, Safety, and Security of Distributed Systems*, pp. 264–279. Springer, 2020.

- [11] Masahiro Shibata, Yuichi Sudo, Junya Nakamura, and Yonghwan Kim. Partial gathering of mobile agents in dynamic rings. In *Stabilization, Safety, and Security of Distributed Systems: 23rd International Symposium, SSS 2021, Virtual Event, November 17–20, 2021, Proceedings 23*, pp. 440–455. Springer, 2021.
- [12] Giuseppe Antonio Di Luna, Paola Flocchini, Linda Pagli, Giuseppe Prencipe, Nicola Santoro, and Giovanni Viglietta. Gathering in dynamic rings. *theoretical computer science*, Vol. 811, pp. 79–98, 2020.
- [13] G Di Luna, Stefan Dobrev, Paola Flocchini, and Nicola Santoro. Distributed exploration of dynamic rings. *Distributed Computing*, Vol. 33, pp. 41–67, 2020.
- [14] Shantanu Das, Giuseppe Antonio Di Luna, Daniele Mazzei, and Giuseppe Prencipe. Compacting oblivious agents on dynamic rings. *PeerJ Computer Science*, Vol. 7, p. e466, 2021.
- [15] Masahiro Shibata, Yuichi Sudo, Junya Nakamura, and Yonghwan Kim. Uniform deployment of mobile agents in dynamic rings. In *International Symposium on Stabilizing, Safety, and Security of Distributed Systems*, pp. 248–263. Springer, 2020.