

頂点重み重み付きグラフ上のペアワイズ行流問題における容量の影響

池田 亮介^{*}; 山内 由紀子[†]

1 はじめに

近年、自律的に移動するモバイルロボットが工学、理論計算機科学、さらに運送や環境観測などでも注目を集めている。例えば、ルンバなどの掃除ロボットや倉庫で荷物を搬送するロボットなどが挙げられる。理論計算機科学、特に分散協調理論の分野では、モバイルロボットの機能と問題の可解性の関係性を明らかにする研究が活発に行われている。この分野では、モバイルロボットは、記憶領域や計算能力が乏しく、識別不能 (匿名) であり、GPS のような大域座標系を知らず、互いに通信を行うことができない低機能なものを想定することが多い。

グラフ上を移動するモバイルロボットに関しては様々な分散協調タスクが考えられてきた。探索問題とは、全ての頂点を少なくとも 1 台のロボットが訪問する問題である。Flocchini らはリング状のグラフで、非同期な無記憶ロボット群による探索問題の解法を提案した [1]。この論文では、グラフの頂点数 n とロボットの総数 k が互いに素であり、かつ $k \geq 17$ の場合、探索問題が解けることが示されている。集合問題とは、全てのロボットが 1 つの頂点に集まる問題である。集合問題はロボットの配置やグラフの構造から、集合する頂点を決定し、その頂点に全てのロボットが集合することで解くことができる。グラフの中心がただ 1 つに定まる場合などは、ロボットが容易に集合する頂点に合意し、その頂点に移動して集合することができる。リングやグリッドといった対称性が高いグラフにおいてもロボットの初期配置、つまりどの頂点にロボットがいるかによって、集合問題を解くことができる場合がある。Klasing らは、非同期な無記憶なロボット群による集合問題の初期配置が対称な場合の解法を提案した [2]。ロボットの初期配置の対称性は初期配置に対する自己同型写像としてとらえることができる。代表的なものが、リング状のグラフの初期配置に対する回転対称性 (回転中心に対する対称性) や鏡映対称性 (軸に対する対称性) である。この論文では、ロボットの総数 k が $k > 18$ かつ、初期配置の対称軸が 1 つしかない場合について集合問題が解けることが示されている。 k が奇数なので、初期配置が対称でも集合する頂点を対称軸上に決定することができる。D'Angelo らはグリッドやツリーで、非同期、無記憶で識別不能なロボット群による集合問題の解法を提案した [3]。この論文ではグリッドについて、縦と横の長さがともに奇数の場合と片方が偶数の場合について、グリッドの縦と横を通る中心線を利用して集合する頂点を決定している。ツリーについては、ツリーの中心と呼ばれる頂点を集合する頂点に決定している。

しかし、現実のモバイルロボットでは、物理的制限があるため、同じ場所に複数のモバイルロボットは存在できない可能性がある。もし、集合しようとする頂点に全てのロボットが存在することができないなら、集合問題を解くことができない。そこで、頂点に存在するロボット数に制限がある場合の集合問題に代わる問題を考える必要がある。

本研究では頂点重みつきグラフにおけるペアワイズ合流問題を提案する。頂点の重みは、各頂点に同時に存在することのできるロボットの最大の数を表す。2 台のロボットがある時点で同じ頂点に同時に存在するとき、その 2 台のロボットは合流したという。ペアワイズ合流問題とは、各

^{*}九州大学工学部電気情報工学科

[†]九州大学大学院システム情報科学研究院

ロボットを他のロボットすべてと合流させる問題である。重みがロボットの台数以上の頂点が存在する場合は、その頂点に全てのロボットを集合させることでペアワイズ合流問題を解くことができるが、すべての頂点の重みがロボットの台数未満の場合は集合によりペアワイズ合流問題を解くことはできない。本研究では、頂点に少数のロボットを集合させることを繰り返してペアワイズ合流問題を解く手法を与える。

ペアワイズ合流問題に関する既存研究として、非同期なロボット群によるペアワイズ合流問題 [4] がある。この研究では、非同期で無記憶なロボット群によるペアワイズ合流問題では、ロボットの台数が頂点の容量の最大値を超える時、解くことができないことが示された。加えて、非同期で有記憶なロボット群では、全ての頂点の容量が3以上の時、非対称な2分木上でペアワイズ合流問題を解けることが示された。しかし、この研究では全ての頂点の容量によって、実行時間が増加することがないため、頂点の容量が実行時間に与える影響を考慮することができなかった。

そこで、本研究ではペアワイズ合流問題を解く場合の頂点の重みが実行時間に与える影響を考える。まず、同期で有記憶なロボット群では、非対称な木でペアワイズ合流問題が解けることを示す。次に、提案アルゴリズムのロボットが合流を行う部分について、全頂点の容量を a 倍にしたとき、実行時間すなわちロボット全体の移動回数が $1/a^2$ 以下になることを示す。

2 準備

n 頂点の頂点重み付き無向グラフ $G = (V, E)$ 上を自律的に移動する 匿名な k 台のモバイルロボットの集合 $R = \{r_1, r_2, \dots, r_k\}$ を想定する。 r_i という表記は説明のため使用する。 n は $k < n$ を満たすとする。時刻 t に r_i が存在する頂点を $p_i(t)$ と表す。時刻 t に r_i と同じ頂点 $p_i(t)$ に存在するロボット群を $R'_i(t) \subseteq R$ と表す。多重集合 $P(t) = \{p_1(t), p_2(t), \dots, p_k(t)\}$ を R の時刻 t での配置という。 $P(0)$ を初期配置という。

各ロボット r_i は観測、計算、移動という3つのフェーズから成るサイクルを繰り返し実行する。観測フェーズでは、 r_i は観測フェーズ中のある時点でグラフのどの頂点に何台のロボットが存在するかを得る。ただし、各ロボットはグラフの頂点を識別できないので、時刻 t における観測結果は G の自己同型写像を φ とすると、配置 $\{\varphi(p_1(t)), \varphi(p_2(t)), \dots, \varphi(p_n(t))\}$ となる。よって、観測結果はロボット r_i と r_j ($r_i \neq r_j$) で一致するとは限らない。計算フェーズでは、 r_i は観測フェーズの観測結果をもとに、現在いる頂点ににとどまるか、とどまらないなら現在いる頂点のどの隣接頂点に移動するかを共通の決定性アルゴリズム A を用いて計算する。移動フェーズでは、 r_i は直前の計算で求めた頂点に移動する。この移動にかかる時間は一瞬であり、1回の移動で0または1つの辺を移動する。

完全同期モデルでは、観測、計算、移動フェーズの長さが決まっており、全てのロボットが同じタイミングで、観測、計算、移動フェーズを行う。本論文では、モデル完全同期モデルを想定する。

各頂点 $v_i \in V$ の重み w_i に対して、その頂点には同時に w_i 台までしかロボットが存在できないとする。ある頂点 v_i に同時に複数の接続辺から多数のロボットが移動しようとする時、次の時刻に v_i に存在するロボットの台数が w_i を超える場合、次の時刻に v_i に存在するロボットが w_i 台になるように、移動するロボットが選ばれるものとする。

頂点 $v_i \in V$ の次数を δ_i と表す。 v_i に接続する各辺には、ポート番号として、 $\{0, 1, \dots, \delta_i - 1\}$ に含まれる数が重複することなく割り当てられるとする。ポート番号の割り当て方は、各頂点で任意となっている。ロボットは移動を行うと、移動した頂点にどのポートから入ってきたかを認識できるものとする。

本研究では、グラフ G として、 n 個の頂点からなる木を想定する。ある頂点 v_i を根とした根付き木において、頂点 v_j の深さとは頂点 v_i と v_j までに存在する辺の数である。根 v_i の深さは0とする。

ロボットの初期配置の対称性を考える。グラフ G 上のロボットの配置 P が対称なのは、 $\forall v \in V$ に対して、頂点 v と $\varphi(v)$ にいるロボット数が一致するような自己同型写像 $\varphi: V \rightarrow V$ が存在する

ときである．グラフ G 上のロボットの配置 P が非対称について，そのような自己同型写像が存在しないとき， P は**非対称**であるという．図 1 に，木における対称と非対称な初期配置の例を示す．

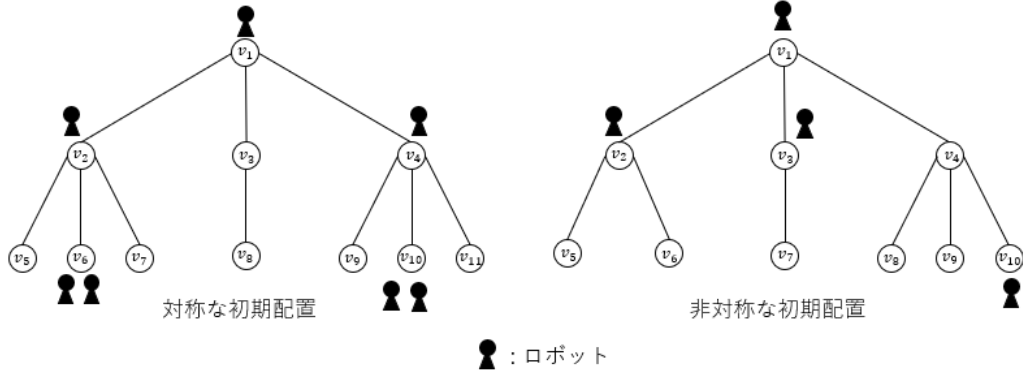


図 1: 対称と非対称な初期配置の例

例えばパスグラフ上のロボットの配置 P が対称について， n が偶数のとき， $1 \leq i \leq n/2$ を満たす任意の i に対して $d_i(0) = d_{n-i}(0)$ が， n が奇数のとき， $1 \leq i \leq (n-1)/2$ を満たす任意の i に対して， $d_i(0) = d_{n-i+1}(0)$ が成立するとき， P は対称である．2 分木におけるロボットの初期配置が対称とは，中心（中心が 2 つある場合は，その間）を通る直線に対して，頂点に存在するロボットの台数が一致する対応が存在することである．

初期配置 $P(0)$ から，完全同期モデルのロボット群がアルゴリズム A を実行する場合， $P(0), P(1), P(2), \dots$ を A の**実行**と呼ぶ．初期配置は，多重点が存在せず非対称なものであると仮定する．

ペアワイズ合流問題 とは各ロボットを他の全てのロボットと**合流**させる問題である．ここで，ある実行 $P(0), P(1), P(2), \dots$ において，ロボット r_i と r_j が**合流**するとは， $p_i(t) = p_j(t)$ となる t が存在することである．ロボットの部分集合 $R' \subseteq R$ が**合流**するとは， R' に含まれる任意の r_i, r_j が合流することである．同じ頂点に複数のロボットが存在しないような初期配置 $P(0)$ から始まるアルゴリズム A の任意の実行 $P(0), P(1), P(2), \dots$ について，有限の $T \geq 1$ が存在し，各ペア $r_i, r_j \in R (i \neq j)$ について， $p_i(t) = p_j(t)$ を満たす $1 \leq t \leq T$ が存在するとき， A は初期配置 $P(0)$ から**ペアワイズ合流問題を解く** という．

与えられた 2 分木 $G = (V, E)$ から根を一意に決定するために必要な**中心**を定義する．まず，頂点 v における**離心率** $\epsilon(v)$ とは， v から最も離れた頂点 w までの距離である．すなわち， $\epsilon(v) = \max d(v, w)$ ($w \in V, w \neq v$) と表される．任意のグラフにおける中心とは，離心率が最小になる頂点である．中心は，グラフの構造から決定することができ，木グラフに必ず 1 つもしくは 2 つ存在する．

3 提案アルゴリズム

提案アルゴリズムは，ブロック作成と合流の 2 つのフェーズに分かれており，ブロック作成，合流という順で行う．ブロック作成フェーズでは， k 台のロボットを， $c/2$ 台ずつのロボット群に分けそれらを中心とその付近の頂点に配置する．合流フェーズでは，まず，作成した配置で中心にいる $c/2$ 台のロボット群が他の $c/2$ 台のロボット群がいる頂点を深さ優先探索することで，自身以外のロボットと合流する．そして，探索を行う $c/2$ 台のロボット群は全てのロボットと合流すると停止し，停止した頂点にいる次の $c/2$ 台のロボット群が中心に戻ってから同様に深さ優先探索を開始する．

3.1 根を一意に決定する方法

木 $G = (V, E)$ から、根 v_{root} を一意に決定する方法を考える。まず、中心が1つなら根 v_{root} として中心が選ばれる。次に、中心が2つある場合を考える。2つの中心を u, v とすると、 u, v は隣接しているので、辺 $\{u, v\} \in E$ を取り除いて得られる2つの木のうち、 u を根とした根付き木を T_u 、 v を根とした根付き木を T_v とし、 T_u と T_v の構造の違いを利用することで根 r を1つに決定する。

根付き木に関しては、その構造を文字列にする AHU アルゴリズム [5] が知られている。この AHU アルゴリズムは、根付き木の同型性判定に用いられている。ここで、根付き木 $T_u = (V_u, E_u)$ と $T_v = (V_v, E_v)$ が同型であるとは、 $\forall u_1, v_1 \in V_u$ において、 $(u_1, v_1) \in E \Leftrightarrow (f(u_1), f(v_1))$ かつ $f(u) = v$ を満たす写像 $f: V_u \rightarrow V_v$ が存在することである。このアルゴリズムでは、まず、葉に文字列 10 を割り当てる。深さ k の頂点 u_k について、 u_k の子供に割り当てられている文字列を辞書式順序に昇順で並び替え、その順に文字列を結合する。そして、結合した文字列の前に 1 を、後ろに 0 を結合した文字列を u_k に割り当てる。最後に、根に割り当てられた文字列を出力し、アルゴリズムが終了する。Algorithm 1 に AHU アルゴリズムを示す。

Algorithm 1 $AHU(v_i)$

```

1: if  $v_i$  が葉ノード then
2:   return 10
3: else
4:    $temp = (w \in child(v_i) \text{ の } AHU(w) \text{ を辞書式順の昇順に並べ、連結する})$ 
5: end if
6: return 1temp0

```

T_u, T_v に対して、 u または v を入力として AHU アルゴリズムを行った結果を $AHU(u)$ と $AHU(v)$ とする。 $AHU(u)$ と $AHU(v)$ を辞書式順序で比較して、 $AHU(u)$ が大きければ v_{root} として u を選び、 $AHU(v)$ が大きければ v_{root} として v を選ぶ。提案アルゴリズムの仮定より、与えられた2分木は中心が2つある場合、2つの中心を結ぶ辺に対して非対称なので、 T_u と T_v が同型ではないことが分かる。提案アルゴリズムの仮定より、 $AHU(u) \neq AHU(v)$ となるので、 u と v のうち、いずれかが必ず v_{root} に選ばれる。ここで、頂点 u, v について、 $AHU(u)$ が $AHU(v)$ より辞書的に大きいことを $AHU(u) > AHU(v)$ と表し、 $AHU(u)$ が $AHU(v)$ より辞書的に小さいことを $AHU(u) < AHU(v)$ と表す。

図2に、中心が2個ある木グラフで、 T_u と T_v のそれぞれに AHU アルゴリズムを行った例を示す。各頂点に書かれている文字列は、AHU アルゴリズムの実行中に割り当てられた文字列を表す。 $AHU(u) = 1110011000$ と $AHU(v) = 1101101000$ であり、この2つの文字列を辞書式順序で比較すると、 $AHU(u)$ の方が大きいので v_{root} として u が選ばれる。与えられた2分木 $T = (V, E)$ を前述の方法で選んだ頂点 v_{root} を根とする根付き木と見なす。

3.2 アルゴリズム中の用語の定義

同じ頂点に存在している2台以上のロボット群のことを**タワー**と呼び、タワーに属しているロボットの数をタワーの**サイズ**と呼ぶ。

提案アルゴリズムにおいて、ある配置に対して移動を行わなければならないロボットを**プレイヤー**と呼ぶ。計算の結果ロボットが進むべき隣接頂点のことを**目的地**と呼ぶ。

v_{root} を根とする木について、深さ i の頂点の集合をレベル i の頂点集合 L_i と記述する。例えば、レベル0の頂点集合は $L_0 = v_{root}$ で、レベル1の頂点集合は、 v_{root} の子頂点全てである。深さ i にある頂点 v を根とした、深さ i 以上の全ての頂点を含む部分木を $T(v)$ と呼ぶ。一方、深さ i にある頂点 v を根とした、深さ j までの全ての頂点を含む木を $T(v, j)$ ($i \leq j$) と呼ぶ。木 $T(v), T(v, j)$ の頂点数をそれぞれ $|T(v)|, |T(v, j)|$ と表す。木 $T(v)$ にいるロボットの総数を $D(T(v))$ と表す。

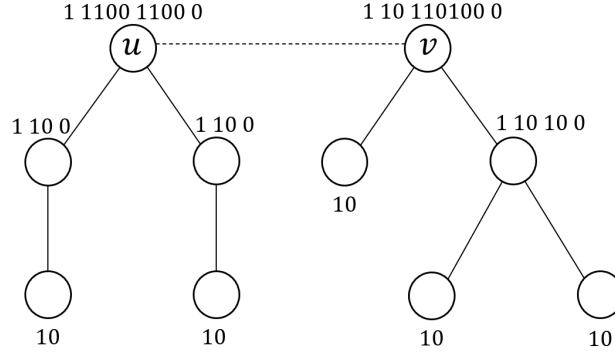


図 2: AHU アルゴリズムの実行例. 頂点に並記した 01 の文字列が AHU アルゴリズムがその頂点に与えた文字列.

頂点の容量の半分を表す c' を $2c' \leq c$ を満たす最大の c' とする. 提案アルゴリズムで作成する配置を c' ブロックと呼ぶ. c' ブロックの配置では, 根の周りにサイズ c' のタワーを集めることで, タワーが存在する頂点の範囲の深さを抑えることを目標とする. 具体的には, 深さ max_d までにすべてのタワーが存在することを目指す. ここで, **ブロックの最大の深さ** max_d を $k \leq c' \times |T(v_{root}, i)|$ を満たす最小の i とする. c' ブロックでは, 深さ $(max_d - 1)$ 以下の頂点には必ずサイズ c' のタワーが存在し, 深さ max_d にはサイズ c' のタワーとサイズ c' 未満のタワー存在しても良く, 深さ $(max_d + 1)$ 以上の頂点には必ずタワーが存在しない配置である. 図 3 に c' ブロックと用語の例を示す.

木の中の頂点 v が**バランスが取れている**とは, v の全ての子頂点 v_c について,

$$c' \times |T(v_c, max_d - 1)| \leq D(T(v_c)) \leq c' \times |T(v_c, max_d)|$$

が成り立つことをいう.

頂点 v の子頂点 v_c の内, v_{cmax} を $D(T(v_c))$ が最大の v_c とし, v_{cmin} を

$$D(T(v_c)) < c' \times |T(v_c, max_d - 1)|$$

を満たす中で $D(T(v_c))$ が最小の v_c とする.

ロボット r_i はそれぞれ変数 $complete_i$ を持っているものとする. 初期値は *false* である. c' ブロックを作成する際, 目的の深さでサイズ c' のタワーが完成されたとき, $complete_i = true$ にすることで, r_i は**完成したタワー**に含まれるようになる. 完成されたタワーに属するロボットは, 合流フェーズが開始するまで, その頂点から移動することはない.

3.3 同期性を用いた衝突の解消方法

提案アルゴリズムの実行中において, 複数の子頂点から同時にロボットが 1 つの親頂点に移動してきた場合, それらのロボットが親頂点でタワーを形成してしまう. この現象を**衝突**と呼ぶ. 衝突したロボット群は, 直前に自分がどのポート番号から来たかを覚えていないと衝突以降は同時に同じ頂点動いてしまうようになる. この衝突を解消するため, 複数の子頂点から同時に 1 つの親頂点に入った場合, 任意の台数のロボットだけがその親頂点に残ることができる方法を考える. この方法では, ロボット r_i は, 衝突してから時間を計測する変数 $timer_i$ を持っている. $timer_i$ の初期値は 0 である. 以下では, ロボット r_i が行う具体的な手順について説明する.

自身がいる頂点で衝突が発生しており, 残したいロボットの台数よりも多くのロボットが存在する場合, ロボット r_i は $timer_i$ が以前自身が通ったポート番号と比較する. 比較したとき, $timer_i$

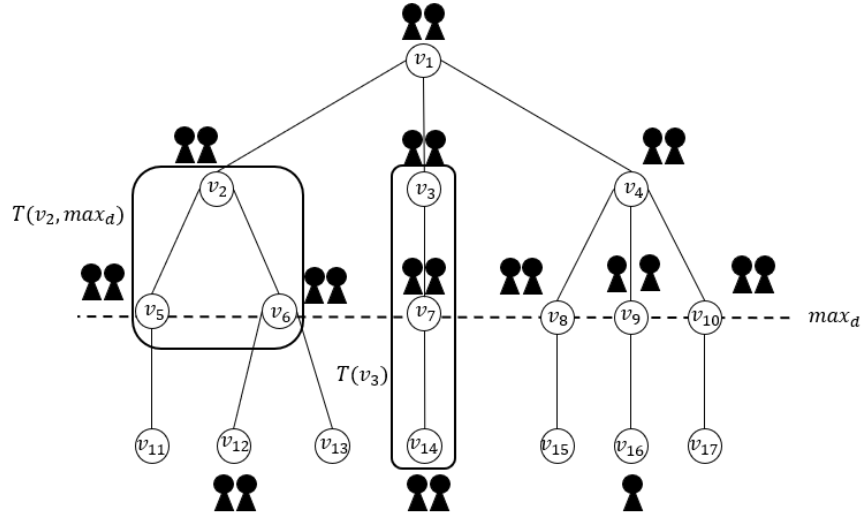


図 3: c' ブロックの例

と自身が通ったポート番号が一致するなら, r_i はそのポート番号の先にある頂点へ移動する. 一致していない場合は, $timer_i$ を 1 増やす.

この手順の結果, $timer_i$ が 1 ずつ増えていくと直前に通ったポート番号が小さいロボットから順に移動するため, ポート番号が最も大きいロボットから目的の台数のロボットが残るようになる.

3.4 ブロック作成フェーズ

ブロック作成フェーズでは, まず, 深さ $i = 0$ の全ての頂点に c' 台のロボットを配置し, 深さ $(i+1)$ 以上の全ての頂点でも c' 台のロボットが配置できるように, 深さ $(i+1)$ 以上の頂点にいるロボットを移動させる. そして, 深さ $i = 1, 2, \dots$ の頂点についてこの動作を, 全ての頂点に c' 台のロボットが配置できなくなるまで行う. 最後に, c' より少ないロボットが存在する頂点にいるロボットを根を介して移動させることで, c' より少ないロボットがいる頂点が 1 つ以下になるように調整する.

以下では, 深さ i の全頂点に c' のタワーを配置するアルゴリズムを, 各配置に対してプレイヤーと目的地を決めることで示す. アルゴリズムにおいて, 頂点 v は $v \in L_i$ を満たすものとする.

- ステップ 1 の配置
深さ $i+1$ 以上の頂点 w にタワーが存在する配置である. この場合のプレイヤーは, w にいるロボットで, 目的地は前述した衝突回避に基づいて決める.
- ステップ 2 の配置
 v に完成したタワーができていない配置である. この時, v にあるタワーのサイズが $c' - 1$ 以下か, $c' + 1$ 以上かでステップ 2a と 2b に分ける.
 - ステップ 2a の配置
この配置のプレイヤーは, v にいるロボットで, 目的地は衝突回避を行うことで, v に c' 台のロボットが残るように決める
 - ステップ 2b の配置
この配置のプレイヤーは, v に最も近いロボットで, 目的地は, 頂点 v である.

- ステップ 3 の配置
 v に完成したタワーができており、かつ $T(v)$ がバランスが取れていない配置である。この場合、 v にあるタワーのサイズが c' か $c' + 1$ 以上かでステップ 3a と 3b に分ける
 - ステップ 3a の配置
 タワーのサイズが c' の配置である。この場合のプレイヤーは、 $T(v_{cmax})$ に属しており、 v_{cmax} に最も近いロボットで、目的地は、頂点 v_c である。
 - ステップ 3b の配置
 タワーのサイズが $c' + 1$ 以上の配置である。この場合、プレイヤーは v にいる頂点の内、完成したタワーに含まれていないロボットで、目的地は v_{cmin} である。目的地が複数存在する場合は、ポート番号が最も小さい方へ進む。
- ステップ 4 の配置
 v にサイズ c' のタワーができており、かつ $T(v)$ がバランスが取れている配置である。この場合、深さ i の全ての頂点にタワーを配置できたといえるので、深さを $i + 1$ にして同様にアルゴリズムを実行する。

深さ max_d のブロック形成では、サイズ c' 未満のタワーの数が 1 つ以下になるように、深さ max_d の頂点にタワーを配置する。サイズ c' 未満のタワーの数が 1 つ以下にする調整を考える。各頂点にいるロボット数を c' で割った時に 0 以外で最も少ない頂点を $m_{c'min}$ 、最も多い頂点を $m_{c'max}$ とする。調整は、衝突回避を行いながら、 $m_{c'min}$ にいる頂点を $m_{c'max}$ に送り続けることで行う。以下では、各配置に対してプレイヤーと目的地を示すことで、アルゴリズムを示す。

- ステップ 5 の配置
 深さ $max_d + 1$ 以上に頂点がある配置である。この場合のプレイヤーは、深さ $max_d + 1$ 以上の頂点の内、最も小さい深さにいるロボットで、目的地は、親頂点である。
- ステップ 6 の配置
 深さ $max_d + 1$ 以上に頂点がない配置である。この場合、深さ $max_d - 1$ 以下の頂点 w にあるタワーのサイズで、ステップ 6a から 6c に分けることができる。
 - ステップ 6a の配置
 深さ $max_d - 1$ 以下の頂点 w に、サイズ $c' + 2$ のタワーが存在する配置である。この場合のプレイヤーは、 w にいるロボットの内、完成したタワーに含まれてないロボットで、目的地は、衝突回避に基づいて決定する。
 - ステップ 6b の配置
 深さ $max_d - 1$ 以下の頂点 w に、サイズ $c' + 1$ のタワーが存在する配置である。この場合のプレイヤーは、 w にいるロボットの内、完成したタワーに含まれてないロボットで、目的地は、最も近い $m_{c'max}$ である。
 - ステップ 6c の配置
 深さ $max_d - 1$ 以下の頂点に、サイズ $c' + 2$ のタワーが存在しない配置である。この場合のプレイヤーは、 $m_{c'max}$ に最も近い $m_{c'min}$ にいるロボットである。目的地は、 $m_{c'max}$ 方向である。
- ステップ 7 の配置
 深さ $max_d - 1$ 以下の頂点には必ずサイズ c' のタワーが存在し、深さ max_d にはサイズ c' のタワーとサイズ c' 未満のタワー存在しても良く、深さ $max_d + 1$ 以上の頂点には必ずタワーが存在しない配置である。この場合の配置は、 c' ブロックとなるので、ブロック作成を行うアルゴリズムは終了する。

3.5 合流を行うアルゴリズム

合流を行うアルゴリズムでは、作成した c' ブロックから始める。 c' ブロックが存在する初期配置において、ロボットが存在する頂点数を S とする。提案アルゴリズムの概要としては、まず根にいる c' 台のロボットがロボットが存在する頂点を深さ優先探索を用いて訪れる。訪れた頂点が新しい頂点か以前訪れた頂点かは、元来たポート番号が親頂点か子頂点かどうかで区別する。そして、 S 個のロボットが存在する頂点を訪れると、根から深さ優先探索を始めた c' 台のロボットは停止する。そして、元からその頂点にいた c' 台のロボットが根に戻り、再び根から $S - 1$ 個の頂点を訪れるまで探索する。探索を行っていない c' 台の頂点は、ロボット群が同期実行を行っており、各ロボット群が探索を始める瞬間を認識できるなら、ロボット群が探索を終了する瞬間も認識することができる。そしてこの動作を c' 台のロボットが 1 個の頂点を訪れるようになるまで行う。以下では、合流フェーズのアルゴリズムを擬似コードで示すために必要な変数について説明する。

- $excounter_i$
 c' 台のロボット群の深さ優先探索が何回目であるかを記録する。初期値は、1 である。ここで、探索の開始は、根から c' のロボット群が移動を始めた瞬間で探索の終了は、深さ優先探索を行うロボットを交代してから、交代したロボットが根に戻るまでである。
- $extimer_i$
 c' 台のロボット群が探索を開始してから時間を記録する。初期値は 0 である。
- $stack_i$
 c' 台のロボット群の深さ優先探索を行うために、新しく頂点を訪れた時、次に訪れるべき頂点に繋がるポート番号を格納するために使用する。初め、 $stack_i$ は空であるとする。
- $depth_i$
 r_i が現在いる頂点の深さを表す。
- $mode_i$
ロボット r_i が探索を行うロボットかを示す。 $mode_i = 0$ の時は、深さ優先探索は行わず停止しており、 $mode_i = 1$ の時は、深さ優先探索を行う。

合流を行う際のロボット r_i アルゴリズムの擬似コードを以下の通りである。

Algorithm 2 *DFS at robot r_i*

Input: $excounter_i, extimer_i, stack_i, depth_i, mode_i$

```
1: if  $extimer_i < 2(S - excounter_i) - detph$  then
2:   // 深さ優先探索を行う
3:   if 元来た頂点が親頂点 then
4:      $stack_i$  に親頂点のポート番号を挿入
5:      $stack_i$  に子頂点のポート番号を小さい順に挿入
6:   end if
7:    $stack_i$  から 1 つポート番号を取り出し, その頂点へ移動
8: else if  $extimer_i = 2(S - excounter_i) - detph$  then
9:   // 1 回の探索が終了
10:   $mode_i \leftarrow 0$ 
11: else if  $extimer_i < 2(S - excounter_i)$  then
12:   根方向に移動
13: else if  $extimer_i = 2(S - excounter_i)$  then
14:   // 探索するロボット群が完全に入れ替わった
15:   // カウンターをリセット
16:    $extimer_i \leftarrow 0$ 
17:    $excounter_i \leftarrow excounter_i + 1$ 
18:   // 深さ優先探索を開始
19:    $stack_i$  に子頂点のポート番号を挿入
20:    $stack_i$  から 1 つポート番号を取り出し, その頂点へ移動
21: end if
```

Algorithm 3 *COUNT at robot r_i*

Input: $excounter_i, extimer_i, stack_i, depth_i, mode_i$

```
1: if  $extimer_i = 2(S - excounter_i) - depth_i$  then
2:   // 探索するロボットを切り替える瞬間
3:   if 現在いる頂点にいるロボット数が  $c'$  台より多い then
4:      $mode_i \leftarrow 1$ 
5:     DFS を実行
6:   end if
7: else if  $extimer_i = 2(S - excounter_i)$  then
8:   // 1 回の探索が終了
9:   // カウンターをリセット
10:   $extimer_i \leftarrow 0$ 
11:   $excounter_i \leftarrow excounter_i + 1$ 
12: end if
```

アルゴリズム 4 が実際に合流を行うアルゴリズムである.

Algorithm 4 *MEETING at robot r_i*

Input: $excounter_i, extimer_i, stack_i, depth_i, mode_i$

```
1: if  $mode_i = 0$  then
2:   COUNT を実行
3: else if  $mode_i = 1$  then
4:   DFS を実行
5: end if
```

4 頂点の容量の合流アルゴリズムの実行時間に対する効果

本章では, 3.2 節で示した合流を行うアルゴリズムに対して, 容量を増やしたときに実行時間がどのように変化するかを考察する. 結果としては, 次の定理に示すようなものになった.

定理 4.1 ロボットの台数を k , 各頂点の容量を c とする. この時, 3.5 節で示された合流を行うアルゴリズムのロボット全体の移動回数の上界は, c を $a(\geq 1)$ 倍すると, $1/a^2$ 以下の回数になる.

証明. 3.5 節で示した合流を行うアルゴリズムのロボット全体の移動回数を考える. 初期配置である c' ブロックにおけるロボットが存在する頂点数を S とする. ここで S は, $S = \lfloor k/c' \rfloor$ を満たす. まず, 初期配置から探索を始めた時, 根にいる c' 台のロボットが S 個の頂点を訪問し終えたときにいる頂点を v とする. この時, c' 台のロボット, v_{root} と v の間のパス上にいる頂点については 1 回訪問しており, それ以外のロボットがいる頂点については 2 回訪問している. ここで, 次に c' ブロック内を探索するロボットが根に戻るまでの過程を一回の探索と考えると, v_{root} と v の間のパス上の頂点も 2 回訪問されていると考えることができる. そして, 次に探索を行うロボットは $S - 1$ 個の頂点を訪問し, その次に探索を行うロボット群は $S - 2$ 個の頂点を訪問すると考えると全体の移動回数は次のように考えることができる.

$$2(S - 1) + 2(S - 2) + \dots + 2 + 0 = \sum_{i=1}^S 2(S - i) = S(S - 1)$$

ここで, $S = \lfloor k/c' \rfloor$ であるため, 移動回数の範囲は,

$$(k/c' - 1)(k/c' - 2) \leq S(S - 1) \leq (k/c')(k/c' - 1)$$

である. よって, 上界は $(k/c')(k/c' - 1)$ である. 上界を c を用いて表す時, $c = 2c'$ と $c = 2c' + 1$ の両方が存在するので, それぞれの場合について c が a 倍されたときを考える. まず, $c = 2c'$ で c が a 倍されたときの移動回数の上界は,

$$(2k/ac)(2k/ac - 1) = 1/a^2 \times (2k/c)(2k/c - a) \leq 1/a^2 \times (2k/c)(2k/c - 1)$$

同様に, $c = 2c' + 1$ で c が a 倍されたときの移動回数の上界は,

$$\{2k/a(c - 1)\}\{2k/a(c - 1) - 1\} = 1/a^2 \times \{2k/(c - 1)\}\{2k/(c - 1) - a\} \leq 1/a^2 \times \{2k/(c - 1)\}\{2k/(c - 1) - 1\}$$

以上より, c を a 倍した時の移動回数の上界は, $1/a^2$ 以下になっていることが示された. □

5 まとめ

本研究では, 同期実行における木でペアワイズ合流問題を解くアルゴリズムを考え, 木の頂点の容量が合流を行うアルゴリズムの実行時間すなわち移動回数に与える影響を考察した. 結果として, 容量を a 倍すると, 移動回数の上界と下界は $1/a^2$ になることが分かった.

今後の課題としては, 一般のグラフでもペアワイズ合流問題が解けるかどうかや, 容量が各頂点で異なる場合の実行時間に対する影響を考察していきたい.

References

- [1] Paola Flocchini, David Ilcinkas, Andrzej Pelc, and Nicola Santoro, "Computing Without Communicating: Ring Exploration by Asynchronous Oblivious Robots", *Algorithmica*, 65(3), pp. 562–583, 2013.

- [2] Ralf Klasing, Adrian Kosowski, and Alfredo Navarra, "Taking advantage of symmetries: Gathering of many asynchronous oblivious robots on a ring", *Theoretical Computer Science*, 411, pp. 3235–3246, 2010.
- [3] Gianlorenzo D'Angelo, Gabriele Di Stefano, Ralf Klasing, and Alfredo Navarra, "Gathering of robots on anonymous grids and trees without multiplicity detection", *Theoretical Computer Science*, 610, pp. 158–168, 2016.
- [4] 池田亮介, 山内由紀子, "非同期なモバイルロボット群のためのペアワイズ合流アルゴリズム", 九州大学工学部電気情報工学科卒業論文, 2024 年 2 月.
- [5] A. Aho, J. Hopcroft, and J. Ullman, "The Design and Analysis of Computer Algorithms." Addison-Wesley Publishing, 1974.