Vol.2010-MPS-77 No.37
2010/3/5

MANET 環境におけるノードの移動特性を考慮した自己安定クラスタリング手法

黒岩 潤平[†1]　山内 由紀子[†1]
孫 為華[†1]　伊藤 実[†1]

モバイルアドホックネットワーク (MANET) は無線通信可能な移動端末が自律的にネットワークを形成し，インフラの敷設なしにシステムを構築可能なネットワークである．クラスタリングはネットワークの階層的な管理を実現する手法であり，MANET 環境でしばしば用いられる．Johnen らは，MANET を頂点重み付きグラフと見なし，頂点重み付きグラフにおける自己安定クラスタリングアルゴリズムを提案した．このアルゴリズムはネットワークの変化に対する自律適応性を有しているが，クラスタの安定性を考慮していない．本稿では，Johnen らが提案したクラスタリングアルゴリズムのクラスタの安定性を改善するために，ノードの移動特性に基づいたノードの重み割り当て手法を提案する．シミュレーション評価実験により，提案手法はクラスタヘッドの変化回数に関して，従来手法と比較して 34% 改善することを確認した．

A self-stabilizing clustering algorithm based on mobility pattern of nodes in mobile ad-hoc networks

Junpei Kuroiwa,[†1] Yukiko Yamauchi,[†1]
Weihua Sun [†1] and Minoru Ito[†1]

A mobile ad-hoc network (MANET) consists of mobile nodes that communicate with each other by wireless communications without any fixed infrastructure. Clustering is a method for hierarchical management of a network, which is often used in MANETs. Johnen et al. proposed a self-stabilizing clustering algorithm which considers a MANET as a weighted graph. The algorithm has autonomous adaptability against changes in a network, but does not consider the stability of clusters. In this paper, we present a weight assignment method for Johnen's algorithm that reflects the mobility of a node to its weight so that the stability of clusters is improved. Simulation results show that the proposed method improves the number of changes in clusterheads of the former method by 34% .

1. はじめに

近年，携帯電話や PDA などの無線端末が広く普及し，モバイルアドホックネットワーク (MANET) への注目が高まっている．MANET では無線通信可能な移動端末 (ノード) が自律的にネットワークを形成するため，インフラの敷設なしにシステムを構築することができる．MANET の利用例としてはインフラに頼ることができない災害救助現場での使用や，低コストで一時的なネットワークの敷設が必要とされるイベント会場での使用が挙げられる．大規模な MANET を効率的に管理することが必要とされているが，スケーラビリティのために集中管理は適しておらず，また，各ノードのバッテリ容量は限られているため，通信量の少ない効率的な手法が必要とされている．MANET を対象としたアルゴリズムには，時々刻々と変化するネットワークトポロジにシステムが自律的に適応する自律適応性や，個々の変化に影響されず，ユーザに安定した機能を提供し続ける安定性が必要とされている．この 2 つの性質はネットワークの変化に対して互いに異なるふるまいを要求する性質であるが，MANET を対象とした，高い可用性をもつシステムを運用する上で必要不可欠である．

大規模なネットワークを効率良く管理する手法として，**クラスタリング**がよく用いられている．クラスタリングとは，ネットワークをクラスタと呼ばれる複数のグループに分割する手法である．各クラスタは**クラスタヘッド**と呼ばれる 1 つのノードと，**通常ノード**と呼ばれるいくつかのノードから成り，クラスタ内のノードの管理，クラスタ間の通信などをクラスタヘッドが一括して行う．クラスタリングはネットワークの階層的な管理を実現し，自律分散的なネットワーク管理の基盤として利用されている．

MANET ではノードの移動によるネットワークトポロジの変化や，個々のノードの一時的な故障といったネットワークの変化が頻発する．ネットワークの変化やノードの故障に対する自律適応性を備えた分散アルゴリズムの設計手法として，**自己安定**という概念[6] がよく知られている．自己安定アルゴリズムは，システムの初期状況に関わらず，システムがやがて目的としたふるまいを行うことを保証する．つまり，ネットワークの変化やノードの故障後の状況を新たな初期状況とし，そこからシステムが自律的に目的の性質を満たす状況（解状況）へと到達することを保証する．また，システム全体の初期化なしに運用を開始で

†1 奈良先端科学技術大学院大学
Nara Institute of Science and Technology

きるといった利点もある．このような性質から，MANET を対象とした自己安定アルゴリズムが近年よく研究されている[2),8),10)]．

本稿では，MANET 環境におけるノードのモビリティを考慮した自己安定クラスタリング手法を提案する．ノードの移動により，クラスタヘッドの交代や，通常ノードが属するクラスタの変更が起こると，通信のオーバーヘッドが増える可能性がある．また，クラスタの再計算が頻発するとシステムの可用性も低下する．そこで，MANET におけるノードのモビリティの特性に着目し，これをもとに長期間安定してクラスタヘッドの役目を担うノードを発見することにより，クラスタの安定性を実現する手法を提案する．

ノードのモビリティパターンに関して，携帯端末を持った歩行者がフィールド内を移動する状況を想定する．特に，イベント会場や駅周辺などをフィールドとして想定する．このようなフィールドでは，歩行者はしばしばある程度のグループを形成し，ある一定期間同じような移動を行う．従って，グループごとにクラスタを構成すれば，安定したクラスタを提供できると考えられる．また，ノードの移動によって異なるグループ同士が一時的に近付き，再び離れるといった，グループが交差するような状況でも，交差の前後でクラスタ構造が維持されることが望ましい．本稿では，このようなグループを割り出し，グループの重心，グループのノード数，グループ内での移動ベクトルをもとにクラスタヘッドを計算することにより，安定したクラスタリングを実現する手法を提案する．

提案手法の性能を評価するため，グループで移動するような MANET 環境を想定したシミュレーション実験を行い，提案手法によるクラスタの安定性の改善を検証した．実験結果より，従来の自己安定クラスタリング手法と比較して，提案手法はクラスタヘッドの変化回数といったクラスタリングの安定性の評価指標に関して改善されており，クラスタヘッドの総数といったクラスタリングの質をほぼ同程度に保ちながら，クラスタの安定性を実現できていることが確認できた．

2. 関 連 研 究

本章では MANET を対象としたクラスタリングに関する関連研究について述べ，従来の手法でまだ十分考慮されていない点を明らかにすることで，提案手法の必要性を示す．2.1 節で MANET のモビリティを考慮したクラスタリングに関する関連研究について述べ，2.2 節で MANET におけるノードのモビリティ特性に関する関連研究について述べる．

2.1 MANET に対するクラスタリング

MANET におけるノードのモビリティやバッテリ残量に着目したクラスタリング手法として，さまざまな手法が提案されている[4),5),7)]．ノードの移動特性やバッテリ残量等をクラスタリングに反映する方法として，ネットワークを頂点重み付きグラフとして扱う方法がある．Chatterjee らは，MANET を想定した頂点重み付きネットワークに対するクラスタリングアルゴリズムを提案した[4)]．この手法では，隣接ノード数がクラスタとして管理するのに適切な数であり，隣接ノードとの距離の総和が少なく，移動量が少なく，クラスタヘッドになっていた時間が短いノードに小さい重みを割り当て，重みが小さいノードほど，クラスタヘッドに適したノードとする．つまり，クラスタサイズとバッテリ容量，ノードの移動量からクラスタヘッドを選択する手法となっている．

Bazzal らは，クラスタの数や安定性，ネットワークのライフタイムに着目したクラスタリング手法を提案している[7)]．各ノードの重みは隣接ノードの数，ノードの平均速度に加えて，バッテリ残量，通信範囲を考慮して割り当てられる．つまり，より隣接ノードの多いノードをクラスタヘッドとすることで，形成されるクラスタの数を少なくしている．

Chinara らは，クラスタの形成にかかるオーバーヘッドの削減やネットワークのライフタイムの延長を目的としたクラスタリング手法を提案した[5)]．各ノードの重みは過去一定時間におけるノードの移動距離とバッテリ残量によって決まる．移動距離が少なく，バッテリ残量が多いノードほど大きい重みを割り当てられ，重みが大きいノードがクラスタヘッドとして選ばれる．クラスタヘッドのバッテリ残量が少なくなった場合，ネットワーク全体ではなく，クラスタ内で新たなクラスタヘッドを選出することにより，クラスタヘッド交代の影響がネットワーク全体へ拡大することを防いでいる．これにより，再計算にかかるオーバヘッドを削減している．

上記の手法はいずれもノードのモビリティ及びバッテリ残量に着目し，クラスタヘッドを決定するが，ノードの移動状況を考慮していない．考慮されるパラメータは隣接ノード数やノードの移動量が主であり，MANET では時々刻々と変化する．ノードのモビリティパターンをノードの重みに反映することによって，クラスタの安定性に関して，より優れたクラスタリングを行える可能性がある．

Johnen らは，MANET を想定した頂点重み付きネットワークに対する自己安定クラスタリングアルゴリズム（RCSA）を提案した[10)]．RSCA はトポロジ変化が起きた場合にも，クラスタに所属しないノードが存在しないという性質を保証するアルゴリズムである．クラスタヘッドはノードの重みに基づいて選択され，重みが大きいほどクラスタヘッドに適したノードとなる．しかし，ノードの重み割り当て方法については議論されていない．山内らは，重みが大きいノードの移動によって一時的なクラスタヘッドの交代が生じ，RSCA の

出力するクラスタ結果が不安定であることを指摘し，クラスタの安定性を改善する重み割り当て手法を提案した[11]．この手法では，ノード間の相対位置の変化から重みを計算し，相対位置の変化が少ないノードほど，大きな重みが割り当てられる．しかし，文献 4)，5)，7) で提案された手法と同じく，現実的なノードの移動状況を想定したものではないため，モビリティの考慮が不十分である．

2.2 MANET におけるノードのモビリティパターン

MANET におけるノードのモビリティを考慮する際，現実の人間や車などの動きを調査，再現するのではなく，それらを抽象化したモビリティパターンがよく用いられている[3]．MANET のモビリティモデルを大別すると，各ノードが独立に移動するモデルと，複数のノードがグループを作り，一群となって移動するモデルが存在する．複数のノードが伴って移動するモデルは一般的に**グループモビリティモデル**と呼ばれ，グループでノードが移動する状況を想定している．

各ノードが独立に移動するモビリティモデルとしてよく用いられているのが，ランダムウォークモデル，ランダムウェイポイントモデル，ランダムディレクションモデルである．**ランダムウォークモデル**は一定時間ごとにノードの移動速度と移動方向をランダムに決定するモデルである．**ランダムウェイポイントモデル** (RWP) では各ノードがランダムに決定した移動速度で，ランダムに決定した移動先へ移動する．移動先へ到着するとランダムな時間ノードは停止する．停止時間が過ぎると，次の移動先と移動速度をランダムに決定し，再び移動する．**ランダムディレクションモデル**では各ノードがランダムに選んだ移動速度と移動方向を，ノードが移動領域の端に到着するまで維持したまま移動する．これらのモデルは実現が比較的容易であり，モビリティモデルの中でも頻繁に用いられるモデルである．

グループモビリティモデルとしてよく利用されているのが *Reference Point Group Mobility* (RPGM) モデルである．RPGM では各グループに対して中央位置を仮想的に決め，中央位置の移動ベクトルとグループ内の各ノードの移動ベクトルが存在する．グループ全体の動きは中央位置の移動ベクトルによって決定し，各ノードは中央位置から一定距離内でランダムに移動する．グループモビリティモデルは他にもさまざまな種類が提案されている[3]．RPGM との主な相違点はグループの中央位置とグループ内の各ノードの位置関係であり，ノードが形成するグループの形によって適切なモデルを選択する．

3. システムモデルとノードへの重み割り当て問題

本章では，はじめに対象とする分散システムモデルについて述べ，その後，クラスタリング問題，ノードへの重み割り当て問題を定義し，提案手法の問題設定を与える．

3.1 システムモデル

分散システムをグラフ $G = \{V, E\}$ として表す．$V = (p_1, p_2, \cdots, p_n)$ はノードの集合であり，E はノード間の双方向通信リンクの集合である．ノード p_i と p_j に対して $(p_i, p_j) \in E$ のとき，p_i と p_j は**隣接する**という．ノード p_i の隣接するノードの集合を N_i と表す．ノード p_i はユニークな識別子 id_i と局所変数を持つ．以降では記述の便宜上，$id_i = p_i$ とし，p_i をノードの ID として扱う．各ノードは局所変数を管理しており，ノードの**状態**をすべての局所変数の値によって定義する．局所変数は入力変数，内部変数，出力変数から成る．分散システムへの入力は，入力変数として各ノードに与えられる．入力変数の値はシステム外部の要因によって変化する．内部変数はシステムが計算のために用いる一時変数，出力変数は外部のユーザに読まれる変数である．各ノードは自身の内部変数，出力変数を書き換えることができるが，入力変数を書き換えることはできない．

分散システムの通信モデルとして**局所的な共有メモリモデル**を用いる．各ノード p_i はノード $p_j \in N_i \cup \{p_i\}$ の局所変数を遅延なく直接読むことが可能だが，書き換えることができる局所変数は p_i 自身の局所変数のみである．アルゴリズムは $< Guard > \rightarrow < Action >$ の形式で表される**ガード付きアクション**の集合で記述される．p_i の *Guard* は p_i と N_i の局所変数に関する論理式となり，*Action* は p_i の局所変数の値を変更するステートメントである．*Guard* が true と評価されるとき，ノードは**動作可能である**といい，動作可能であるときのみ対応する *Action* を実行する．スケジューラとして *d*-**デーモン**を想定する．各計算ステップにおいて，*d*-デーモンは，動作可能であるノードの任意の空でない部分集合を選択し，選択されたノードは対応する *Action* を実行する．ガードの評価と対応するアクションはひとつの原子動作として実行される．また，スケジューラは**弱公平性**を持つと仮定する．弱公平性を持つスケジューラでは，無限にしばしば動作可能となるノードは *d*-デーモンによって必ず無限にしばしば選択される．

システムの状況を全ノードの状態の組として定義する．システムの実行は状況の無限系列 $e = c_0, c_1, c_2, \cdots$ で表記する．ここで c_{i+1} は c_i に 1 つの計算ステップを適用した結果，もしくは動作可能なノードが存在しない終端状況である．C をシステムがとりうる状況の集合とし，ε をシステムがとりうる実行の集合とする．

問題はすべてのノードの出力変数に対する論理式である．問題の仕様を満たす状況の集合は**正当な状況**と呼ばれる．問題 T に対するアルゴリズム $A(T)$ の正当な状況は状況に対する論理式 $L(A(T))$ で定義する．

定義 1：**自己安定**
アルゴリズム A は以下の 2 つの性質を満たす時，かつその時に限り**自己安定**であるという．
(1)　**収束性**：任意の状況から始まる実行に正当な状況が含まれる．
(2)　**閉包性**：正当な状況から始まるどのような実行も正当な状況のみから成る．

一時故障は（複数の）ノードの局所変数の値を任意の値に書き換える．これは，メモリのクラッシュなどに相当する故障である．自己安定アルゴリズムは任意の数の一時故障に対して耐性がある．つまり，一時故障が発生した後の状況を任意の初期状況と見なし，システムがやがて問題の仕様を満たすことを保証する．

システムへの入力に変化が起こった際にも，安全性に対する何らかの性質が保証されることが望ましい．ここで，入力の変化とは，ノードの入力変数の変化を意味している．また，安全性に対する性質を満たすための条件を**安全条件**と呼ぶ．Johnen らは入力の変化に対する頑健性を導入している[9]．

定義 2：**頑健性**
安全条件を SP，入力の変化の集合を IC とする．SP を満たす状況の集合 SC が以下の性質を満たす時，かつその時のみ，自己安定分散システム S が IC に対して**頑健性**を持つと言う．
(1)　SC が閉包性を持つ．
(2)　IC の下でも SC が閉包性を持つ．

クラスタリング問題はノードの集合を**クラスタヘッド**と**通常ノード**から成る**クラスタ**に分割する問題である．MANET では，各通常ノードが所属しているクラスタのクラスタヘッドと直接通信できる必要がある．そのため，以下の支配性をクラスタリング問題の制約とする．

定義 3：**クラスタリングの支配性**
各通常ノードは隣接ノードに少なくとも 1 つのクラスタヘッドを持つ．

3.2　ノードへの重み割り当て問題

本稿では，自己安定クラスタリングアルゴリズム RSCA[10] に対するノードの重み割り当て手法を提案する．RSCA は MANET のネットワークトポロジに対応するグラフと，グラフにおけるノードの重みを入力とし，与えられた重みに従ったクラスタ構造を出力するアルゴリズムである．したがって，ノードの重み割り当て問題の解（出力）は RSCA への入力である．

グラフはモビリティパターンに従って時間の経過とともに変化する．ノードへの重み割り当てとは変化したグラフに対する重みを逐次計算することである．つまり，**ノードへの重み割り当て問題**とは，MANET 環境を抽象化したグラフ G を入力とし，G に対するノードの重みを出力とする．本稿では特に，各ノードが局所的に自身の重みを計算する**重み割り当て関数**を持つとする．G において各ノード p_i は入力変数として自身の x 座標 x_i，y 座標 y_i，移動ベクトル $\overrightarrow{v_i}$，時刻 t を持つ．各状況 c_k において，ノード p_i の重み割り当て関数は，p_i と各 $p_j \in N_i$ の位置，速度ベクトル，状態を入力とし，ノード p_i の重み ω_i を出力する．

4. 提 案 手 法

RSCA は MANET に対して設計された自己安定クラスタリングアルゴリズムである．しかし，RSCA はノードの移動特性を考慮していないため，MANET におけるクラスタの安定性を保証していない．本稿では，ノードの移動特性を考慮した重み割り当て手法を提案する．4.1 節では RSCA の性質を述べ，4.2 節で提案する重み割り当て手法について述べる．

4.1　自己安定クラスタリングアルゴリズム RSCA

RSCA[10] は頂点重み付きネットワークに対して設計された自己安定クラスタリングアルゴリズムである．クラスタヘッドはノードの重みに基づいて選択され，重みが大きいノードほど，クラスタヘッドとして選択されやすい．

RSCA は，MANET におけるクラスタリングに対する性能を保証するために，文献 1) で定義された，以下の 3 つの性質を満たしている．

1. 各通常ノードは，自らの重みよりも大きな重みを持つクラスタヘッドに常に所属する．
2. 各通常ノード p_i，各クラスタヘッド $z \in N_i$ に対して，$\omega_z \leq \omega_{ch_{p_i}} + h$ が成り立つ（ノード p_i のクラスタヘッドを ch_{p_i} とする）．
3. クラスタヘッドは最大 k 個のクラスタヘッドと隣接する $(0 \leq k < n, k \in \mathbb{Z})$.

RSCA はトポロジ変化に対して頑健性を保証した自己安定アルゴリズムである．RSCA では各ノード p_i が T, F, NF という値を取る出力変数 $state_i$ を持つ．ノード p_i がクラスタヘッドであるとき，$state_i = T$ となり，p_i が通常ノードであるときは $state_i = F$，それ以外のとき $state_i = NF$ となる．ノード p_i のクラスタヘッドを ch_{p_i} としたとき，安全条件 SP を以下のように定義する．

$$SP \equiv \forall p_i \in V : (ch_{p_i} \in N_i \cup \{p_i\}) \wedge (state_{ch_{p_i}} \neq F)$$

RSCA では，クラスタヘッドが通常ノードに遷移するとき，一度 $state_i = NF$ という状態

をとってから通常ノードになる．$state_i = NF$ のとき，それまでそのノードをクラスタヘッドとしていた通常ノードは，他の新しいクラスタヘッドを隣接ノードから探し，ふさわしいノードがなければ自身がクラスタヘッドとなる．このように，$state_i = NF$ という中間的な状態をおくことにより，クラスタヘッドの再計算中も通常ノードが常にクラスタヘッド，もしくは $state_i = NF$ というノードに属していることを保証している．$state_i = NF$ であるノードは $state_i = T$ であるノードと同様にクラスタヘッドとして振る舞い，クラスタ内のノードを管理する．したがって，SP が満たされる時はクラスタリングによる階層構造が存在する．

文献 10) では以下のようなネットワークの動的な変化に対して，RSCA が SP を維持することを保証している．

(1)　ノードの重みの変更
(2)　通常ノードの一時故障
(3)　SP を満たす部分ネットワークの結合
(4)　2 つの通常ノード間，または 2 つのクラスタヘッド間のリンクの切断

しかし，クラスタヘッドの故障や，通常ノードとクラスタヘッド間のリンクの切断，また，頻繁なノードの重みの変更に対しては，RSCA が SP を満たすことは保証されていない．提案手法はノードの重みの変更を行うので，SP が満たされていることが保証される．

4.2 提 案 手 法

本節で提案手法 $GPGND$ について述べる．安定したクラスタを実現するために，以下の事項を考慮する．

- **グループの判定**： 各ノードはグループ内での相対的な位置関係の変化や一時停止を繰り返しているが，グループを形成して移動している期間はグループに属するノード同士でクラスタを形成し，維持することが望まれる．
- **グループの合流への対処**： 異なるグループが合流する場合，クラスタリングを再計算せず，クラスタを合併させる方が良い．その際，サイズの小さいクラスタがサイズの大きいクラスタに加わることで，クラスタ変更にかかるオーバヘッドを抑制できる．
- **グループの交差への対処**： 異なるグループが一時的に接近して交差する場合，クラスタ変更のオーバヘッドの観点から合併させない方が良い．このような場合には，グループごとのクラスタを維持する必要がある．

以上の検討事項を踏まえ，提案手法では，(i) グループに属するノードの判定，(ii) グループの移動軌跡，(iii) グループのサイズ，(iv) グループの移動性について考慮し，重み割り当てを行う．

グループの判定

ノード同士が同一のグループに属するか否かの基準として，ノード同士の位置が考えられる．しかし，現在のノード同士の位置が近くても，各ノードの移動ベクトルが大きく異なれば将来的にノード間の距離が大きくなることが予想できる．そこで，各ノードの移動ベクトルをグループの判定基準に用いる．定数 μ に対して以下の式 (1) を隣接するノード間で評価し，式 (1) が成り立つとき，ノード p_i は $p_j \in N_i$ が自身と同じグループに所属すると判定する．式 (1) はノードの距離が近く，移動ベクトルが近いノードを同一グループとして判定する．式 (1) によるグループの判定はネットワーク全体の情報を得てグループの ID 等を決定するといった方法ではなく，隣接ノードのうち，自身と同一グループに属するノードの判定のみを行う．

$$a_1 |x_i - x_j| + a_2 |y_i - y_j| + a_3 |\overrightarrow{v_i} - \overrightarrow{v_j}| < \mu \quad (1)$$

p_i が属するグループを以下 g_i とする．本稿では同一グループに属するノード同士は 1 ホップ内に存在し，直接通信が可能であると仮定する．

各ノードは式 (1) を用いて同一グループに属するノードを識別した後，グループ内のノードのモビリティをもとに自身の重みを計算する．重みを割り当てる要因としてグループの重心，グループのノード数，グループの移動ベクトルを考える．重み割り当て関数に関して，グループの重心を考慮した項を GPG，ノード数を考慮した項を GPN，移動ベクトルを考慮した項を GPD とする．GPG，GPD，GPN は，式 (1) において同一グループであると判定されたノード間でのみ計算する．

GPG

GPG はノード p_i の重みに，p_i が属するグループの重心の移動軌跡から，グループの判定の結果を重みに反映させ，ノードが交差する状況において，クラスタを維持させるための項である．グループの重心は，同一グループに属するノードの座標の平均値である．過去 T 秒間における重心の軌跡をグループ毎に計算し，各ノードが属しているグループの重心の軌跡とノードの移動軌跡を比較することで，移動軌跡がグループの重心の移動軌跡と近いノードほど重みが大きくなるように重みを割り当てる．

ノード p_i が属するグループの重心座標は $(x_{g_i}, y_{g_i}) = \dfrac{1}{|g_i|}\left(\sum_{p_j \in g_i} x_j, \sum_{p_j \in g_i} y_j\right)$ と定義される．

現時刻 t におけるノード p_i の重心との位置の差に応じた項 $GPG_i(t)$ は式 (2) となる．式 (2) において x_i^k は時刻 k におけるノード p_i の x 座標である．

$$GPG_i(t) = 1/\left(\sum_{k=t-T}^{t-1}\left(\left|x_i^k - x_{g_i}^k\right| + \left|y_i^k - y_{g_i}^k\right|\right) + 1\right) \tag{2}$$

GPN

GPN はノード p_i の重みに，p_i の属するグループのサイズを重みに反映させるための項である．想定した MANET 環境におけるノード数の上限を N とすると，ノード p_i のグループサイズに応じた項 $GPN_i(t)$ は式 (3) となる．式 (3) において $\left|g_i^t\right|$ は現時刻 t における，ノード p_i が属するグループ g_i のノード数である．$GPN_i(t)$ はグループのノード数が多いほど重みが大きくなる．

$$GPN_i(t) = \frac{\left|g_i^t\right|}{N} \tag{3}$$

GPD

GPD はノード p_i の重みに，p_i の属するグループの移動特性を重みに反映させるための項である．グループごとの移動ベクトルの平均値を求め，各ノードの移動ベクトルとの差に応じた重みを割り当てる．現時刻 t におけるノード p_i が属するグループ g_i の移動ベクトルの平均値 $E_{g_i}(t)$ は $E_{g_i}(t) = \dfrac{1}{|g_i^t|}\sum_{p_j \in g_i^t} \overrightarrow{v_j}$ となる．各ノード p_i に対して，p_i が属するグループの平均移動ベクトルとの差 $D_{\overrightarrow{v_i}}(t)$ は $D_{\overrightarrow{v_i}}(t) = E_{g_i}(t) - \overrightarrow{v_i}$ となり，これらの式からノード p_i の移動ベクトルに応じた重み要因 $GPD_i(t)$ は式 (4) となる．$GPD_i(t)$ は移動ベクトルがグループの平均の移動ベクトルに近いほど重みが大きくなる．

$$GPD_i(t) = \frac{1}{D_{\overrightarrow{v_i}}(t) + 1} \tag{4}$$

ノードの重み割り当て関数

上記の $GPG_i(t)$，$GPN_i(t)$，$GPD_i(t)$ からノード p_i の重みを式 (5) で定義する．

$$f(p_i) = b_1 \cdot GPG_i(t) + b_2 \cdot GPN_i(t) + b_3 \cdot GPD_i(t) \tag{5}$$

RSCA は各ノードの重みが互いに異なるネットワークに対して設計されたアルゴリズムである．式 (5) は異なるノード p_i，p_j で $f(p_i)$，$f(p_j)$ が同じ値となる場合がある．その場合，ノードの識別子によって比較を行う．つまり，ノード p_i，$p_j (i < j)$ に対して $f(p_i) = f(p_j)$ であるとき，ノード p_j の重みは p_i の重みよりも大きいと判断する．

5. シミュレーション実験

提案した重み割り当て手法の有効性を確認するために，Java でシミュレータを作成し，既存手法[11] との比較実験を行った．本章でははじめに 5.1 節でシミュレーション環境について示し，その後 5.2 節で実験結果について示す．

5.1 シミュレーション環境

評価実験では，ノードの移動状況としてグループを形成してノードが移動するモデルを想定している．そこで，提案手法が有効であると考えられる RPGM と，ノードのモビリティパターンとして代表的なモデルである RWP という 2 つのモビリティパターンに対して，いくつかの既存手法と比較実験を行った．シミュレーション領域は 1 辺が 200[m] である正方形領域とし，各ノードはこの領域内をモビリティパターンに従い移動する．RPGM に関して，グループ内の最大のノード数を 10 とした．シミュレーション全体に関するパラメータを表 1 に示す．グループの最大の直径は表 1 における通信範囲と同じ値となっており，グループ内の全ノードは互いに直接通信可能である．提案手法と関連手法は 2 つのモビリティパターンに対して，後に述べる評価指標ごとに各 100 回ずつシミュレーションを行い，その平均値を比較する．

表 1 シミュレーションに関するパラメータ

ノード数	500
シミュレーション時間 [s]	1800
最大速度 [m/s]	1
最大停止時間 [s]	60
通信範囲 [m]	10

表 2 重み割り当てに関するパラメータ

提案手法	a_1, a_2, a_3	1
	μ	10
	T	15
	N	500
	b_1	1
	b_2	5
	b_3	1
文献 11)	M	30
	$h(x)$	1
	T	15

比較手法として，毎秒各ノードにランダムに重みを割り当てる手法と，文献 11) で提案されている重み割り当て手法を用いる．文献 11) ではクラスタの安定性を目的とし，以下のようにノードの相対位置の変化に注目した 3 つの重み割り当て手法 Nei，$NumN$，$DifN$ をそれぞれ提案している．3 つの手法は時刻 t におけるノード p_i の隣接ノードの集合 $N_i(t)$

を用い，重みを割り当てる．Nei は最も単純な方法であり，隣接ノードの数を重みとして割り当てる．隣接ノードが多ければ多いほど重みは大きくなる．$NumN$，$DifN$ は時間ごとの隣接ノードの数の差，隣接ノードの集合差のサイズを重みとする．

Nei ： $f(p_i) = |N_i(t)|$

$NumN$ ： $Num_i(t) = \sum_{k=0}^{T-1} (h(T-k) \times (|N_i(t-k)| - |N_i(t-k-1)|))$ とし，
$f(p_i) = M/(Num_i(t)+1)$

$DifN$ ： $Dif_i(t) = \sum_{k=0}^{T-1} h(T-k) \times$
$(|N_i(t-k) \backslash N_i(t-k-1)| + |N_i(t-k-1) \backslash N_i(t-k)|)$ とし，
$f(p_i) = M/(Dif_i(t)+1)$

また，提案手法に関して，重み割り当て関数の 3 つの項 GPG，GPN，GPD がどのように結果に影響するか確認するために，それぞれの項を単独で重みに割り当てた場合の結果の比較も行う ($b_1 = 1, b_2 = b_3 = 0$ 等)．重み割り当てに関するパラメータを表 2 に示す．

本実験で用いた評価指標はクラスタヘッドの変化回数とクラスタヘッドの総数である．

クラスタヘッドの変化回数： クラスタの安定性を評価するための指標である．各ノードに関して，各時間ごとにクラスタヘッドから通常ノード，もしくはその逆の変化を起こした回数を計算する．クラスタヘッドが変化すれば，クラスタの再構築が行われ，システムのオーバヘッドが高くなる．そのため，クラスタヘッドの変化回数が少ない方が性能が良い．

クラスタヘッドの総数： クラスタリングの質を評価するための評価指標である．クラスタヘッドが極端に多ければクラスタヘッドの変化回数は小さくなる可能性はあるが，クラスタのサイズが小さくなり，クラスタリングの効果がない．そのため，クラスタヘッドの総数が少ないほど，よいクラスタリングと言える．

5.2 実 験 結 果

図 1 に RPGM モデルと RWP モデルにおけるクラスタヘッドの変化回数，図 2 にクラスタヘッドの総数を示す．図 1，図 2 の数値はいずれもシミュレーション時間中の各時間における値の平均値である．図において，GPG，GPN，GPD はそれぞれの項を単独で重みに割り当てた場合の結果であり，GPGND（提案手法）はこれらの 3 つの項を表 2 の b_1, b_2, b_3 の値にしたがって組み合わせた重みを割り当てた結果である．

まず，RPGM に関して提案手法と比較手法を比較する．クラスタヘッドの変化回数を見ると，提案手法は比較手法のいずれの手法に対しても小さい値となっている．比較手法のうち，最もクラスタヘッド変化回数の小さい $DifN$ と比較しても，提案手法はクラスタヘッドの変化回数が 34% 削減されており，クラスタの安定性に関して，提案手法は有効であると言える．クラスタヘッドの総数を見ると，ランダム，Nei，$DifN$ に対して提案手法はより小さな値であるが，$NumN$ に対しては 7% 大きい．しかし，提案手法のクラスタヘッド変化回数は，$NumN$ のクラスタヘッド変化回数より 54% 少なく，クラスタヘッドの総数の増加率に比べて，クラスタヘッドの安定性は大きく改善されている．2 つの評価指標について総合的に見ると，提案手法は比較手法におけるクラスタヘッドの総数を抑えつつ，クラスタヘッドの変化回数を大きく削減しており，RPGM モデルに対してクラスタの安定性を改善した手法である．

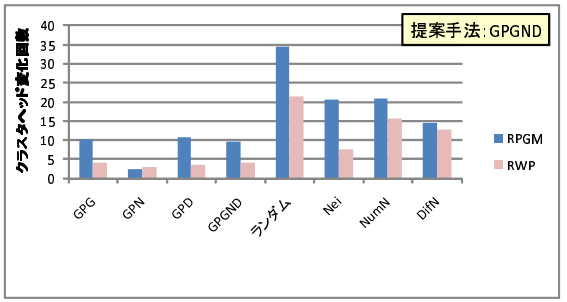

図 1 クラスタヘッド変化回数の比較

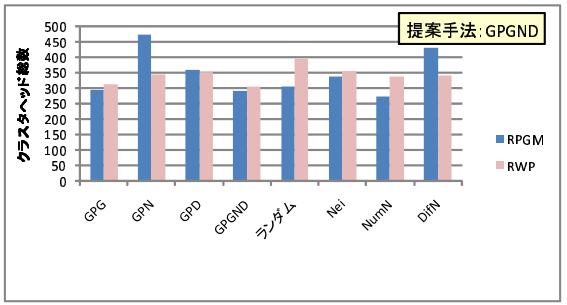

図 2 クラスタヘッド総数の比較

次に，RWP に関して提案手法と比較手法を比較する．クラスタヘッドの変化回数を見ると，比較手法のいずれの手法と比較しても，提案手法は小さい値となっている．比較手法のうち，最もクラスタヘッド変化回数の値が小さい Nei と比較しても，提案手法はクラスタヘッドの変化回数を 48% 削減しており，RPGM と同じく，クラスタの安定性に関して提案手法は有効である．クラスタヘッドの総数を見ると，提案手法はいずれの比較手法よりも小さな値となっている．比較手法のうち，最も値が小さい $NumN$ と比較しても，提案手法はクラスタヘッドの総数を 9% 削減しており，RWP モデルにおいては，クラスタリングの質に関しても提案手法は有効である．2 つの評価指標について総合的に見ると，提案手法はクラスタヘッドの変化回数とクラスタヘッドの総数のいずれの値も削減できており，RPW モデルに対してクラスタの安定性とクラスタリングの質を改善していると結論出来る．

さらに，重み割り当て関数の項である GPG，GPN，GPD それぞれを単独で重みに割り当てた場合における結果を比較する．RPGM モデルについて見ると，GPN はクラスタヘッド変化回数に関して最も小さい．しかし，クラスタヘッドの総数に関しては GPN の値が 480 ほどと最も大きく，ほとんどのノードがクラスタヘッドとなっている．そのため，RPGM モデルにおいて GPN はクラスタリングの質がかなり悪いものとなっている．クラスタヘッド総数の値は GPG が最も小さい．RWP モデルについても，RPGM モデルの場合と同じく，クラスタヘッドの変化回数に関しては GPN，クラスタヘッドの総数に関しては GPG が最も小さい．よって，クラスタヘッドの変化回数を優先したい場合は GPN，クラスタヘッドの総数を優先したい場合は GPG を重視すればよいと考えられる．

以上のことから，提案手法は比較手法と比較して，RPGM，RWP のいずれのモビリティパターンにおいてもクラスタリングの質を維持しつつ，クラスタの安定性を大きく改善することができている．また，重み割り当て関数に関する 3 つの項は，GPN がクラスタの安定性，GPG がクラスタリングの質に大きく影響する項であることが分かった．また，今回の実験はノード数を 500 としており，手法の適用例であるイベント会場としてはやや低い密度である．今後異なる密度における比較実験を行う予定である．

6. おわりに

本稿では MANET 環境においてクラスタの安定性を改善する自己安定クラスタリング手法を提案した．シミュレーション実験により，提案手法はクラスタヘッドの変化回数に関して，従来手法に比べて 34% 改善した．提案手法は比較手法よりもクラスタヘッドの総数が微かに大きくなる場合もあるが，クラスタリングの質を維持しつつ，クラスタの安定性を大きく改善することが確認できた．今後の課題としては，さまざまなシチュエーションを包括的に扱える重み割り当て手法を考案することが考えられる．

参考文献

1) Basagni, S.: Distributed and mobility-adaptive clustering for multimedia support in multihop wireless networks, *VTC'99: Proceedings of the IEEE 50th International Vehicular Technology Conference*, pp.889–893 (1999).
2) Bein, D., Datta, A.K., Jagganagari, C.R. and Villain, V.: A Self-stabilizing Link-Cluster Algorithm in Mobile Ad Hoc Network, *Proc. ISPAN'05*, pp.436–441 (2005).
3) Camp, T., Boleng, J. and Davies, V.: A Survey of Mobility Models for Ad Hoc Network Research, *WCMC : Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, Vol.2, pp.483–502 (2002).
4) Chatterjee, M., Das, S.K. and Turgut, D.: WCA: A Weighted Clustering Algorithm for Mobile Ad Hoc Networks, *Journal of Cluster Computing, Special Issue on Mobile Ad hoc Networking*, Vol.5, No.2, pp.193–204 (2002).
5) Chinara, S. and Rath, S.K.: TACA: A Topology Adaptive Clustering Algorithm For Mobile Ad Hoc Network, *Proc. WORLDCOMP'09* (2009).
6) Dijkstra, E. W.: Self-stabilizing system in spite of distributed control, *Commun. ACM*, Vol.17, No.11, pp.643–644 (1974).
7) El-Bazzal, Z., Kadoch, M., Agba, B.L., Gagnon, F. and Bennani, M.: An efficient management algorithm for clustering in mobile ad hoc network, *Proc. PM2HW2N '06*, pp.25–31 (2006).
8) Flauzac, O., Haggar, B. S. and Nolot, F. : Self-Stabilizing Clustering Algorithm for Ad Hoc Networks, *Proc. ICWMC'09*, Vol.0, pp.24–29 (2009).
9) Johnen, C. and Nguyen, L.H.: Route preserving stabilization, *Proc. SSS'2003*, pp. 184–198 (2003).
10) Johnen, C. and Nguyen, L.H.: Robust Self-Stabilizing Clustering Algorithm, *Proc. OPOSIS'06*, pp.410–424 (2006).
11) Yamauchi, Y., Itou, T., et al: Clustering algorithms for mobile ad-hoc networks to improve the stability of clusters, *Proc. IASTED'08*, pp.9–15 (2008).