Vol.2015-AL-155 No.16
2015/11/21

# 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，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，という 3 つの点から一度探索した場所に再び侵入者が訪れる可能性がある．

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, 5]．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手順を考える．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，グラフの形を知らない複数の探索者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．本稿では連結で閉路を持たないグラフである木について，木の形を知らない複数のエージェント（探索者）によるオンライン木探索を考える．

### 1.2 関連研究

石置きゲームによるグラフ探索問題のモデル化の 1 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．$G$ を辺探索モデルで探索するとき必要となる最小のガード数を $es(G)$ と表す．問題の入力はグラフ $G$ である．問題の目的は $es(G)$ と，探索手順を与えることである．許されるガード操作は，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，オンライン探索モデルの 1 つであるバリケードモデルを紹介する．このモデルでは 1 人のエージェントがバリケードを使って探索する．ここで，バリケードがある頂点に置かれていると，侵入者はその頂点を通過できない．$G$ をバリケードモデルで探索するときに必要となる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ち，探索者は探索開始時には $G$ を知らない．問題の入力はグラフ $G$ とバリケード数 $r$，探索者の初期位置 $v \in V$ である．探索者は探索開始時 $r$ 個のバリケードを持っている．探索者は $G$ 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完了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でき，頂点ではバリケードの設置や回収が実行できる．また，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を行える．バリケードモデルに関して，以下の定理が成立する．

**定理 1** ([6])**.** 任意の木 $T$ について，$es(T) = r(T) + 1$ が成立する．

また，オンライン木探索アルゴリズム TSB は，任意の木 $T$ を $r(T) = es(T) - 1$ 個のバリケードを用いてオンライン探索する．

**定理 2** ([6])**.** $r$ 個のバリケードを用いる TSB で木 $T$ の探索に失敗するならば，$r$ 個のバリケードで $T$ をオフライン探索するアルゴリズムは存在しない．

### 1.3 主結果

本稿ではメモリを持ち，互いに情報交換できる複数のエージェントによるオンライン探索を考える．バリケードは使用しない．エージェントが ID を持つ場合と持たない場合のそれぞれを考える．エージェントが ID を持つ場合は，任意の木をエージェント数 $es(T)$ で探索するアルゴリズム OATS を提案する．また，エージェントが ID を持たない場合，任意の木について最大エージェント数 $es(T)+1$ で探索するアルゴリズム AATS を提案する．AATS でエージェントが $es(T)+1$ 人必要な木 $T$ では，どのオンラインアルゴリズムでも $es(T)+1$ 人のエージェントが必要となる．すなわち，OATS と AATS は共にエージェント数に関して最適である．

## 2 準備

### 2.1 グラフ

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．$V$ は木 $T$ の頂点集合，$E$ は $T$ の辺集合である．頂点 $v$ の次数を $\deg(v)$ 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 $v \in V$ に接続する辺には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．以降，ラベル付け関数の集合 $\Lambda$ をラベルと呼ぶ．頂点 $v \in V$ から任意の頂点までの距離の最大値を離心数と呼ぶ．$V$ の頂点で離心数が最小の頂点を中心と呼ぶ．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がない）．このモデルでは，エージェントが辺を通過するには時間がかかり，2 人のエージェントが辺の中で出会った場合には，そのことが認識できると仮定する．

### 2.2 グラフ探索問題

オンライングラフ探索問題では，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 $S \subseteq V$ $(|S|=k)$ 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 $\mathcal{A}$ を開始する．各エージェントは $G$ 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．

侵入者の時刻 $t \in [0,\infty)$ での位置を，連続関数 $i(t)$ で表す．$k$ 人のエージェントを $a_0, a_1, ..., a_{k-1}$ とし，それぞれのエージェントの時刻 $t$ での位置を連続関数 $a_0(t), a_1(t), ..., a_{k-1}(t)$ で表す．関数 $a_j$ $(0 \leq j \leq k-1)$ は各エージェントが実行するアルゴリズム $\mathcal{A}$ によって決まる．関数 $i(t)$ と $a_j(t)$ $(0 \leq j \leq k-1)$ は平面に埋め込まれたグラフ $G$ の頂点または辺上の 1 点の座標を表す．グラフ $G$ を $k$ 人の探索者で探索可能であるとは，任意の $i(t)$ について，あるエージェント $a_j$ $(0 \leq j \leq k-1)$ が存在し，ある時刻 $\tau$ で $a_j(\tau)=i(\tau)$ となるような非同期エージェントの探索アルゴリズム $\mathcal{A}$ が存在することである．言い換えると，侵入者がグラフ $G$ をどのように移動したとしても，ある時刻 $\tau$ で必ずエージェントが侵入者を捕まえることができることを意味する．

グラフ探索問題では，グラフ $G$ 上の各点は**クリア**または**非クリア**のどちらかの状態を取る．$G$ のある点 $x$ が時刻 $t_0$ でクリアであるとは，$i(t_0)=x$ となる任意の関数 $i(t)$ について，あるエージェント $a_j$ $(0 \leq j \leq k-1)$ が存在し，ある時刻 $t \in [0,t_0]$ で $a_j(t)=i(t)$ であることである．非クリアな点はクリアでない点のことである．直感的に言うと，時刻 $t$ でクリアな点は，侵入者がエージェントと出会うことなく時刻 $t$ にたどり着くことはできない点で，時刻 $t$ で非クリアな点は，侵入者がエージェントと出会うことなく時刻 $t$ たどり着ける点である．グラフ $G$ は全ての点が非クリアな状態で与えられる．非クリアな点をクリアにする具体的な方法は次の 3 つである．

- 頂点 $v$ に $x$ $(x \geq 2)$ 人以上のエージェントがいて，そのうち $y$ $(1 \leq y < x)$ 人のエージェントが同時に移動する場合：$v$ から移動するあるエージェント $a$ が通過する辺を $e$ とする．辺 $e$ の端点 $v$ と $e$ 内のエージェント $a$ の間の点はクリアである．
- 頂点 $v$ に $x$ $(x \geq 1)$ 人以上のエージェントがいて，その $x$ 人のエージェント全員が同時に移動する場合：$v$ から移動するあるエージェント $a$ が通過する辺を $e$ とする．$v$ から移動する $x$ 人のどのエージェントも通過しない任意の辺 $e'$ が以下の条件 (i) または (ii) のどちらかを満たせば，辺 $e$ の端点 $v$ と $e$ 内のエージェント $a$ の間の点はクリアである．
  (i) $e'$ はクリアな辺である

(ii) $e'$ 内にはあるエージェント $a'$ がいて，$e'$ の端点 $v$ からエージェント $a'$ までの範囲がクリアである

- 辺 $e = (u, v)$ 内を $u$ から $v$ に移動するエージェントを $a$，$v$ から $u$ に進むエージェントを $a'$ とする．辺 $e$ でエージェント $a$ と $a'$ がすれ違った後，辺 $e$ のエージェント $a$ と $a'$ の間の範囲はクリアである．

特に，辺 $e$ の全ての点がクリアな場合，$e$ をクリアな辺，辺 $e$ の全ての点が非クリアな場合，$e$ を非クリアな辺と呼ぶ．グラフ $G$ のクリアな点が非クリアになる場合があり，それを再汚染と呼ぶ．ある非クリアな点からあるクリアな点までエージェントがいる点を通過せずに到達できる経路が存在する場合，その経路上の点は全て再汚染され非クリアになる．直感的に，侵入者がエージェントと出会うことなく移動できる範囲は全て非クリアになる．グラフ $G$ の全ての辺が同時にクリアな辺になれば探索は終了する．

### 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 を持たない場合のそれぞれを考える．$G$ 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．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．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．

- エージェントが辺 $e$ を通過して $v$ に到達したときの $\lambda_v(e)$．つまり通過してきた辺のポート番号．
- 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．辺上の同一点上にいることを**すれ違う**と言う．このとき，すれ違ったエージェントの移動方向も分かる．

各エージェントは，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の情報交換ができる．

### 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**を以下で定義する．

**定義 1.** 木 $T = (V, E)$ 内の全てのエージェントがある 1 つの頂点 $g \in V$ に集まる問題を集合問題と呼ぶ．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**が重要であることが知られている．[*1]

**定義 2** ([1])**.** ラベル $\Lambda$ 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 $f : V \to V$ が存在することである．

1. 任意の $v \in V$ について，$v \neq f(v)$ 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，$f(u)$ は $f(v)$ に隣接する．
3. 任意の辺 $(u, v)$ について，$\lambda_u((u, v))$ は $\lambda_{f(u)}((f(u), f(v)))$ と等しい．

図 1 に対称性を持たない木と対称性を持つ木の例を示す [1].

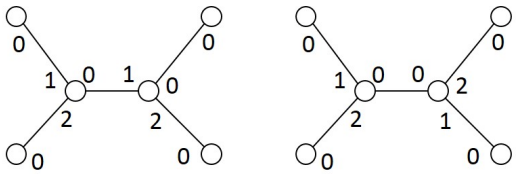


図 1: 対称性を持たない木と対称性を持つ木

Baba らは，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が集合する問題について議論している [1]．ラベル $\Lambda$ を持つ任意の木 $T$ が中心を 1 つだけ持つとき，または，中心を 2 つ持ちかつ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る．$T$ が中心を 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．そのアルゴリズム Gathering の概要を説明する．

Gathering ではエージェントは，**基本手順**と呼ばれる手順でグラフ内を移動する．この基本手順は木

[*1] 木 $T$ が対称性を持つための必要十分条件は，[4] の用語を用いれば，対称度 (symmetricity)$\sigma(T) = 2$ となることである．

の深さ優先探索を基にした手順である．エージェントが辺 $e$ を通過して頂点 $v$ に到達したときに通過したポートを $i = \lambda_v(e)$ とする．エージェントは次に $\lambda_v(e') = (i+1) \mod \deg(v)$ を満たす辺 $e'$ に進む．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．

Gathering は 3 つのフェーズから成る．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 $\Lambda$ の情報を集める．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，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，中心の一方に全てのエージェントが集まることができる．しかし，$T$ が中心を 2 つ持ちかつ対称性を持つなら，2 つの中心は区別できない．各エージェントは 2 つの中心のどちらかを選び移動するため，全てのエージェントが 1 つの頂点に集まることが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，2 つの中心を結ぶ辺上でのエージェント同士の情報交換や ID の比較を行うことで全てのエージェントが 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 OATS

ID を持つエージェントに対するオンライン木探索アルゴリズムを OATS(onymous agents tree search) とする．アルゴリズムの概略を示す：まずラベル $\Lambda$ 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 つの頂点に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．

OATS アルゴリズムは 3 つのステップから成る．
（移動ステップ）各エージェントは木 $T$ とラベル $\Lambda$ の情報を集める．$T$ が中心を 1 つ持つ，または，中心を 2 つ持ちかつ対称性を持たない場合は集合点 $g$ を計算しそれぞれ移動する．しかし，木が中心を 2 つ持ち，かつ対称性を持つ場合は中心の一方を仮の集合点 $p$ に決定し移動する．
（確認ステップ）(i)$T$ が中心を 1 つ持つまたは，中心を 2 つ持ちかつ対称性を持たない場合，$g$ にいるエージェントの中から ID が最小のエージェント $a_{min}$ を選ぶ．$a_{min}$ は木 $T$ を基本手順を用いて一度だけ走査し，まだ集合できていないエージェントがいるか確認する．全てのエージェントが $g$ に集合するまで確認ステップを繰り返す．(ii)$T$ が中心を 2 つ持ちかつ対称性を持つ場合，$p$ ではない中心を $p'$ とする．$p'$ を仮の集合点に選んでいるエージェントがいる場合がある．(i) と同様の手順で全てのエージェントが頂点 $p$ または $p'$ にいることを確認する．そして，ID が最小のエージェントがいる方の中心を集合点 $g$ に決定する．
（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 $\Lambda$ を持つ木 $T$ が中心を 2 つ持ちかつ対称性を持つ場合のみこの値を取る．他の *state* = Select のエージェントと共に仮の集合点 $p$ を決定する．

Stop
: 集合点 $g$（または仮の集合点 $p$）まで到達し停止している状態

Check

集合点 $g$（または 2 つの中心）以外の頂点や辺にエージェントがいないか確認しているときの状態

Wait

集合点 $g$（または仮の集合点 $p$）にいるエージェントのうち，ID が最小のエージェント $a_{min}$ に選ばれなかったエージェントはこの状態になり，$a_{min}$ が木 $T$ を走査している間待機する

$state$ = Explore の場合，エージェントは集合点 $g$ や仮の集合点 $p$ を知らない．$state$ = Select の場合は，木 $T$，ラベル $\Lambda$ の情報を全て持っているが，$g$ や $p$ 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 $g$ または $p$ の位置と木 $T$，ラベル $\Lambda$ の情報を全て持っている．

以下，各ステップの詳細を説明する．

### 3.1 移動ステップ

まず，エージェントは木 $T$ の形とラベル $\Lambda$ を求める．ラベル $\Lambda$ 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．

中心が 2 つかつ対称性を持つ場合，移動ステップでは仮の集合点 $p$ を決定し，次の確認ステップで集合点を決定する．2 つの中心を区別することはできないが，2 つの中心を共に端点に持つ辺は唯一に定まる．エージェントは $state$ を Select に変更し，2 つの中心とその間の辺を往復し続ける．そして，$state$ = Select かつ $p$ を知らないエージェント同士がすれ違ったとき，仮の集合点 $p$ を決定する．移動ステップを実行中，もし $g$（または $p$）を知らないエージェントが他のエージェントから $g$（または $p$）や木 $T$，ラベル $\Lambda$ の情報を受け取った場合は，$g$（または $p$）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．

OATS の移動ステップの詳細は以下に示す．

1: $state$ := Explore;
2: Gathering のフェーズ 1 を実行し，葉に到達するまで移動する（到達した葉を $s$ とする）;
3: Gathering のフェーズ 2 を実行し，木 $T$ の形とラベル $\Lambda$ を求める;
4: **if** ラベル $\Lambda$ を持つ木 $T$ が中心を 1 つ持つ，または，中心を 2 つ持ちかつ対称性を持たない **then**
5: Gathering のフェーズ 3 を実行し，集合点 $g$ を求めて移動する
6: **else if** ラベル $\Lambda$ を持つ木 $T$ が中心を 2 つ持ち，かつ，対称性を持つ **then**
7: $s$ からの距離が大きい方の中心へ移動する;
8: $state$ := Select;
9: **while** 仮の集合点 $p$ を決定していない **do**
10: **if** $state$ = Select である他のエージェントと同じ頂点にいる **then**
11: 今いる頂点を仮の集合点 $p$ に決定する
12: **end if**;
13: もう一方の中心へ移動する．移動中に $state$ = Select である他のエージェントとすれ違った場合，ID を比較し，ID が最大のエージェントが進もうとしている頂点を仮の集合点 $p$ に決定する
14: **end while**;
15: 仮の集合点 $p$ へ移動する
16: **end if**;
17: **if** $state$ = Wait のエージェントがいる **then**
18: $state$ := Wait
19: **else**
20: $state$ := Stop
21: **end if**;

移動ステップでは集合点 $g$（または仮の集合点 $p$）に少なくとも 2 人のエージェントが集まることが保証される．

### 3.2 確認ステップ

確認ステップに入ったとき，$state$ は Stop か Wait のどちらかである．確認ステップでは (i)$T$ が中心を 1 つ持つまたは，中心を 2 つ持ちかつ対称性を持たない場合と，(ii)$T$ が中心を 2 つ持ち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 $\Lambda$ を知っているので，(i) と (ii) は区別できる．

(i)$T$ が中心を 1 つ持つまたは，中心を 2 つ持ちかつ対称性を持たない場合を考える．$g$ に集まったエージェントの中から ID が最小のエージェント $a_{min}$を選ぶ．$a_{min}$ は $T$ を走査する．このとき $T$ の走査の途中で，$g$ 以外の頂点で他のエージェントを発見したり，辺で他のエージェントとすれ違ったりしなければ，全てのエージェントは $g$ に集合できていることが分かる．なぜなら，頂点 $g$ にいない任意のエージェント $a$ は基本手順で移動しているので，エージェント $a$ が次に頂点 $g$ に到達するまでに通過する辺の列は，$a_{min}$ が $T$ を走査する際に通過する辺の列の部分列であるからである．よって，エージェント $a$ は $a_{min}$ とすれ違うか $a_{min}$ とすれ違う前に $g$ に到達するかのどちらか一方である．もし，$g$ に全てのエージェントが集合できていなければ，再び $a_{min}$ を選び，$T$ を走査することを繰り返す．具体的な手順は以下に

示す．

**if** $state$ = Stop **then**
  $state$ = Stop のエージェントが $g$ に 2 人以上集まるまで待つ
**else if** $state$ = Wait **then**
  $state$ = Check のエージェントが $T$ を走査し終えるまで待つ
**end if**;
**while** 全てのエージェントが頂点 $g$ にいることが確認できていない **do**
  $g$ に集まっている ID が最小のエージェント $a_{min}$ を選ぶ;
  **if** 自分が $a_{min}$ である **then**
    $state$ := Check;
    基本手順で $T$ を 1 回走査する．このとき，集合点 $g$ 以外の頂点で他のエージェントを発見したり，辺で他のエージェントとすれ違ったりしたか否かを記録する;
  **else**
    $state$ := Wait;
    $a_{min}$ が $T$ を走査し終えるまで待つ
  **end if**;
  $state$ := Stop
**end while**;

(ii)$T$ が中心を 2 つ持ちかつ対称性を持つ場合を考える．仮の集合点 $p$ は中心の一方である．もう一方の中心を $p'$ とすると，$p'$ を集合点に選んでいるエージェントもいる．まず (i) の場合と同様の手順で全てのエージェントが頂点 $p$ か $p'$ のどちらかにいることを確認する．そして，最小の ID を持つエージェントがいる方の中心を集合点に決定する．具体的な手順は以下に示す．

**if** $state$ = Stop **then**
  $state$ = Stop のエージェントが $p$ に 2 人以上集まるまで待つ
**else if** $state$ = Wait **then**
  $state$ = Check のエージェントが $T$ を走査し終えるまで待つ
**end if**;
**while** 全てのエージェントが頂点 $p$ または $p'$ にいることが確認できていない **do**
  $p$ に集まっているエージェントから ID が最小のエージェント $a_{min}$ を選ぶ;
  **if** 自分が $a_{min}$ である **then**
    $state$ := Check;
    基本手順で $T$ を 1 回走査する．このとき，集合点 $g$ 以外の頂点で他のエージェントを発見したり，辺で他のエージェントとすれ違ったりしたか否かを記録する;
  **else**
    $state$ := Wait;
    $a_{min}$ が $T$ を走査し終えるまで待つ
  **end if**;
  $state$ := Stop
**end while**;
$p$ に集まっているエージェントから ID が最小のエージェント $a_{min}$ を選ぶ;
**if** 自分が $a_{min}$ である **then**
  $p'$ に移動し，自身の ID を伝える
**end if**;
ID が最小のエージェントがいる方の中心を集合点 $g$ に決定し，移動する;

確認ステップでは，全てのエージェントが木内の 1 つの頂点に集合することが保証される．

### 3.3 探索ステップ

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 1.** 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，このエージェントが $T$ 内のエージェントは自身 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 OATS アルゴリズムを終了することができない．OATS アルゴリズムより，次の定理が導かれる．

**定理 3.** ラベル $\Lambda$ 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 AATS

ID を持たないエージェントの提案アルゴリズムを AATS(anonymous agents tree search) とする．アル

ゴリズムはエージェントが ID を持つ場合とほぼ同じである．異なる点は ID が使えないので代わりに**計算履歴の列**（通過したポート番号の列）$H$ を用いる点である．$H$ の構成の仕方は，辺 $e$ を通過して頂点 $v$ に到達したとき，$v$ から見た辺 $e$ のポート番号 $\lambda_v(e)$ を列の最後尾に追加していく．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ので，同じ頂点にいる異なるエージェントが同じ計算履歴の列 $H$ を持つことはない．しかし，辺で列 $H$ の比較を行った場合は，同じ $H$ を持つエージェントがいる場合がある．この列 $H$ を ID の代わりに用いて，エージェント同士を比較したり，リーダーの決定に使用したりする．

次の条件 C を満たす場合は，AATS 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 C）以下の 4 つを満たす関数 $f : V \to V$ が存在する．

1. 任意の $v \in V$ について，$v \neq f(v)$ 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，$f(u)$ は $f(v)$ に隣接する．
3. 任意の辺 $(u, v)$ について，$\lambda_u((u, v))$ は $\lambda_{f(u)}((f(u), f(v)))$ 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，$f(v)$ にもエージェントがいる．

AATS 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 つから成る．

### 4.1 移動ステップ

基本的には，OATS アルゴリズムの移動ステップと同じである．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，計算履歴の列 $H$ 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，計算履歴の列 $H$ を辞書式順序で比較し並べ，先頭に来るエージェントを選択する．注意すべき点は，エージェントが列 $H$ の比較を行った際，エージェントの区別ができない場合があることである．図 2(a) はエージェントの初期配置である．2 人のエージェントが常に同じタイミングで移動する場合（同期しているように動く場合）を考える．図 2(b) は 2 人のエージェントが $state$ を Select に変更して初めて辺ですれ違う瞬間を表している．このとき，2 人のエージェントの計算履歴の列 $H$ は等しいので仮の集合点を決定できない．

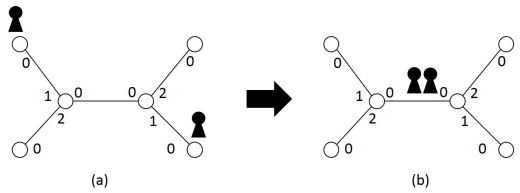


図 2: 異なるエージェント同士の計算履歴の列 $H$ が等しい例

### 4.2 確認ステップ

AATS アルゴリズムの確認ステップは OATS アルゴリズムの確認ステップとほぼ同じである．木 $T$ の形とラベル $\Lambda$ で場合分けをする．

(i)$T$ が中心を 1 つ持つまたは，中心を 2 つ持ちかつ対称性を持たない場合を考える．$g$ に集まったエージェントの計算履歴の列 $H$ を辞書式順序で比較して並べ，先頭に来るエージェントを $a_{min}$ に選び，OAST アルゴリズムの確認ステップと同様の手順で全てのエージェントが $g$ に集まっているかを確認する．

(ii)$T$ が中心を 2 つ持ちかつ対称性を持つ場合を考える．エージェントは仮の集合点 $p$ を決定し集まっている．$p$ は中心の一方であり，もう一方の中心を $p'$ とすると，$p'$ にもエージェントが集まっている場合がある．$p$ に集まったエージェントは (i) と同様の手順で $p$ にいるエージェントの中から $a_{min}$ を選び，$p$ と $p'$ 以外の頂点，辺にエージェントがいないか確認する．全てのエージェントが $p$ または $p'$ のどちらかにいることが分かると，集合点 $g$ を決定する．AATS アルゴリズムでは以下の方法で集合点 $g$ を決定する．再び $a_{min}$ を選び，$p$ に集まっている全てのエージェントの計算履歴の列 $H$ を繋げた列 $\mathcal{H}$ を $p'$ に集まっているエージェントに伝える．$p$ に集まったエージェント $l$ 人の列 $H$ からなる $\mathcal{H}$ は $\mathcal{H} =<$（1 人目の $H$）;（2 人目の $H$）; ... ;（$l$ 人目の $H$）$>$ のように構成する．$p'$ に集まったエージェントも同じ手順を行っていて，$p$ に集まったエージェントに，$p'$ に集まったエージェントの $H$ を繋げた $\mathcal{H}'$ を伝えに来る．$\mathcal{H}$ と $\mathcal{H}'$ を辞書式順序で比較し並べ，$\mathcal{H}, \mathcal{H}'$ の順であれば $p$ を集合点に，$\mathcal{H}', \mathcal{H}$ の順であれば $p'$ を集合点に決定する．全てのエージェントが $g$ に集合したら探索ステップに進む．ただし，アルゴリズム開始時に条件 C を満たすときは，$\mathcal{H}$ と $\mathcal{H}'$ が一致する場合があり，そのときは集合点 $g$ を決定できない．例えば，図 3(a) は条件 C

を満たす例である（図ではエージェントを色で区別しているが，実際はエージェントを区別することはできない）．全てのエージェントが常に同じタイミングで移動する場合を考える．このとき，2 つの中心 $c_a$ と $c_b$ には 2 人ずつエージェントが集まる．図 4(b) はそれぞれのエージェントが仮の集合点を決定して移動した直後である．$c_a$ に集まった 2 人のエージェントの計算履歴の列を繋げた $\mathcal{H}_a$ と，$c_b$ に集まった 2 人のエージェントの計算履歴の列を繋げた $\mathcal{H}_b$ は等しいので，この場合は集合点 $g$ は決定できない．このとき，必ず 2 つの中心のそれぞれに $k/2$ 人のエージェントがいる．エージェントが $k/2$ 人のグループ 2 つが，それぞれが探索ステップに進む．

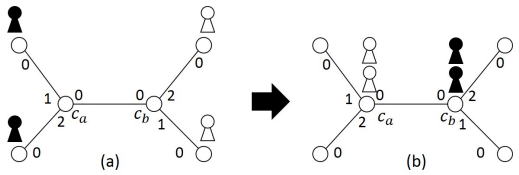


図 3: 集合点を決定できない例

### 4.3 探索ステップ

集合したエージェントの列 $H$ を辞書式順序で比較して並べ，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．確認ステップまでで集合点を 1 つに決定できないのは $T$ 内のエージェント数が偶数のときのみである．つまり，エージェント数が奇数であれば必ず 1 つの頂点に集合できることから，次の定理が導かれる．

**定理 4.** ラベル $\Lambda$ を持つ任意の木 $T$ について，$as_a(T) \leq es(T) + 1$ が成立する．

定理 4 に関して，$es(T) + 1$ 人のエージェントが必要になるのは条件 C を満たしている場合である．また，条件 C を満たす入力について，全てのエージェントが常に同じタイミングで移動する場合を考えると，次の定理が成立する．

**定理 5.** 与えられたラベル $\Lambda$，木 $T$，エージェントの初期配置 $S$ が条件 C を満たす場合，$T$ を $es(T)$ 人のエージェントでオンライン探索する ID を持たないエージェントのアルゴリズムは存在しない．

## 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．そして，提案アルゴリズムでエージェントが $es(T) + 1$ 人必要な木については，任意のオンラインアルゴリズムで $es(T) + 1$ 人のエージェントが必要であることを示した．今後の課題は，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ことである．一般のグラフの探索においても，グラフ内のエージェントの集合問題は課題の 1 つである．

## 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] M. Yamashita, and T. Kameda,”Computing on anonymous networks. I. Characterizing the solvable cases”, *Parallel and Distributed Systems*, IEEE Transactions on 7(1), 69–89, 1996.

[5] 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.

[6] 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.