M-062

無線端末位置推定法 Gomashio の改良

An improved method of Gomashio for wireless Ad-Hoc networks

武田 浩志　　岡崎 裕之
Hiroshi Takeda　　Hiroyuki Okazaki

信州大学大学院 工学系研究科 情報工学専攻

1 はじめに

近年，アドホックネットワークに関する試みが盛んにおこなわれている．

アドホックネットワークは，多くの無線端末がお互いに接続されることで構築されるネットワークである．無線端末は近くに存在する別の無線端末を自分で見つけ出し接続を行う．そのため，アドホックネットワークは無線端末を複数の場所に配置するだけで，特別にメンテナンスを行わなくとも利用できるネットワークである．ネットワークで使われる端末のほとんどは移動可能な端末であり，動的に位置が変化する．

アドホックネットワークの利用において位置情報を得る事が望ましい場合も多い．位置情報を得るには GPS を利用する方法が考えられるが，ハードウェアや環境の制限によりそれ以外の方法を用いなければならない場合がある．接続状況から位置情報を推定する簡易な方式は小型化や低コスト化，バッテリの長寿命化なども期待できるため，本研究では各端末同士の通信状況を用いた移動端末の位置推定を試みる．

本研究では，接続状況から移動端末の位置推定を行う方式の一つである Gomashio 方式 [1] を改良する．Gomashio 方式はいくつかの円の重なった範囲を推定結果として出力するという簡易な方法を使用している．本研究ではこの方法に変更を加え，より小さな推定結果を出力するよう改善を試みた．また，この改良案と Gomashio 方式の性能の差を調べるために実験を行った．その結果，得られる存在可能な範囲は Gomashio 方式より小さいものとなり，ある条件では推定結果を平均 70%程度の面積に抑える事が出来た．

2 位置推定の様々な方法

GPS を利用した方式以外に，位置推定に電界強度などの電波の特性を利用した方式や，それらを利用せずにネットワークの構成状況を用いる方法などある．これらは様々な方法が提案されているが，とくにネットワークの構成状況を使用した方式は DV-Hop 方式 [2] や Centroid 方式 [3]，Sextant 方式 [4]，Gomashio 方式などである．これらの方式では，多くの端末のうち，いくつかの固定局などが GPS などにより正確な位置情報を得ているとしている．

DV-Hop 方式は，固定局同士の距離とホップ数から，端末から各固定局への平均的な距離を求め，それを基に位置を推測するというものである．平均距離を使用しているため，精度はあまりあがらない．

Centroid 方式では，移動端末の位置を接続された複数の固定局の位置情報の重心をとることで，推定した位置とする．これは，位置推定を行いたい移動端末の周りに多くの固定局が存在する必要がある．

Sextant 方式や Gomashio 方式などの端末が存在するある一点を推定するのではなく，存在する領域を推定する方式もいくつかある．

Sextant 方式では，ベジエ曲線を用いて推定した結果を非凸領域として表現しており，データ効率や計算効率が良い．

Gomashio 方式では，各固定局を中心に固定局からのホップ数に比例して大きくなった半径の円を書き，重なった範囲を推定した結果とする．これは，単に円の重なった範囲を出力しているだけであるので，改善する余地がある．

3 想定するネットワークの詳細

今回想定するネットワークは，ある平面上に複数の端末が配置されており，一定距離以内の端末同士が接続されている．これらのうち，一部の端末は定点に設置されているものであるとする．また，今回我々が計算する対象のネットワークは，動的なものではなくてある瞬間のネットワークの状態であるとする．

ある瞬間のアドホックネットワーク全体の構成を無向グラフの様に見立てる．各端末や各固定基地局はノード (頂点) であり，また，ノード同士の接続はエッジ (辺) である．各ノードや固定ノードをどのような記号で表すかを以下に示す．

1. ネットワーク全体をグラフ $G = (V, E)$ とする．
2. 各ノード $v \in V$ は二次元の座標 (x_v, y_v) に存在する．
3. 二つのノードの距離が r 以内であれば，これらは接続されている．すなわち，あるノード $v_1 \in V$ とあるノード $v_2 \in V$ について，$(x_{v_1} - x_{v_2})^2 + (y_{v_1} - y_{v_2})^2 \leq r^2$ ならば，v_1 と v_2 は隣接している．
4. すべての固定ノード (固定基地局) の集合を $V_s (V_s \subset V)$ とする．

これらの情報すべてを位置推定に使用するわけではない．今回使用する情報は以下のようなもののみである．

1. 固定端末の座標情報 $((x_v, y_v)(v \in V_s))$
2. ノードの存在 (V)
3. 端末接続状況 (E)

4 Gomashio 方式の詳細

Gomashio 方式とは，ノードごとに存在可能な範囲 (エリア) を求める方式である．範囲を求める際に各固定ノードとの距離 (最小ホップ数, 最短経路長) を利用する．

$v \in V$ の存在可能な範囲を求めるには，次のような方法を取る．

v に連結している複数の固定ノード $v_s \in V_s$ について，各々の距離 $(d(v, v_s))$ を使い $(x - x_{v_s})^2 + (y - y_{v_s})^2 \leq (r \times d(v, v_s))^2$ の様な円形の範囲を求める．これをすべて満たすような範囲を，各々の固定ノードにより制限された v の存在可能な範囲とする．

複数の固定ノードにより制限を求めた例を，図 1 に示す．固定ノード $v_b, v_{b'} \in V_s$ があり，$d(v_a, v_b) = 2, d(v_a, v_{b'}) = 1$ であったとする．このような場合は，v_b を中心とした円と $v_{b'}$ を中心とした円の重なる範囲が，v_a の存在可能な範囲である．

5 改良した方法の提案

4 章では調べたいノードの存在可能な範囲を，固定ノードとの距離から複数の円の範囲を使って求めた．しかし，本来は存在しえないと言える範囲まで，存在可能な範囲として求めてしまう事がある．

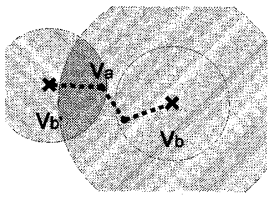

図 1 Gomashio 方式

そのような例を図 2 に示した．この例では，$v_{a'}$ の存在可能な範囲を灰色で示している．$v_{a'}$ の存在可能な範囲は v_a により制限されなければならないが，Gomashio 方式ではこの様な点は考慮されない．

この改良案として，固定ノード以外のノードの情報を利用し，求める存在可能な範囲をさらに狭く制限する方法を提案する．($A(v)$ をある $v \in V$ の存在可能な範囲として表記する．)

改良案では，単に固定ノードでない v_a を中心とした座標に半径 r の円を書き，これを $A(v_{a'})$ とすることはできない．そこで，円の方程式の代替として，ミンコフスキー和を使用する．

ミンコフスキー和 $A \oplus B$ とは，次のような位置ベクトルの集合として表される．二つの図形 A, B があり，これらは位置ベクトルの集合である．その時，A と B のミンコフスキー和は次のように表される．

$$A \oplus B := \{\boldsymbol{a} + \boldsymbol{b} \mid \boldsymbol{a} \in A, \boldsymbol{b} \in B\}$$

$A(v_{a'})$ を求めたい時，$v_{a'}$ に隣接するノード v_a の存在可能な範囲 $A(v_a)$ と半径 r の円 (C) のミンコフスキー和を求めると，$v_{a'}$ の存在可能な範囲を得られる．

$$\begin{aligned} C &:= \{\boldsymbol{a} \mid \|\boldsymbol{a}\| \leq r\} \quad \text{半径 } r \text{ の円} \\ A(v_{a'}) &= A(v_a) \oplus C \\ &= (\{(x_{v_{s2}}, y_{v_{s2}})\} \oplus C) \cap \\ &\quad (\{(x_{v_{s1}}, y_{v_{s1}})\} \oplus C)) \oplus C \end{aligned}$$

この式を観察するとわかる通り，各固定ノードに近いノードから順に存在可能な範囲を求め，最終的に目的のノードの存在可能な範囲を得ている．

また，実際にミンコフスキー和を使用して得られた $A(v_{a'})$ を，図 3 に示す．

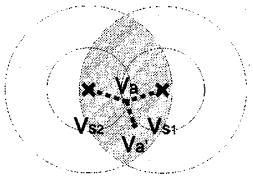

図 2 Gomashio 方式

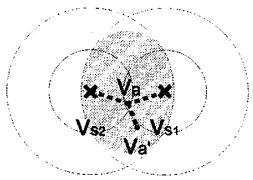

図 3 改良案

5.1 Gomashio 方式との比較

Gomashio 方式と異なる方法で存在可能な範囲を求めたが，これは実際にどのようなグラフで結果に違いが出るのかを考える．あるノード $v \in V$ について $A(v)$ を求めたい時，v と v に連結する各固定ノード $(v_{s1}, v_{s2}, \ldots \in V_s)$ の各々の最短経路上に共通するノード $(v' \in V)$ があると，Gomashio 方式と改良案で異なる結果が得られる．

たとえば，図 3 では，$v_{a'}$ から各固定ノードへの最短経路は $\{v_{a'}, v_a, v_{s1}\}$ と $\{v_{a'}, v_a, v_{s2}\}$ である．この二つの経路には，共通して v_a が存在しているが，これにより $v_{a'}$ の存在可能な範囲は Gomashio 方式と改良案で異なってくると言える．これは，$v_{a'}$ の存在可能な範囲を求める際に，Gomashio 方式では v_a のような，$v_{a'}$ と各固定ノードの最短経路上のノードの存在可能な範囲を考慮しないため，改良案は $A(v_a)$ が小さな範囲になる事を考慮に入れている改良案とは異なる結果となるのである．

6 アルゴリズム

改良案を計算機上で実行するためのアルゴリズムを考える．

存在可能な範囲を求めたいノード $v \in V$ がある時，$A(v)$ を求めるためには，v に連結する各々の固定ノード $(v_{s1}, v_{s2}, \ldots \in V_s)$ に近いものから順に存在可能な範囲を決めていく．また，この計算は各々の固定ノードから v への最短経路上にて行われる．この計算は，ネットワーク構成 G を幅優先探索 [5] し，v を根としたグラフ $(G' := BFS(G))$ を得ることで簡単に考える事が出来る．

得られた G' であるが，v と各固定ノードは，G における最短経路長と同じ経路長で結ばれている．この事から，各固定ノードを起点とし，根に向かって各ノードを順に存在可能な範囲を求める事で計算を行える事がわかる．各ノードでは，子ノードの存在可能な範囲と C のミンコフスキー和をとる事と，それら全ての積集合をとることで，同ノードの存在可能な範囲を求める事が出来る．

このような操作を，表 1 に示した．これは，存在可能な範囲を求めたいノードを引数とし，入力する．入力されたノードを根とした BFS を考え，根から遠い固定ノードから順に存在可能な範囲を求めるというものである．

6.1 隣接していない固定ノードの除外

より小さな面積を出力する方法として，隣接していない固定ノードの存在可能な範囲の除外を考える．

Gomashio 方式 [1] で言及されているが，あるノード $v \in V$ の存在可能な範囲を求める際に，ある固定ノード $v_s \in V_s$ に v が隣接していない場合，v_s を中心とした半

表 1 $Area(root)$

```
1: L[0] ← root
2: mark root as visited
3: // 固定ノードより深く調べない
4: for v_s ∈ V_s do
5:   mark v_s as visited // 走査不要であるとフラグを指定
6:   A(v_s) ← {(x_{v_s}, y_{v_s})} // 存在範囲を点とする
7: end for
8: i ← 0
9: j ← 1
10: // root から距離が小さい順にノードを並べる.
11: repeat
12:   // 隣接したノードを辿る
13:   for v_c ∈ L[i].adjacent() do
14:     if v_c is not marked as visited then
15:       L[j] ← v_c
16:       mark v_c as visited
17:       j ← j + 1
18:     end if
19:   end for
20:   A(L[i]) ← U // 初期値として A(L[i]) を全域とする
21:   i ← i + 1
22: until i = j
23: // root から遠い順に存在可能な範囲を求める
24: repeat
25:   i ← i − 1
26:   // 隣接するノードにより範囲を狭める
27:   for v_c ∈ L[i].adjacent() do
28:     if v_c is marked as visited then
29:       A(L[i]) ← A(L[i]) ∩ (A(v_c) ⊕ C)
30:     end if
31:   end for
32: until i = 0
33: return A(root)
```

径 r の範囲は v の存在可能な範囲として除外することができる．これは改良案にも適用する事が出来る．

改良案にてこれを行うと，先に考えた様な Gomashio 方式と改良案で同じ結果となると言える条件下でも，異なる結果となり，さらに小さな範囲を得る事が出来る．例えば，ある $v \in V$ について $A(v)$ を求めたい時，v と固定ノードへの最短経路上に存在する別のノード $v' \in V$ について，$A(v')$ が v' と隣接していない固定ノードの範囲を除去しているとする．改良案では $A(v')$ は $A(v)$ に影響し，より小さな結果を得る．Gomashio 方式では v の存在可能な範囲を求める際にはこのような v' があったとしても，v' を考慮しないため小さな範囲を得にくい．

7 実行例

6 章で示したアルゴリズムを基に，GUI を利用したグラフの作成と存在可能な範囲の表示を行うプログラムを作成した．なお，プログラミング言語として C++を，GUI の表示に SDL を使用した．GUI 上においてネットワークを構成する任意のノードを選択し，このノードの存在可能な範囲を適当な解像度のラスタ画像として表示する事が出来る．

このプログラムにより，実際に画面上で配置した複数のノードに対して Gomashio 方式と改良案のそれぞれでどのような範囲を出力したかを確認する．

7.1 単純な配置のグラフ

図 2 に似た配置のグラフを作成した．(図 4 から図 9)

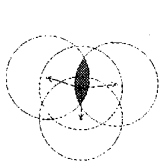

図 4 v_a(Gomashio 方式)

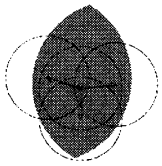

図 5 $v_{a'}$(Gomashio 方式)

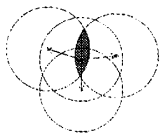

図 6 v_a(改良案)

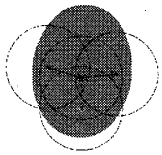

図 7 $v_{a'}$(改良案)

$v_{a'}$ の存在可能な範囲は Gomashio 方式と改良案で異なった結果を得て，v_a では同じ結果を得ている．これについては，ノードの幅優先探索のグラフ (図 8 と図 9) を見ると簡単に確認することができる．図 9 では，$v_{a'}$ から各固定ノード (v_{s1}, v_{s2}) までの最短経路上に共有のノード v_a があるため，Gomashio 方式と改良案で異なった結果を得ている．

8 性能

Gomashio 方式と改善案で出力する面積の差について調査する．

同じ位置のノード・固定ノードにて Gomashio 方式と改良案をそれぞれの方法を用い，隣接していない固定ノードの除外の有り無しの両方で面積を求めた．これは複数回行われ，ノードの場所は毎回ランダムに決定される．

図 10 と図 11 は，横軸を Gomashio 方式で得た面積，縦軸を改良案で得た面積としてプロットしたものである．試行回数は 22 回である．図中，×印は固定ノードの及ぶ範囲を除去する Gomashio 方式と改良案の比較，＋印は固定ノードの及ぶ範囲を除外しない Gomashio 方式と改良案の比較となっている．

各点の位置が $y = x$ の直線から離れているほど，改良案は Gomashio 方式より小さな範囲を出す事が出来ている．なお，改良案は Gomashio 方式より大きな面積を出さないため，必ず各点は $y = x$ の直線より低い位置にプロットされている．

図 10 は固定ノードを除外する Gomashio 方式と改良

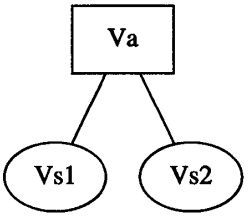

図 8 v_a が根

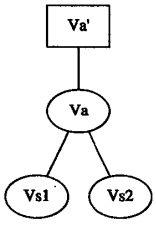

図 9 $v_{a'}$ が根

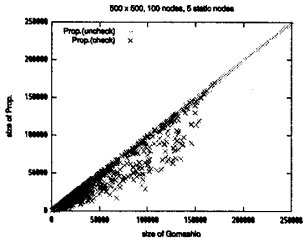

図 10　差が大きい

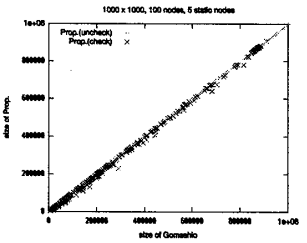

図 11　差が小さい

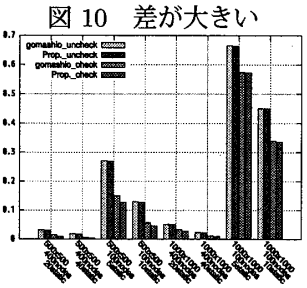

図 12　面積

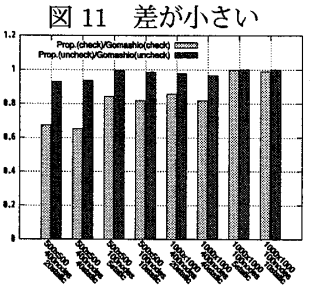

図 13　面積比

案は比較的大きな面積差を得た．それに比べて，図 11 は固定ノードの及ぶ範囲を除外する場合と除外しない場合のどちらも大きなさが見られない．これら二つはグラフを構成するノード数や固定ノード数は変わらない．しかし，エリアの面積が 500x500 と 1000x1000 と異なっており，エリアの面積が小さい方が存在可能な範囲の面積差が大きい．

図 10 のようなノードが密集いる場合，Gomashio 方式と改良案で異なる結果を出力するという条件を満たすノード $v_a \in V$ が多いようである．また，v_a に隣接しない固定ノードが除外された $A(v_a)$ は，v_a に隣接した他のノード $v_b \in V$ の $A(v_b)$ に影響を与える．そのため，v_b に連結しているノード v_b は，v_a の面積の変化が与えられ，$A(v_b)$ の面積を小さくしている．v_b のような間接的に存在可能な範囲の面積を小さくする事が出来るノードが多く存在する構成ほど，改良案は Gomashio 方式に比べて小さな存在可能な範囲を出力する事が出来る．

図 13 の「Prop.(check)/Gomashio(check)」は，隣接しない存在可能な範囲を除外する場合における Gomashio 方式と改良案の面積比である．エリア 500x500・ノード数 400 では 70%程度，エリア 500x500・ノード数 100 では 80%程度の面積比となった．ノードが密集しているような状況ほど，改良案は Gomashio 方式に比べて小さな存在可能な範囲を得られる事がわかる．

9　おわりに

本研究では，Gomashio 方式の改良を試みた．その結果，ミンコフスキー和を用いることで，Gomashio 方式を改良することに成功した．

今回実験したところによると，Gomashio 方式と比較して改善案では出力した存在可能な範囲の面積は最大で 70% 程度に抑えることに成功した．これは，固定ノード数が多いほど見られるものであるので，隣接していない固定ノードの範囲を除外することが大きく作用したようである．また，全体のノード数が増えるにつれてこの傾向は大きくなるようである．ノード数が同じで，かつエリアの面積が異なる場合，エリアの面積が小さく密集している方がより面積を小さくできることも分かった．このような条件下でなくとも，Gomashio 方式に比べて小さな範囲を出力するために方法がないかを今後見つけていきたい．

改良案では隣接しない固定ノードに隣接していない場合はその領域を除外する処理を行う事が出来た．そのようなノードに連結されたノードの存在可能な範囲は影響を受けて存在可能な範囲が小さくなる．これは，単純に円の重なった範囲から存在可能な範囲を求めている Gomashio 方式では出来ないことである．

Gomashio 方式や改良案は，実地での実験を行っていない．アドホックネットワークは実際には今回のシュミレーションプログラムを作成したような均一な条件下で良好な接続が行えるとは限らない．このような問題に対応することは，実際に実地で使用するにあたり必ず必要な事であると考えられる．また，今回はある瞬間のネットワークの状態を保存し，それに対して処理を行うという方式であった．今後は複数の時間に計測し，ノードの移動状況の確認やさらなる精度の向上または不完全なネットワーク情報を得た場合の補正などがさらに追加できる機能であると考えられる．

参考文献

[1] 岩谷晶子, 西尾信彦, 村瀬正名, 徳田英幸:
ごましお：アドホックセンサネットワークにおけるノード位置決定法式,
http://ci.nii.ac.jp/naid/110002937796/

[2] Dragos Niculescu, Badri Nath:
Ad Hoc Positioning System (APS),
http://www.cs.rutgers.edu/dataman/FourierNet/aps.html

[3] Tomoyuki Tarumi:
重心法 {centroid method},
http://case.f7.ems.okayama-u.ac.jp/statedu/hbw2-book/node122.html

[4] S. Guha, R. N. Murty, E. G. Sirer:
Sextant: A unified node and event localization framework using non-convex constraints
http://doi.acm.org/10.1145/1062689.1062715

[5] T. コルメン, C. ライザーソン, R. リベスト C. シュタイン:
アルゴリズム イントロダクション [第一巻][第二巻]