最短経路問題の性質とその高速探索アルゴリズム

栗野俊一[†1] 高橋俊雄[†2] 吉開範章[†1]

単連結な双方向グラフにおいて，特定のノードを終点とし，グラフ内の任意のノードから，その終点ノードへ最短経路からなる部分グラフを，その終点ノードへ最短経路網と呼ぶ．この計算には，ダイクストラの SSSP アルゴリズムが知られている．

この論文では，この最短経路網の性質を幾つか示し，それを用いることによって，すでに計算済みの，あるノードへの最短経路網から，そのノードに隣接している他のノードへの最短経路網を求めるアルゴリズムを提案する．このアルゴリズムは，単に直接 SSSP を用いて目的の最短経路網求めるより高速であることを示す．

Attributes of Shortest Path Network and the Fast Algorithm for Betweenness Centrality

Syun-ichi Kurino,[†1] Toshio Takahashi [†2] and Noriaki Yoshikai[†1]

In connected undirected graph $G =< N, V >$, we call G_o as "shortest path graph of o" which is collection of shortest paths to o form any node n in G. Dijkstra's SSSP algorithm can make a G_o in $O(|V|)$.

In this paper, we show some attributes in shortest path graph, and introduce a new algorithm which make G_s from G_o, where s is neighbor node of o. And show our new algorithm is faster then Dijkstra's algorithm.

1. まえがき

オンライン・コミュニティのようなネット上の組織活動は，ネットワークを基盤に展開される社会的（ソーシャル）ネットワークと考えることができる．これらを「アフィリエーション・ネットワーク」[1] としてグラフ理論に基くモデル化を行い，そのグラフ上の様々な特性値を計算することによって，その社会活動における個人や社会に関する様々な情報を得ることができる．例えば，ノード間の仲介機能を評価する媒介中心性は，グラフ内におけるノードの特徴を表す重要な指標の一つで，大学や企業内の人的ネットワークの可視化や、その情報を利用した情報ネットワークの制御等の様々な応用が知られている[2),3)]．

しかし，このような特性値に基く評価では，ネットワークの規模の拡大に伴う，それらの特性値の計算時間の増大が，分析の上での大きな課題となる．

例えば，媒介中心性は，グラフのノード数 n とリンク数 m に対して，$O(n \times m)$ の計算量を必要とする[4)]．

これは，媒介中心性の計算のために，グラフの全てのノード n に対して，それらを終端とする最短経路網を計算する必要があり，その最短経路網の計算に利用されるダイクストラのアルゴリズム[5)] が $O(m)$ の計算量を持つからである．

ところが，この従来の方法では，個々のノードに対する最短経路網を求める際に，他のノードに対する最短経路網の情報があったとしても，これを利用していない．その一方，隣接したノードに対する最短経路網は，互いに共通な部分が多く含まれている．したがって，もし高速に，新しいノードに対する最短経路網と，既に計算済みの最短経路網との間の共通部分が発見でき，変更が必要な部分だけを書き換えることによって，新しい最短経路網を作ることができるのであれば，これにより高速に最短経路網を求め，より短時間で媒介中心性を計算することができる．

そこで，本論文では，新たなノードの最短経路網を計算する際に，そのノードに隣接するノードの最短経路網の情報を利用すれば，単にそのノードに対してダイクストラのアルゴリズムを適用した場合より，高速に計算できる場合があることを示し，この性質を利用した最短経路網の高速探索アルゴリズムの提案と，その性能評価を行う．

2. 準　　備

（定義）グラフ：ノード集合 N と，順序対の集合 $N \to N = \{< i,j > | i,j \in N, i \neq j\}$ の部分集合であるリンクの集合 $V \subset N \to N$ との対 $G =< N, V >$ をグラフと呼ぶ．ここでは，$< i,j >$ と $< j,i >$ は区別し（有向グラフ），i,j をそれぞれリンク $< i,j >$ の始点，終点と呼ぶ．また，$i \in N, < i,j > \in V$ となることを，それぞれ $i \in G, < i,j > \in G$ と省略する．

†1 日本大学理工学部数学科
Nihon University, College of Science & Technology , Department of Mathematics
†2 （独）雇用・能力開発機構
Employment and Human Resources Development Organization of Japan

（定義） 無向グラフ：グラフ $G =< N, V >$ において，$< i, j > \in V \rightarrow < j, i > \in V$ が成立する場合，このグラフ G は，無向グラフであるという．

（定義） 部分グラフ：グラフ $G =< N, V >$ に対して，グラフ $G' =< N', V' >$ が，$N' \subset N \wedge V' \subset V$ を満す場合，このグラフ G' を G の部分グラフと呼び $G' \subset G$ と表す．

（定義） 経路：$i_0, i_1, .., i_k \in G$ の時，$\{< i_0, i_1 >, < i_1, i_2 >, .., < i_{k-1}, i_k >\}$ を，i_0 から i_k への経路と呼び，$path_G(i_0, i_1, .., i_k)$ で表す．この時，i_0, i_k を，それぞれ経路の始点，終点と呼ぶ．特に，経路の途中の詳細が問題とならない場合や，文脈から G が明確な場合はそれらを省略し，単に $path(i, .., j)$ や，$path(i, j)$ などと略記する．また，i から j への経路 $p = path(i, j)$ が，グラフ $G =< N, V >$ の中に存在する場合，すなわち $p \subset V$ である場合，$p \in G$ と表す．

（定義） 経路の接続：p_0 の終点と p_1 の始点が等しい場合，$p_0 \cup p_1$ も経路となる．これを経路の接続と呼び，$p_0 + p_1$ で表す．

（定義） 単連結：グラフ G の任意の二つのノード i, j に対し，必ず，i から j への 経路 $path(i, j)$ が G の中に存在する時，すなわち $\forall i, j \in G[\exists path(i, j) \in G]$ の時，そのグラフ G は，単連結であると言う．

（定義） 経路の長さ：i_0 から i_k の経路 $p = path(i_0, i_1, .., i_k)$ に対し，k を，その経路の長さと呼び，$|p|$ で表す．

（定義） 経路の延長と短縮：j から k への経路 $p = path(j, .., k)$ に対して，先頭に新しいノード i を加えた $path(i, j, .., k)$ は，i から k への経路になっている．このように既存の経路から，新しい経路を作る操作を，経路の延長と呼び，$i + p$ （始点側に追加した場合）あるいは $p + i$ （終点側に追加した場合）で表す．また，その逆の操作を短縮と呼ぶ．

（定義） 距離：単連結な G の二つのノード i, j において，G に含まれる i から j への経路 $path(i, j)$ の長さの最小値を，G での i, j 間の距離と呼び，$d_G(i, j)$ で表す．すなわち $d_G(i, j) = \min_{p=path(i,j)\in G} |p|$ である．G が明かな場合は，それを省略して $d(i, j)$ と表す．

3. 最短経路網の性質

（定義） 最短経路集合：G の二つのノード i, j に対し，G の中にある i から j への経路 $path(i, j) \in G$ の中で，最も長さの短い経路の集合を最短経路集合と呼び $spath_G(i, j)$ で表す．G が単連結であれば $spath_G(i, j)$ は空にならない．また，この時，次が成立する．

$$path(i, j, .., k) \in spath_G(i, k) \longrightarrow path(j, .., k) \in spath_G(j, k)$$

（定義） 最短経路網：無向グラフ $G =< N, V >$ とノード o に対して，$V' = \bigcup_{i \in G} spath_G(i, o)$ とする．この時，G の部分グラフ $G_o =< N, V' >$ を o を終点とする最短経路網と呼ぶ．

（命題） リンク $< i, j >$ が最短経路網 G_o に含まれていれば，i から o への距離 $(d(i, o))$ は，j からの距離 $(d(j, o))$ $+1$ となる．

$$< i, j > \in G_o \longrightarrow d(i, o) = d(j, o) + 1$$

（証明） 最短経路網 G_o の定義から，$< i, j > \in G_o$ であれば $\exists path(i, j, .., o) \in spath(i, o) \subset G_o$ となる．すると，これを短縮した $path(j, .., o)$ は $spath(j, o)$ に入るので，やはり，G_o に含まれる．よって，$d(i, o) = |path(i, j, .., o)| = |path(j, .., o)| + 1 = d(j, o) + 1$ となる．

（命題） G_o において，リンク $< i, j >$ が元のグラフ G に含まれていれば，i, j から o への，それぞれの距離 $d(i, o), d(j, o)$ の間には，次のような関係がある．

$$< i, j > \in G \longrightarrow d(j, o) - 1 \leq d(i, o) \leq d(j, o) + 1$$

特に，距離の関係と，リンク $< i, j >$ の G_o への所属関係は，次の三つのどれかになる．

$$\begin{aligned} d(i, o) = d(j, o) - 1 &\leftrightarrow < i, j > \notin G_o \wedge < j, i > \in G_o \\ d(i, o) = d(j, o) &\leftrightarrow < i, j > \notin G_o \wedge < j, i > \notin G_o \\ d(i, o) = d(j, o) + 1 &\leftrightarrow < i, j > \in G_o \wedge < j, i > \notin G_o \end{aligned}$$

（証明） 仮定より $< i, j > \in G$ なので，$|path(i, j, .., o)| = |path(j, .., o)| + 1$．よって，$d(i, o) \leq d(j, o) + 1$．$i$ と j を入れ替えることによって，$d(j, o) \leq d(i, o) + 1$．この二つをまとめると $d(j, o) - 1 \leq d(i, o) \leq d(j, o) + 1$ となる．

（定義） 上流：最短経路網 G_o において，G の任意のノード i から o への経路の内，s を経由する経路をだけを考える．この経路から，s より後の部分を取り除いた（結果，その経路の終点は s となる）経路からなる部分グラフを，s の上流と呼び $U_o(s)$ で表す．

$$U_o(s) = \bigcup_{p=path(i,..,s)\in G_o} p$$

（命題） s を o の隣接ノードとする．G_o における s の上流 $U_o(s)$ を考えると，それは G_s に含まれる．

（証明） $p = path_G(i, .., s) \in U_o(s)$ を考える．すると，$U_o(s)$ の作り方により，$p + o \in G_o$ であり，これは，i, o 間の最短経路である．したがって，p は，i, s 間の最短経路となるので $p \in G_s$ となる．

（定理 1） 変更されないノード： ノード o, s が隣接しているのであれば，$U_o(s) \cup U_s(o) \subset G_o \cap G_s$ である．

（証明） $U_o(s) \subset G_o, U_o(s) \subset G_s, U_s(o) \subset G_o, U_s(o) \subset G_s$ より $U_o(s) \cup U_s(o) \subset G_o \cap G_s$

（定義） 境界： $U_o(s)$ にも $U_s(o)$ にも含まれないノードの集合（このノードに接続しているリンクは G_o と G_s で異なる可能性がある）を $U_o(s)$ と $U_s(o)$ の境界と呼び E_{os} で表す．

（命題） $U_o(s), E_{os}, U_s(o)$ と距離の関係は以下のようになっている．

$$\begin{aligned} n \in U_o(s) &\leftrightarrow d(s,n) = d(o,n) - 1 \\ n \in E_{os} &\leftrightarrow d(s,n) = d(o,n) \\ n \in U_s(o) &\leftrightarrow d(s,n) = d(o,n) + 1 \end{aligned}$$

（証明） 一つ目と三つ目は，定義より明らか，二つ目は仮定より，$< o, s > \in G$ であるので，$d(n,o) - 1 \leq d(n,s) \leq d(n,o) + 1$ であるが，$d(n,o) - 1 = d(n,s)$ すなわち $d(o,n) - 1 = d(s,n)$ であれば $n \in U_o(s)$ となり，$d(n,o) - 1 = d(n,s)$ すなわち $d(o,n) - 1 = d(s,n)$ であれば $n \in U_s(o)$ となるので，結局 $n \in E_{os}$ となる必要十分条件は $d(s,n) = d(o,n)$ となることが解る．

（定理 2） $U_o(s)$ と E_{os} の推移性：

$$\begin{aligned} n \in U_o(s) &\rightarrow U_o(n) \subset U_o(s) \\ n \in E_{os} &\rightarrow U_o(n) \subset E_{os} \cup U_o(s) \end{aligned}$$

（証明） 一つ目は定義より明らか，二つ目は，$m \in U_o(n)$ とすると，定義より $\exists path(m,..,n,..,o) \in G_o$ である．一方，$n \in E_{os}$ なので，$\exists path(n,..,s) \wedge |path(n,..,s)| = |path(n,..,o)|$ となる．すなわち，$\exists path(m,..,n,..,s) \in G_o \wedge |path(m,..,n,..,s)| = |path(m,..,n,..,o)|$ となる．この $path(m,..,n,..,s)$ が最短経路であるかはわからないが，この経路があるので，少くても $d(s,m) \leq d(o,m)$ となる．したがって，$m \in E_{os}$ （等式が成立する場合）または，$m \in U_o(s)$ （不等式が成立する場合）である．

（定理 3） E_{os} の要素は，$U_o(s)$ の要素を経由して s に到達する：

$$n \in E_{os} \rightarrow \exists i \in U_o(s)[\exists n' \in E_{os}[< i, n' > \in G \wedge n \in U_o(n')]]$$

（証明） 背理法によって示す．結論を否定すると，$\forall n'[\exists i \in U_o(s)[< n', i > \in G] \rightarrow n \notin U_o(n')]$ となる．これは，n から o へ最短経路 $p = path(n,..,o)$ には，$U_o(s)$ と隣接するノードは含まれないことを意味する．したがって，n から s への最短経路は，$p + s$ となり，この長さは p より大きいので，$n \in E_{os}$ に矛盾．

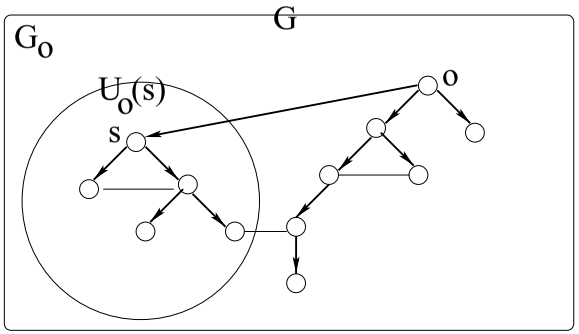

図 1 G_o と $U_o(s)$

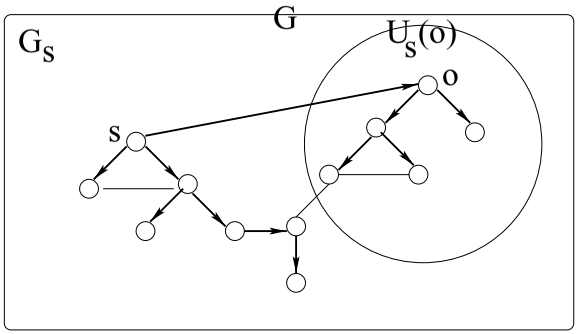

図 2 G_s と $U_s(o)$

4. 最短経路網の探索法

4.1 終点の変更による最短経路網の変化

すでに，G_o （**図 1**）が計算済みとして，これから，ノード o に隣接するノード s を終点とする G_s （**図 2**）を構成することを考える．

定理 1 から $U_o(s) \cup U_s(o) \subset G_o \cap G_s$ なので，少なくとも $U_o(s) \cup U_s(o)$ の部分は変化しない．そこで，それ以外の変化する部分を探し，その部分を変更すれば，G_o から，高速に G_s を作成できる可能性がある．

この例は，図 1 と図 2 を比較する事によって，**図 3** のように三ヶ所のリンクを書換えるだ

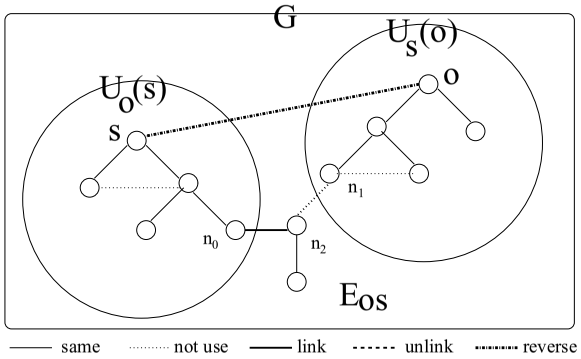

図 3 G_o から G_s への変更部分

けで目的を達成できることが解る．即ち，s, o 間のリンクを逆向きにし，$U_o(s)$ の要素 n_0 と E_{os} の要素 n_2 間のリンクを追加し，$U_s(o)$ の要素 n_1 と E_{os} 間のリンクを削除することによって，G_o から G_s を作ることができる．

4.2 最短経路の探索アルゴリズム

G_o から，次のような手続きを順番に呼び出して G_s を求める．

ただし，リンク $< i, j >$ の削除並びに，追加を行う手続きは，共に $O(n^2)$ の領域を費すことによって，$O(1)$ の時間計算量の手続きとして実装されている．

markIn o を終点とする最短経路網 G_o と，次の終点となるノード s を用いて，s の上流 $U_o(s)$ を計算する．定理 2 より，$U_o(s)$ の要素はこれだけである．

markEdge E_{os} の要素一部は，$U_o(s)$ の要素と G において隣接するので，$U_o(s)$ の要素の G において隣接するノードを全て調べ，E_{os} に入れると同時に，その上流もまた，E_{os} に入れる．定理 3 から，これによって，E_{os} の全ての要素が得られることが保証される．

relink E_{os} の要素の G におけるリンクを全て調べ，必要ならば書き換える．

clean 作業領域の後始末を行い，次のステップのために，距離の再計算をおこなう．

4.3 提案アルゴリズムの評価

単連結なグラフ $G =< N, V >$ において，$n = |N|, m = |V|$ とする．更に，$G_o =< N, V' >$ とした時，$m_o = |V'|, m' = \dfrac{\sum_{o \in G} m_o}{n}$ とする．

一般に次の関係式が成立する．

- $n - 1 \leq m \leq \dfrac{n(n-1)}{2}$
- $m_o \leq m$ $(\forall o \in N)$ よって $m' \leq m$.

4.3.1 Brandes のアルゴリズムの計算量

Brandes のアルゴリズム[4] は二つの部分から構成されており，次のような計算量になっている．

前段 $O(nm)$：n はノード毎に，最短経路網を作りなおすためであり，また，m は，その最短経路網の構成に，Dijkstra のアルゴリズムを利用するためである．

後段 $O(nm')$：後半は，媒介次数の計算のために，各々の最短経路網 G_o のリンクを全て辿るためである．

一般に，$m' \leq m$ であるため，Brandes のアルゴリズムの計算量は $O(nm)$ となる．

4.3.2 提案手法のアルゴリズムの計算量

提案のアルゴリズムでは，Brandes の手法に対して，前段の最短経路網の計算部分をのみを改良する．

そこで，以下では，最短経路網の計算部分のみを比較する．

定義より，$|G| = |U_o(s)| + |U_s(o)| + |E_{os}|$ が成り立つ．また，$|G| = n$ とした時，$n_i = |U_o(s)|, n_o = |U_s(o)|, n_e = E_{os}$ とすれば，一般に $0 \leq n_i, n_o, n_e \leq n$ となる．

更に，グラフ G における，ノード v のリンク数を $G(v)$ で表すとすれば，定義により，$\sum_{v \in G} G(v) = m$ となる．

この時，提案手法の個々の手続きのアルゴリズムの計算量は以下のようになる．

markIn $O(\sum_{v \in U_o(s)} G_o(v)) \leq O(m)$

markEdge $O(\sum_{v \in U_o(s)} G(v) + \sum_{v \in E_{os}} G_o(v)) \leq O(m)$

relink $O(\sum_{v \in E_{os}} G(v)) \leq O(m)$

clean $O(|U_o(s)| + |E_{os}|) \leq O(n)$

提案するアルゴリズムでは，$v \in U_s(o)$ 内のリンクを操作しないため，その分だけ高速化される．

$U_s(o)$ のサイズは，グラフの形状ならびに o, s の取り方によって変化するが，$o \in U_s(o)$ であるので 0 になる事はない．

したがって，$U_s(o)$ 内のノードに接続されたリンク数の総和の平均を m'' とすれば，$m - m'' \leq m$ となり，提案するアルゴリズムは，最短経路網の計算量を，$O(nm - nm'') \leq O(nm)$ と改良していることになる．

4.4 規則的なグラフに対する計算量

4.4.1 完全二分木

深さ $d(>1)$ の完全二分木 $B_d =< N_d, V_d >$ は，根あるいは枝と呼ばれ，二つの子を持つ $2^{d-1}-1$ 個のノードと，葉と呼ばれ，子のない 2^{d-1} 個のノードからなる，左右対称なグラフであり，そのノード数 $|N_d|$ は，$2^d - 1$，リンク数 $|V_d|$ は，$|N_d| - 1 = 2^d - 2$ となる．

これに対して，提案するアルゴリズムを常に，根から葉の方向に適用することを考える．

すると，提案するアルゴリズムでは，枝の先の要素だけをアクセスすることになるので，B_d の深さ k にある枝 n_k での最短経路網の再計算時間 $T(n_k)$ は，次の式で表せる．

$$T(n_k) = O(|B_{d-k}|) = O(2^{d-k} - 1) = \frac{O(2^d)}{2^k}$$

従って，B_d の全てのノードを終点とする最短経路網の計算時間 $T(B_d)$ は，次のようになる．

（ $T(B_0)$ は高さが 0 の木なのでその計算時間は 0 となる）．

$$\begin{aligned} T(B_d) &= T(n_0) + 2T(B_{d-1}) \\ &= O(2^d) + 2T(B_{d-1}) \\ &= O(2^d) + 2 \times O(n_1) + 2^2 T(B_{d-2}) \\ &= O(2^d) + O(2^d) + 2^2 T(B_{d-2}) \\ &= O(d2^d) \end{aligned}$$

ところが，このグラフに対して，Dijkstra のアルゴリズムをそのまま適用すると，$O(|N_d||V_d|) = O(2^d 2^d) = O(2^{2d})$ となるので，これに比較して高速化されている事が解る．

4.4.2 多次元トーラス上のメッシュグラフ

$d\,(>0)$ 次元のトーラスを $l\,(>3)$ 等分したメッシュ（ただし，計算を容易にするために，l を偶数の場合を考える）を考え，その交点をノードとするグラフ $M_{d,l} =< N_{d,l}, V_{d,l} >$ を考える．

このグラフのノード数 $|N_{d,l}|$ は，l^d となり，リンク数 $|V_{d,l}|$ は，$|N_{d,l}| \times d \times 2 = 2dl^d$ となる．

このグラフに提案するアルゴリズムを適用すると，$U_o(s)$ のノード数は，$M_{d,l}$ の半分の $\frac{l^d}{2}$ となり，E_{os} のノード数は，境界となるため，一次元減った $2 \times l^{d-1}$ となる．

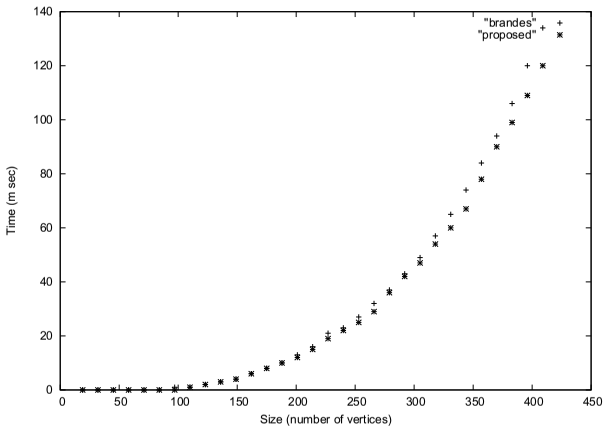

図 4 完全グラフ

したがって，提案するアルゴリズムを適用した場合の計算量は $\frac{l^d}{2} + 2l^{d-1}$ となる．この結果，もし，l が d に比べ十分に大きければ，ほぼ，半分の時間で処理できる事になる．

4.5 数値計算による計算量

アルゴリズムの性能を評価するために，幾つかのグラフ上での提案するアルゴリズムの実行速度を測定し比較した．実験の環境は，CPU：Intel Celeron 2.40GHz, Memory：1 G byte, OS：Fedora Core 3 で，その上に，C 言語を用いて実装した．以下，図の縦軸は最短経路網を計算するためだけに費した時間（m sec）．横軸はノード数をそれぞれ表す．

4.5.1 完全グラフ

完全グラフは全てのノード間にリンクがあるグラフで，この場合は，$U_s(o) = \{o\}$ となるため，ほぼ同等の速度になっている（**図 4**）．

4.5.2 二次元格子

二次元格子は，平面上に，ノードが格子状に配置されているグラフで，その上下左右のにリンクが張られ，縁を持つ．左から x 番目にあるノード o への最短経路網から，その左の $x+1$ 番目ノード s への最短経路を求めた場合，$U_s(o) = \{p|\ p$ は 左から x 番目以下のノード $\}$ となるため，平均的には 2 倍の速度になっている（**図 5**）．

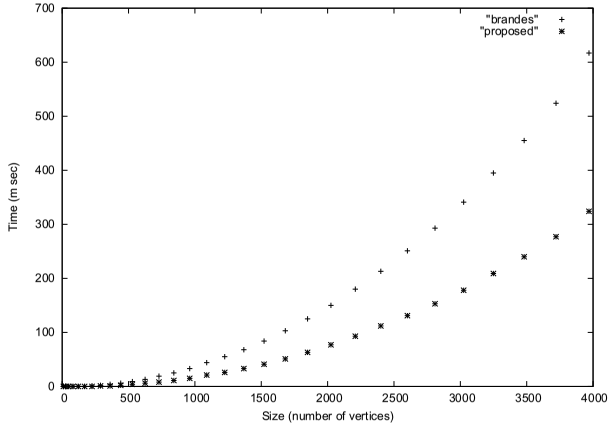

図 5 二次元格子

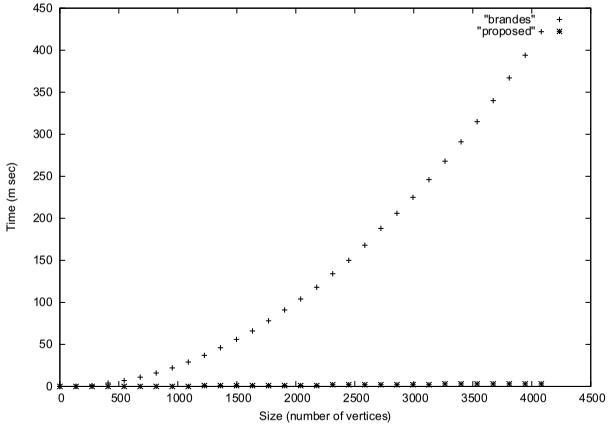

図 6 完全二分木

4.5.3 完全二分木

完全二分木の場合，Dijkstra のアルゴリズムをそのまま適用するとノード数 n に対して $O(n \times l) = O(n(n-1)) = O(n^2)$ となるが，提案するアルゴリズムでは $O(n \log n)$ となる（**図 6**）．

5. 結　　論

媒介中心性を高速に計算するために，Brandes のアルゴリズムの前段に当る，最短経路網の計算部分を変更する方法を述べ，そのアルゴリズムを実際のグラフに適用し，その振舞いを評価した．

この結果，この変更によって，ノード数に比較し，リンク数が少い場合は，速度が改善できることが解った．また，二分木などの特殊な場合には，より効果が高まることが，理論的にも数値的にも示すことができた．

しかし，本手法では，計算済みのデータを次々と書き換えているため，どの順番でノードを選択し最短経路を計算して行くかが，計算速度に本質的に影響する．例えば，完全二分木の計算では，ノード選択が最適になるようにしており，その最適な順番を得るため，グラフの中心を求め，そこへの最短経路網を作り，その上で，深さ優先でノードを選択しているが，これが一般的なグラフにおいても適切かは判断できていない．

今後は，一般のグラフに対する論理的な振舞いを調べ，どのグラフの場合に，どのような順序でノードの選択を行えば，提案するアルゴリズムが効率的に働くかを考えてゆきたい．

また，今回は，計算の目的を媒介中心性としたため，リンクに重みがない (全てのリンクの重みが 1 である) 場合に対象を限定したが，E_{os} の探索を行う MarkEdge 関数に対し、リンクの重みを配慮するような改良を加えることにより，重み付けされたグラフ上にもアルゴリズムが拡張できると考えている．これは，媒介中心性の一般化にあたるフロー中心性[1)]の計算に必要であり，次の課題と考えている．

参　考　文　献

1) 金光,「社会ネットワーク分析の基礎」, 勁草書房 (2003).
2) Freeman, L. C., “A set of measures of centrality based on betweenness”, Sociometry, 40:35-41.(1977).
3) 栗野, 吉開, “アフィリエーションネットワーク概念を用いた組織内情報分析法のモデル化と情報ネットワーク設計への応用”, FIT2008, R0-002, pp 81-84 (2008).
4) Ulrik Brandes,“A Faster Algorithm for Betweenness Centrality”, Journal of Mathmatical Sociology vol.25(2), pp.163-177 (2001).
5) Dijkstra, E.W. “A note on two problems in connexion with graphs”, In Numerische Mathematik, 1 (1959), S. 269-271.