複数のエージェントによるオンライン木探索アルゴリズム

八神貴裕 * 山内由紀子 † 来嶋秀治 † 山下雅史 †
* 九州大学大学院システム情報科学府
† 九州大学大学院システム情報科学研究院

1 序論

1.1 はじめに

グラフ内を逃げ回る侵入者を探索者が移動しながら発見するグラフ探索問題について考える．この問題は Parsons[3] が，洞窟ではぐれた友人を見つけ出すためには何人の探索者が必要で，それぞれがどのように行動すべきかという問題を洞窟をグラフとみなして考察したことから生じた問題である．この問題では，侵入者は常にグラフ内を動き回っている．そして，探索者は侵入者がどこにいるのか知ることができず，侵入者の移動速度も分からない．これらのことは，一度探索した場所に再び侵入者が訪れる可能性があることを意味する．

グラフ探索問題は，探索者を石と見なしてグラフ上の石置きゲームとしてモデル化される [2, 4]．このモデルではグラフの形を知っているプレイヤーがグラフの外から石を動かして探索する．このようにプレイヤーが探索開始時に既にグラフの形を知っている探索を**オフライン探索**と呼ぶ．移動する探索対象を発見する問題は，建物の警備ロボットや，火事，地震などの災害が起こった際に，建物に逃げ遅れた人がいないか探すロボットの開発に役立つと考えられる．しかし，このようなロボットは探索を行う建物の構造を探索開始前から必ず知っているとは限らない．そこで本稿では，グラフの形を知らない複数の**非同期エージェント**が実際にグラフ内に入りグラフを探索するモデルを考える．このように探索開始時にグラフの形が与えられていない探索を**オンライン探索**と呼ぶ．また，オンライングラフ探索のモデルとしては，一人のプレイヤーが探索者として実際にグラフの中に入り，侵入者の移動を制限するバリケードを用いて探索するバリケードモデルも考えられる [5]．石置きゲームのモデルは，グラフの外にいるプレイヤーの指示で石と見なされている探索者が自らグラフ内を移動していると考えられる．バリケードモデルでは，グラフの中にいる探索者が自身では移動できないバリケードを運ぶ点で異なる．バリケードモデルについては，任意の木について最小のバリケード数で探索可能なオンラインアルゴリズムが存在する [5]．

本稿では，次のようなモデルを扱う．各エージェントはメモリを持ち，それぞれ異なる頂点から探索を開始すると仮定する．本稿ではエージェントが個別の ID を持つ場合と，ID を持たない場合の二つを考える．これらのモデルを用いて，連結かつ閉路を持たないグラフである木を最小のエージェントで探索するオンラインアルゴリズムを提案する．アルゴリズムの流れは，まず全てのエージェントが同じ頂点に集合し，その後リーダーとなるエージェントを 1 人決める．そしてそのリーダーをバリケードモデルの探索者，それ以外のエージェントをバリケードと見なして探索を行う．本稿では特に，エージェントが 1 つの頂点に集合し，リーダー（探索者）を決定する部分について述べる．

1.2 関連研究

石置きゲームのモデルの一つに辺探索モデルが存在する [2]．辺探索モデルにおける石（探索者）をガー

ドと呼ぶことにする．G を辺探索モデルで探索するときの最小のガード数を $es(G)$ と表す．問題の入力はグラフ G で，$es(G)$ の計算と，ガードの操作の手順を考えることが目的である．ガードの操作は，ガードを頂点に置く，ガードを頂点から取り除く，ガードをある頂点から隣接する頂点へ辺を通過して移動するの 3 つである．

次に，バリケードモデルのオンライン探索について紹介する．このモデルは 1 人のエージェントがバリケードを使って探索をしていると考えることもできる．G をバリケードモデルで探索するときの最小のバリケード数を $r(G)$ と表す．探索者とバリケードはメモリを持つ．探索者は探索開始時グラフの形を知らないと仮定する．問題の入力はグラフ G とバリケードの個数 r，探索者の初期位置 $v \in V$ で，G は r 個のバリケードで探索可能か否か判断し，可能であれば探索を実行することが目的である．

探索者は，隣接する頂点に辺を通過して移動する，頂点にいるときバリケードの設置や回収をする，今いる頂点のバリケードのメモリの読み出しと書き込み，の 4 つの動作を組み合わせて探索を行う．任意の木 T について，探索開始時に持っているバリケード数が $r(T)$ 以上であれば必ず探索に成功するオンライン探索アルゴリズムが存在することが示されている [5]．また，バリケードモデルに関して，次の結果を既に得ている．

定理 1 ([5])**.** 任意の木 T について，$es(T) = r(T)+1$ が成立する．

1.3　主な結果

任意の木 T のオンライン探索について，エージェントが ID を持つ場合は，エージェント数は $es(T)$ と一致する．また，エージェントが ID を持たない場合，提案アルゴリズムではエージェント数は $es(T) + 1$ 必要になる場合がある．

2　準備

ここでは，問題やモデルの定義と既存の研究の紹介を行う．

2.1　グラフに関して

本稿では単純かつ連結な無向グラフを扱う．特に平面に埋め込まれた連結かつ閉路を持たない木 $T = (V, E)$ について議論する．

以下，本稿で扱うグラフの詳細な定義を行う．頂点 v の次数は $\deg(v)$ で表す．グラフの各頂点，各辺は固有のラベルを持たない．しかし，グラフ内の各頂点 $v \in V$ から接続する辺に対してラベル付け関数の集合 $\Lambda = \{\lambda_v | v \in V\}$ によって，反時計回りに 0 から $\deg(v) - 1$ までのポート番号が付けられている．ラベル付け関数 $\lambda_v(e)$ は頂点 v から見た接続する辺 e のポート番号を表す．また，エージェントは頂点や辺にメッセージを残すことはできない（頂点や辺にはメモリを与えない）．この問題では，辺は直線のように見なして，エージェントが辺の上にいるといった場合は，辺のどの位置にいるかを考えることにする．

2.2　グラフ探索問題

オフラインの**グラフ探索問題** (graph search problem) の入力はグラフ $G = (V, E)$ であり探索に必要なエージェント数と各エージェントの行動について考える．オンライン探索では，問題の入力はグラフ G とエージェント数 k とエージェントの初期位置 $S \subseteq V$ で，k 人のエージェントが異なる頂点からそれぞれ同じアルゴリズムを開始し，G を探索可能であるか否か判断し，可能であれば探索を実行することを目的とする．探索開始時刻のグラフの全ての辺は**非クリア** (contaminated) と呼ばれる状態である．非クリアな辺はエージェントの通過によって**クリア** (cleared) になる．しかし，クリアな辺と非クリアな辺の両方が接続する頂点 $v \in V$ があり，v にエージェントがいない場合，v に接続するクリアな辺は全て非クリ

アに戻る．このことを**再汚染** (recontaminated) という．グラフ G の全ての辺が同時にクリアになれば探索は終了する．直感的に言うと，非クリアな辺は侵入者がいる可能性があり，クリアな辺は侵入者がいない．

2.3 複数のエージェントのオンライン探索モデル

エージェントが個別の ID を持つ場合と ID を持たない場合の 2 つを考える．G を ID を持つ複数のエージェントがオンライン探索できる最小のガード数を $as_o(G)$ と表す．G を ID を持たない複数のエージェントがオンライン探索できる最小のガード数を $as_a(G)$ と表す．また，エージェントはそれぞれメモリを持ち，非同期で行動する．全てのエージェントは異なる頂点から同じアルゴリズムを開始すると仮定する．また，エージェントは探索開始時，G の形や頂点数，G 内のエージェントの総数を知らない．

各エージェントは探索中次のことを知ることができる．

- 今いる頂点にどの辺を通過して到達したのか
- 同じ頂点いる他のエージェント数
- 他のエージェントと辺ですれ違ったことを知ることができる．ここでは辺上の同一点上にいることを**すれ違う**と言うことにする．

各エージェントは次の動作ができる．

- 頂点 u から隣接する頂点 v に辺 (u, v) を通過して移動することができる．辺を移動するエージェントは常に同じ速度で移動する（辺の途中で立ち止まることはない）．
- 同じ頂点上のエージェント及び辺ですれ違ったエージェントと情報の受け渡しができる．

エージェントの動作に関して，Baba ら [1] は同じ辺に他のエージェントがいたとしてもそのことを知ることはできず，エージェント同士の情報の交換もできないと仮定されていた．しかし，グラフ探索問題では洞窟や建物をグラフと見なしているので，頂点や辺でエージェントの能力を分けることはしない．そのため，このモデルでは辺ですれ違ったエージェントを知ることができるのは自然な仮定であると考える．また，ID の比較を行うためにエージェント同士は情報の交換ができる必要がある．

非クリアな辺 $e = (u, v)$ は以下の 2 つの方法でクリアになる．一般性を失わず，エージェントは頂点 u から頂点 v に移動すると仮定する．

- 頂点 u に 2 人以上のエージェントがいるとき，そのうちの 1 人のエージェントが辺 e を通過して移動することで e はクリアになる．
- 頂点 u に接続する e 以外の全ての辺がクリアであれば，u にいるエージェントが 1 人だけでも辺 e を通過して移動することで e はクリアになる．

2.4 集合問題

本稿では木における複数のエージェントの**集合問題** (gathering problem) を以下のように定義する．

定義 1. 木 $T = (V, E)$ 内のエージェントは k $(k \geq 2)$ 人とする．木 T 内の k 人のエージェントがある 1 つの頂点 $g \in V$ に集まり，全てのエージェントが g にいることを確認することを集合問題と呼ぶ．またこのような頂点 g を集合点と呼ぶ．

集合問題を解く上で，木の**対称性** (symmetry) が重要であることが知られている．

定義 2 ([1])**.** ラベル Λ を持つ木 T が対称性を持つことは以下の 3 つの条件を満たす関数 $g : V \to V$ が存在することと同値である．

1. 任意の $v \in V$ について，$v \neq g(v)$ が成立する．

2. 任意の $u, v \in V$ について，u が v に隣接するとき，かつそのときに限り，$g(u)$ は $g(v)$ に隣接する．

3. 任意の辺 (u,v) について，$\lambda_u((u,v))$ は $\lambda_{g(u)}((g(u),g(v)))$ と等しい．

図 1 は対称性を持たない木と対称性を持つ木の例である [1].

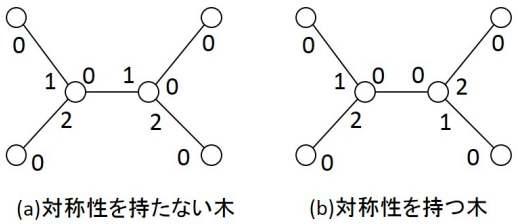

図 1: 対称性を持たない木と対称性を持つ木 [1] Fig 1. を基に作成

Baba ら [1] は，ID を持たず，互いに情報交換のできない複数のエージェントを集める問題について議論している．主な結果は，ラベル Λ を持つ任意の木 T が中心を 1 つだけ持つとき，または，対称性を持たないとき，エージェントは 1 つの頂点に集まることができ，中心を 2 つ持ち，かつ対称性を持つとき，エージェントは隣り合う 2 つの頂点に集まることができるアルゴリズム：Gathering が示されている．ここでは，そのアルゴリズムの概要を説明する．

Gathering ではエージェントは，基本手順 (basic step) と呼ばれる手順でグラフ内を移動する．この基本手順は深さ優先探索をもとにした手順である．エージェントが頂点 v に到達したときに通過したポートを i とする．エージェントは次に進むポートを $(i+1) \bmod \deg(v)$ によって定める．また，本稿ではある頂点 v から基本手順でグラフの全ての頂点，辺を訪れて再び v に戻ってくることを**走査**と呼ぶことにする．

Gathering は 3 つのフェーズに分かれている．フェーズ 1 では，エージェントは葉に到達するまで基本手順で移動する．到達した葉を s とする．フェーズ 2 では，s から基本手順を用いて木全体を走査する．このとき，木 T の形とラベル Λ の情報を集める (詳細は [1])．最後にフェーズ 3 では，フェーズ 2 で集めた情報をもとにグラフの中心を計算する．中心が 1 つの場合は，全てのエージェントはその頂点に集まる．中心が 2 つの場合でも，Λ を持つ木 T が対称性を持たないなら，2 つの中心は区別できるので，エージェントは 1 点に集まることができる．しかし，中心が 2 つかつ，Λ を持つ木 T か対称性を持つなら，2 つの中心は区別できず，各エージェントは 2 つの中心のどちらかに移動するため 1 つの頂点に集合できない場合がある．本稿では ID を持つエージェントの集合問題に関しては，エージェント同士の情報の受け渡しや ID の比較を行うことで 1 つの頂点に集合する．

3 ID を持つエージェントのオンライン木探索アルゴリズム

今回提案する探索アルゴリズムの概略は，まずラベル Λ を持つ木 T 内のエージェントが 1 つの頂点に集合し，エージェント数を確認する．その後，バリケードモデルの探索者となるエージェントを 1 人選び，残りのエージェントをバリケードとして T の探索を行う．本稿では，T 上の k $(k \geq 2)$ 人のエージェントが T の 1 点で集合する方法について特に具体的に説明する．集合に関する部分は Baba ら [1] の Gathering アルゴリズムを基にする．Gathering は木の 1 つまたは，隣り合う 2 つの頂点に集合するアルゴリズムであったが，ここでは全てのエージェントの ID と互いの情報交換を用いることで 1 つの頂点に集まり，全てのエージェントが集合できていることを確認するまでを行う．

探索アルゴリズムは以下の 3 つのステップに分かれる．

(移動ステップ) 各エージェントは Gathering のフェーズ 1 とフェーズ 2 を行い木 T とラベル Λ の情報を集める．そして集合点 g を計算しそれぞれ移動する．

(確認ステップ)g に 2 人以上のエージェントが集まると ID が最小のエージェント a_{min} を選び，まだ集合できていないエージェントがいるか確認するために木 T を基本手順を用いて一度だけ走査する．全てのエージェントが g に集合するまで確認ステップの先頭に戻り，a_{min} の選択と T の走査を繰り返す．

(探索ステップ)g に集合したエージェントからリーダーを 1 人選び，バリケードモデルの探索者，その他のエージェントをバリケードとして T を探索する．

3 つのステップのうち，エージェントの集合は移動ステップと確認ステップで行う．移動ステップと確認ステップにおいて，ほとんどの移動は基本手順によって行う．

このアルゴリズムではエージェントはメモリ $state$ の値に従って行動する．まず，$state$ の取りうる値とそれぞれの意味を説明する．

Explore

アルゴリズム開始時の値．この状態のエージェントは全て集合点 g の位置を知らない．

Select

ラベル Λ を持つ木 T が中心を 2 つ持ちかつ対称性を持つ場合，1 人のエージェントの計算だけでは g を決定できない．その場合，$state = \text{Select}$ に変更し g を決定する．

Stop

g まで到達し停止している状態.

Check

g に集合したエージェントの中で，ID が最小のエージェントがまだ集合できていないか確認しているときの状態.

Wait

g のエージェントのうち，$state = \text{Check}$ でないエージェントはこの状態になり，$state = \text{Check}$ のエージェントが g に帰ってくるのを待つ.

$state = \text{Explore}$ の場合，エージェントは集合点 g を知らない．$state = \text{Select}$ の場合は，木 T，ラベル Λ の情報全て持っているが，g の位置を知っているエージェントと知らないエージェントがいる．それ以外の場合は g の位置と木 T，ラベル Λ の情報を全て持っている．$state = \text{Explore}, \text{Select}$ のエージェントは $state = \text{Stop}, \text{Check}, \text{Wait}$ のエージェントと同じ頂点にいるとき，またはすれ違う際に，g や木 T の形の情報を受け取ることで，一部の手順を省略することができる．

以下，移動ステップと確認ステップの詳細を説明する．

3.1 移動ステップ

初期状態で各エージェントの $state$ は Explore である．エージェントは Gathering のフェーズ 1，フェーズ 2 を行い，木 T の形とラベル Λ を求める．そして，ラベル Λ を持つ木 T が中心を 1 つしか持たない場合，および，中心が 2 つかつ対称性を持たない場合は，Gathering のフェーズ 3 によって集合点 g を唯一に決定することができる．エージェントは g へ移動し，g に $state = \text{Wait}$ のエージェントがいれば自身の $state$ も Wait に変更する．それ以外の場合は $state$ を Stop に変更する．そして確認ステップに進む.

中心が 2 つかつ対称性を持つ場合は，フェーズ 1 で初めて到達した葉 s からの距離が大きい方の中心 c' へ基本手順で移動する．c' へ移動するまでに他のエージェントから集合点 g の位置を受け取れなかった場合，$state$ を Select に変更し，2 つの中心とその間の辺を往復し続ける．今は $k \geq 2$ の場合を考えているので，少なくとも 2 人のエージェントが 2 つの中心とその両方を端点に持つ辺に集まる．$state = \text{Select}$ かつ，g の情報を持たないエージェント同士がすれ違ったとき（もしくは同じ頂点にいるとき），ID を比較し ID の最も大きいエージェントが進もうとしている（もしくは今いる）頂点を集合点 g に決定する．ただし，$state$ を Select に変更した後一度も移動していないエージェントだけで g は決定しない（図 2）．また，辺上で同じ方向に移動しているエージェントのみで g を決定することはしない（図 3）．これらの場合に g を決定すると，異なる g を持つエージェントが存在する可能性があるからである．例えば，図 2 はエージェント a_1 と a_2 が同時に中心の一方 c_a に到達し，それと同時に a_3 と a_4 がもう一方の

中心 c_b に到達し，それぞれのエージェントが $state$ を Select に変更した直後を表している．a_1, a_2, a_3, a_4 は g を知らないと仮定する．このとき，c_a と c_b には g を知らないエージェントが 2 人ずついるが，このまま g を決定すると，a_1 と a_2 は c_a，a_3 と a_4 は c_b を g に選択してしまう．

図 2: g を決定しない場合の例 1

次に図 3 は図 2 の状況から，a_1 と a_2 が同時に c_a から移動を開始し，a_3 と a_4 も同時に c_b から移動を開始した状況である．このとき，a_1 と a_2，a_3 と a_4 は辺上の同じ点にいる（情報の受け渡しが可能）が，ここで g を決定してしまうと，a_1 と a_2 は c_b を，a_3 と a_4 は c_a を g に選択してしまう．

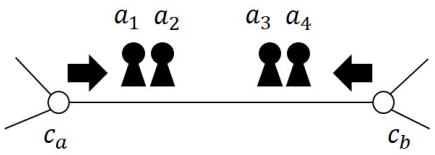

図 3: g を決定しない場合の例 2

g を決定した後は，g へ移動し，g に $state = \text{Wait}$ のエージェントがいれば自身の $state$ も Wait に変更する．それ以外の場合は $state$ を Stop に変更する．そして確認ステップに進む．

もし，g を知らないエージェントが g を知っているエージェントと同じ頂点にいたり，すれ違ったりした場合は，g や木 T，ラベル Λ の情報を受け取り，g へ基本手順で移動することで移動ステップの一部を省略する．

3.2 確認ステップ

g には移動ステップで必ず 2 人以上のエージェントが集まる．$state = \text{Stop}$ であるエージェントが 2 人以上になると，ID が最小のエージェント a_{min} を選ぶ．そして，a_{min} の $state$ を Check，それ以外の頂点 g のエージェントの $state$ は Wait とする．

a_{min} は 1 度だけ T 全体を基本手順で走査する．このとき，すれ違ったエージェントの有無を記録しておく．T の走査を終えた後，すれ違ったエージェントがいれば確認ステップの先頭に戻り，最小の ID を持つエージェントを選んで，再び木全体を走査する．もしいなければ全てのエージェントは g に集まっている．

確認ステップの正当性に関して，次の補題が成立する．

補題 1. a_{min} が基本手順で T 全体を走査したとき，頂点 g 以外ですれ違うエージェントがいなければ，全てのエージェントは g に集合している．

3.3 探索ステップ

確認ステップで，全てのエージェントは木の 1 つの頂点に集合している．集合したエージェントの中から，ID が最小のエージェントを選び，そのエージェントをバリケードモデルの探索者，残りのエージェントをバリケードとみなして，バリケードモデルのオンライン探索アルゴリズムを実行する．このアルゴリズムより，以下の補題が導かれる．

補題 2. 任意の $as_o(T) \geq 2$ である木 T について，$as_o(T) \leq r(T) + 1$ が成立する．

エージェント数が 1 のときについても考えなければならない．$k = 1$ のとき，このアルゴリズムでは T 内のエージェントは 1 人なのか，それ以上存在しているのか知ることができない．エージェント 1 人で探索できる木（つまり道グラフ）の場合は，木の形を求めた時点で探索が成功していることが分かる．よって，与えられた木 T が道グラフであった場合は，

木の形を求めた時点で探索成功とし，アルゴリズムを停止すればよい．しかし，T がエージェントが 1 人では探索できない木であった場合，木内に存在する 1 人のエージェントは集合アルゴリズムを終了することができない．

定理 2. ラベル Λ を持つ任意の木 T について，$as_o(T) = es(T)$ である．

4 ID を持たないエージェントのオンライン木探索アルゴリズム

基本的なアルゴリズムの方針は ID を持つ場合と似ている．アルゴリズムの流れは，まずはエージェントが 1 つの頂点に集合し，その後リーダーを決めてバリケードモデルのアルゴリズムで探索する．ここで問題になるのはリーダーの決め方である．エージェントが ID を持っているときは ID の大小関係でリーダーを決めることができたが，エージェントが ID を持たない場合この方法は使えない．ここでは，各エージェントは異なる頂点からアルゴリズムを開始するという仮定より，アルゴリズム開始時からエージェントが集合点 g に集まるまでに通過したポート番号の列 P を記録し，それを ID のように用いることにする．より正確には，頂点 v に到達したとき，どのポート番号の辺を通過してきたかを記録する．P の比較でエージェントを区別したり，P を辞書式順序で並べたときに先頭に来るエージェントをリーダーとしたりすることができる．

しかし，次の条件 C1 を満たす場合は，この提案アルゴリズムではエージェントは 1 点に集合できない場合がある．この条件は定義 2 の条件を拡張したものである．

（条件 C1）以下の 4 つを満たす関数 $g : V \to V$ が存在する．

1. 任意の $v \in V$ について，$v \neq g(v)$ が成立する．

2. 任意の $u, v \in V$ について，u が v に隣接するとき，かつそのときに限り，$g(u)$ は $g(v)$ に隣接する．

3. 任意の辺 (u, v) について，$\lambda_u((u, v))$ は $\lambda_{g(u)}((g(u), g(v)))$ と等しい．

4. 任意の $v \in V$ について，初期状態で v にエージェントがいるとき，かつそのときに限り，$g(v)$ にもエージェントがいる．

条件 C1 を満たす場合，辺上で P の比較を行ったとき異なるエージェント同士の P が同じ場合がある．そのため，P の比較を行ったエージェント間の大小関係が定まらない場合がある．

ID なしの複数のエージェントによる探索アルゴリズムも移動ステップ，確認ステップ，探索ステップの 3 つに分けられる．各ステップの詳細を説明する．

4.1 移動ステップ

基本的には，エージェントが ID を持つ場合と同じである．ただし，エージェントが ID の比較を行っていた部分に関しては，通過したポートの列 P を比較する動作に置き換える．具体的には，ID が最小もしくは最大のエージェントを選ぶ際は，P の辞書式順序で先頭に来るエージェントを選択する．

注意すべき点は，エージェントが P の比較を行った際，辞書式順序で先頭に来るエージェントが 1 人に定まらない場合があることである（図 4）．最終的に全てのエージェントが 1 つの頂点に集合できるか否かは確認ステップの最後で判断することにし，移動ステップでは，エージェントが 2 つの頂点に集まっていてもよいことにする．

図 4(a) は条件 C1 を満たす木とエージェントの初期配置の例である．図 4(b) は全てのエージェントが同時に $state = \text{Select}$ になった直後の状態を表す．その後，全てのエージェントが同時にもう一方の中心に移動しようとすると，2 つの中心を端点に持つ辺で 4 人のエージェントが同時にすれ違う (図 4(c))．このとき，4 人のエージェントの P を辞書式順序で並べると先頭 2 つとその後ろの 2 つの P はそれぞれ同じ値であるので，この 4 人では集合点 g を決定できない．それぞれが頂点に到達すると，c_a と c_b はどちらとも $state = \text{Select}$ かつ，$state = \text{Select}$ に変

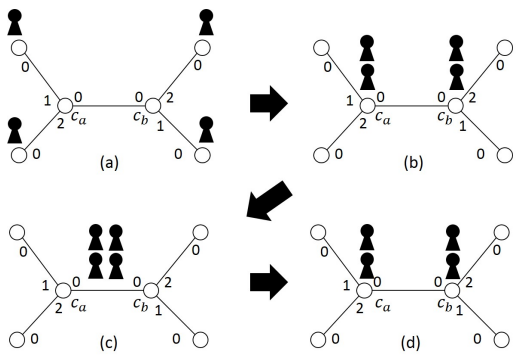

図 4: 1 つの頂点に集合できない例

更されて少なくとも一度は移動しているエージェントが 2 人いるので，それぞれのエージェントは今いる頂点を集合点 g に決定してしまう．

4.2 確認ステップ

確認ステップでは (i) 中心が 1 つまたは，中心が 2 つかつ対称性を持たない場合と，(ii) 中心が 2 つかつ対称性を持つ場合で内容が変わる．確認ステップに入ったエージェントは既に木 T とラベル Λ を知っているので，(i) と (ii) は区別できる．まず (i) の場合について考える．(i) の場合，全てのエージェントは各自の計算で同じ集合点 g を計算できる．エージェントが ID を持つ場合の確認ステップと同様の手順で全てのエージェントが g に集合できているか確認する．

(ii) の場合，移動ステップで求めた集合点（これは中心の一方）を g，もう一方の中心を g' とする．エージェントが ID を持つ場合の確認ステップと同様に，a_{min} を決定し，T を走査する．a_{min} が再び g に戻ってくるまでに g と g' 以外で他のエージェントと同じ頂点にいたり，すれ違ったりしなければ，T 内のエージェントは全て g と g' に集まっている．そうでないなら，a_{min} を選び，T の走査を繰り返す．また，確認ステップで T を走査したとき，g' にエージェントがいるか否か分かる．g' にエージェントがいないのであれば，探索ステップへ進む．もし，g' にエージェントがいるなら，最後に再び a_{min} を選ぶ．選ばれたエージェントは g' に移動し，g に集合しているエージェント数と g の全てのエージェントの P を辞書式順序で並べ，先頭から順につなげた列 $\mathcal{P}$ を伝える．エージェント l 人の P からなる $\mathcal{P}$ は $\mathcal{P}$ =<（1 人目の P）;（2 人目の P）; ... ;（l 人目の P）> のように構成する．

ここまでは g' に集まったエージェントも同じ手順を行っている．g と g' のそれぞれに集まっているエージェント数が分かると，集まったエージェント数が少ない方の頂点にいるエージェントはもう一方の中心に移動する．しかし，エージェント数 k が偶数のとき，g と g' で探索者数が等しい場合がある．このときは互いの $\mathcal{P}$ を比較し，辞書式順序で先頭に来る方が，もう一方の中心に移動する．エージェントが 1 点に集まると探索ステップに進む．

ただし，アルゴリズム開始時に次の条件 C1 を満たすときは，g で求めた $\mathcal{P}$ と g' から受け取った $\mathcal{P}$ が一致する場合があり，そのときは集合点を 1 つに決定できない．例えば，図 4(d) の後，c_a と c_b に集まったエージェントは，T を一度だけ走査し他のエージェントがいないことを確認すると，c_a と c_b のエージェントは互いにエージェント数と $\mathcal{P}$ を伝えるが，エージェント数はどちらも 2 人で，$\mathcal{P}$ も同じ値であることから，1 点に集合することができない．この場合は $k/2$ 人のエージェントのグループ 2 つとして，探索ステップに進む．

また，特に $k = 2$ かつ条件 C1 を満たす場合，2 人のエージェントは $state$ = Select のままで確認ステップを終了できない．

4.3 探索ステップ

集合したエージェントの中から，P が辞書式順序で先頭に来るエージェントを選ぶ．選ばれたエージェントを探索者，残りの人のエージェントをバリケードと見なして，木 T の探索を行う．

このアルゴリズムより次の定理が導かれる．

定理 3. ラベル Λ を持つ任意の木 T について，$as_a(T) \leq es(T) + 1$ が成立する．

5 まとめ

複数のエージェントによるオンラインの木探索アルゴリズムを提案した．任意の木 T について，エージェントが ID を持つ場合は，エージェント数は $es(T)$ で十分であることを示した．一方，エージェントが ID を持たない場合，エージェント数は $es(T) + 1$ で十分であることを示した．今後の課題は，ID を持たないエージェントのオンライン探索について，任意の木 T を $es(T)$ 人のエージェントで探索することができるか否か考察する．また，複数のエージェントによる一般のグラフのオンライン探索アルゴリズムを考える．

参考文献

[1] D. Baba, T. Izumi, F. Ooshita, H. Kakugawa, and, T. Masuzawa, "Linear time and space gathering of anonymous mobile agents in asynchronous trees.", Theoretical Computer Science 478, pp. 118–126, 2013.

[2] N. Megiddo, S. L. Hakimi, M. R. Garey, D. S. Johnson, and C. H. Papadimitriou, "The complexity of searching a graph", Journal of the ACM 35.1, pp. 18–44, 1988.

[3] T. D. Parsons, "Pursuit-evasion in a graph", In *Proceedings of the Theory and Applications of Graphs*, pp. 426–441, 1976.

[4] 山下雅史, "捜索—移動する対象を探索する", 室田一雄編, 離散構造とアルゴリズム III, 近代科学社, pp. 115–162, 1994.

[5] 八神貴裕, 山内由紀子, 来嶋秀治, 山下雅史, "バリケードを用いたオンライン木探索について", 一般社団法人情報処理学会九州支部 火の国情報シンポジウム 2015 論文集, 4A-1, 2015.