道路網に対する全対最短経路索引に関する検討

田中　翔† 壷内　貴弘† 蒲原　智也†† 上島紳一 †††

†,††† 関西大学総合情報学部　〒 569–1095 大阪府高槻市霊仙寺町 2-1-1
†† 関西大学先端科学技術推進機構　〒 564–8680 大阪府吹田市山手町 3-3-35
E-mail: †††ueshima@res.kutc.kansai-u.ac.jp

あらまし　本稿では，道路網上の任意の 2 点間の全対最短経路の経路索引を生成する手法について検討する. 本手法では最短経路の持つ経路コヒーレンスの特徴を生かすため, ノードの位置座標による領域分割を用い, 最短経路が領域分割線と交差してできる境界に着目して, 最短経路の部分経路と経路索引を生成する. これにより最短経路は出発地点と目的地点と境界の列を用いて表すことができ，　最短経路抽出のみならず, 領域の包含関係を用いた刈込処理や道路網の詳細度に応じた経路の抽出が可能となる. 経路索引は, 全対最短経路集合の全データ量に対して小さく, モバイル応用などに利用しやすい特徴を持つ. 提案手法を評価するため, 国土地理院発行数値地図を用いて経路索引の生成時間, サイズ, 問合せ時間などを定量的に検討する. また経路索引のデータベース格納についても検討する.

キーワード　経路索引, 全対最短経路, 経路コヒーレンス, 道路網, グラフ

1. はじめに

近年, 大規模グラフに対して全対最短経路の高度計算環境を利用した計算法の研究が進められている [1] [3] [4] [5]. 全対最短経路データは汎用性が高く，グラフ航行，旅程計画，工程管理など様々な分野での利用が期待できる [2]. さらに，データを構造化して格納することで，最短経路計算を行う度にグラフを探査する処理が不要となり，複数の利用者間で経路データを共有したり，注釈を付けるなど，経路を属性付けて利用できる.

本稿では, 道路網応用を想定してグラフが参照座標系に埋め込まれているものと仮定し，全対最短経路データに対する経路索引を構成する手法について議論する. ここでは最短経路の経路コヒーレンス [7] [8] [9] を生かして，領域分割でできる境界に着目して，ノード位置を基準に道路網データから領域間全対最短経路を生成し，木構造へ格納する. 一方, 最短経路の問合せ処理では木構造を辿って境界を抽出し，最短経路を抽出する．問合せ処理は z コードの接頭辞処理に帰着できる [11].

提案手法は次の特徴を持つ．すなわち

- 最短経路の各部分を領域に対応付けることにより，領域の包含関係による刈込みを用いた問合せ処理が可能となる.
- 整形・不整形なグラフにも適用することができる．また，経路の詳細度に応じた最短経路の抽出が可能である.
- 経路索引のデータ量は，全対最短経路の全データ量に対して小さく，モバイル環境での最短経路探査，距離問合せ，範囲問合せ処理などを用いた位置情報サービスなどに利用できる.

提案手法を評価するため, 国土地理院発行の数値地図 [14] を用いて，経路索引の生成時間，問合せ処理時間，サイズなどを定量的に評価する．さらに，経路索引をデータベースに格納し，SQL を用いて検索を行う方法についても検討する.

以下，2 節で関連研究について述べ，3 節で全対最短経路の特徴，4 節で経路索引の生成と抽出について述べ，評価し，5 節で索引のデータベースへの格納と問い合わせ法を述べ，評価する.

2. 関連研究

[経路コヒーレンス] Samet ら [6] は, 最短経路の出発地点の持つ out エッジに着目してデータ集合を連続領域に分割できることに着目したデータ構造 (SILC 構造) を提案している. この性質は到達地点の経路コヒーレンスと呼ばれる. さらに Sankaranarayanan らは最短経路集合に対して出発地点・到達地点間の経路コヒーレンスを定義し, 最短経路集合を経路コヒーレント対の集合に分割 (PCPD 分割) して四分木に格納する方法について議論している [7], [8], [9]. これにより最短経路を（出発地点, 目標地点, 中間地点）の 3 つ組で捉え, 最短経路の距離情報と合わせて距離問合せにも対応できる格納法を提案している. ここで中間地点はノードまたはエッジである. しかし, ノードが平面上に偏在する場合への対応や領域分割でのエッジの取扱いや不整形なグラフへの対応が必ずしも明確でない. Wu ら [10] は, 上記手法とノードの重要度付けを用いる手法とを比較しながら評価している.

[部分領域の全対最短経路計算] Jing ら [12] は 2 次元方向に領域を分割してグラフを分割し, 部分領域での全ての地点間の最短経路を前処理として計算し, 階層化したデータ構造 (経路ビュー) を提案している。蒲原ら [13] は, グラフを NV 分割し [11], ボロノイ領域毎に境界と母点からの全対最短経路を求めて経路索引を構成している. さらに隣接するボロノイ領域を統合して階層化し, 遠方の 2 点間の最短経路を求める手法を提案している.

[提案手法] 最短経路を領域分割線でできる境界に着目し, ノード位置による領域分割と, 次にエッジの接続関係をもとにした各最短経路の部分経路生成を行って, グラフ上の任意の 2 点間を結ぶ最短経路の構成を問合せ処理で行っている. これにより出発地点と目的地点と境界の列を用いて最短経路を表すことができ, 二分木の高さに応じた境界列の飛び抽出などにより詳細度に応じた経路の抽出が可能となる. またノード・エッジ位置に粗密のある不整形グラフにも適用できる. さらに経路索引をデータベースに格納し SQL を用いた検索法を検討している.

3. 全対最短経路集合

3.1 定義と特徴

V をノード集合，E をエッジ集合 $(E \subset V \times V)$ とし，$e \in E$ は非負値を属性値として持つグラフ $G(V,E),(N=|V|,M=|E|)$ を考える (単に G とも書く)．本稿で G は無向グラフであり，参照座標系に埋め込まれていることとする．また，任意のノード v は，位置座標 (v_x,v_y) を持つものと仮定する．

G を道路網とみる場合は，交差点をノード，交差点を結ぶ道路片をエッジ，道路片の属性値をエッジの属性値とする．

[定義] (全対最短経路集合) $G(V,E)$ 上の全ての 2 ノード間を結ぶ最短経路の集合を全対最短経路集合という．各最短経路は $G(V,E)$ 上の 2 ノード間に対して定義され，ノード列で表せる．

[特徴] 全対最短経路集合と各最短経路は次の特徴を持つ．

（1） 全対最短経路集合は N^2 の経路を含み，$N >> 0$ の場合，大規模データとなる．

（2） 最短経路の部分経路はその区間で最短経路である．

（3） 最短経路は**経路コヒーレンス**を持つ．

[Note] 上記 (3) で，G 上のある地点 v_s から最短経路で到達可能なノード集合は，v_s の第 1 エッジにより分類できる性質を，**到達地点の経路コヒーレンス**という [6]．また，出発地点 v_s と目的地点 v_t のそれぞれの近傍からの最短経路集合が，共通エッジ・ノードを持つ性質を**経路コヒーレント対**と呼ぶ．

[例 1] 図 2 で，赤色のエッジを共通エッジとして持つ最短経路集合が黄色部分であり，これは経路コヒーレンス対を示す．

つまり近傍領域にある v_s,v_s' から遠方領域にある v_t,v_t' への 2 つの最短経路は同一の部分経路 $\psi(\psi$ はノードまたはエッジ）を持つ [7] [8] [9].

上記の特徴を考慮し，全対最短経路集合の格納構造に対する次の要件を挙げる．

[要件 1] (3) より全対最短経路集合の各最短経路のノード列を全て格納するのではなく，領域の包含関係を利用しながら，最短経路と部分経路の関係を組み込んで，格納する方法がよい．

[要件 2] (v_s,v_t) の組に対して格納・抽出できる方法がよく，従来のデータベース技術を利用できることが望ましい．

上記要件を考慮して，提案手法では [7], [9] の手法を発展させて，ノードの位置座標をもとにして領域分割を行い，最短経路が交差する領域の境界に着目して，索引を生成する．

3.2 領域分割線と全対最短経路集合の分割

[境界と最短経路] G に対して，水平・鉛直な領域分割線 l を引く．l と交差する G のエッジを**境界エッジ** (単に**境界**）といい，境界エッジの両端のノードを**境界ノード**という．

[例 2] 図 1 左のグラフで全対最短経路集合を考えると，鉛直な分割線 l_1 に対して，b_1,b_2,b_3 が境界となり，境界ノードはそれぞれ異なる 2 つの部分領域 A_0,A_1 に属する．ノード $v_s \in A_0$ からノード $v_t \in A_1$ を結ぶ最短経路 p(緑色) を考えると，p は必ず 3 つの境界のうちの 1 つを通過する．但し，G が整形な場合，最短経路が通過する境界は 1 つであるが，G が不整形 (図 1 右) の場合，$v_s' \in A_0, v_t' \in A_0)$ のように G 上では到達可能であって，l_1 による分割後に A_0 の部分グラフのみでは到達できず，一旦

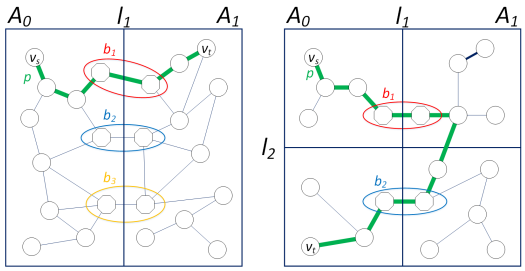

図 1　分割線 l_1,l_2 による領域分割

Fig. 1　Region Splitting by l_1,l_2

図 2　経路コヒーレンス (左：疎 ($N=960$)，右：密 ($N=5284$))

Fig. 2　Path Coherence (left : sparce ($N=960$), right : dense($N=5284$))

A_1 を通過しないと到達できない最短経路が存在する．従って境界と交差する最短経路に着目すれば，最短経路は両端ノード v_s,v_t と通過する境界のリストを用いて次のように表せる．

$$(v_s,[b_{j_1},\ldots b_{j_m}],v_t,c), v_s \in A_i, v_t \in A_j,\ i \neq j \qquad (1)$$

$$(v_s,[b_{j_1},\ldots b_{j_m}],v_t,c), v_s \in A_i, v_t \in A_i, \qquad (2)$$

但し $i,j \in \{0,1\}$ で，境界リスト $[b_{j_1},b_{j_2},\ldots b_{j_m}]$ は，p が l_1 を m 回交差する場合を示し，式 2 で m は偶数，式 2 で m は奇数である．c は v_s,v_t 間の最短経路のコストである．

[境界による全対最短経路集合の分割] 全対最短経路集合は分割線 l によって 4 種類の最短経路集合に分割される．すなわち

(i)　A_0 内の 2 ノードを結ぶ最短経路で l_1 と交差しない経路．即ち A_0 内にあるノード列で表せる最短経路集合．

(ii)　A_1 内の 2 ノードを結ぶ最短経路で l_1 と交差しない経路．即ち A_1 内にあるノード列で表せる最短経路集合．

(iii)　A_0 内のノードから l_1 を（1 回または奇数回）跨いで A_1 内のノードに至る経路．

(iv)　A_0 内のノードから l_1 を（2 回または偶数回）跨いで A_0 内のノードに至る経路．

[例 3] 上記 (i)-(iv) の分割で境界から見れば，各領域 A_i において A_i 内の全ての境界ノードから，並列ダイクストラ法 [13] を実行すれば，各境界から A_0 内と A_1 内でそれぞれ最も近い複数のノード集合が得られる．同様に右側領域に属す境界ノードからも同様にノード集合を分割できる．これは領域内の各ノードから最も近い境界を求めていることに相当する．

同様にして，次に水平方向に集合を分割し，分割を各領域にノードが 1 つになるまで繰り返すことができる．

4. 全対最短経路索引

3.2 節の全対最短経路集合の分割手順を用いて, 本節では G の全対最短経路の計算と, 全対最短経路のための経路索引の生成手順を述べる. 経路索引は二分木構造を持ち, G 上の任意の 2 点 v_s, v_t に対して木を辿ることで最短経路を求めることができる.

4.1 生 成 手 順

与えられた G に対する生成手順は以下の通りである.

(1) 二分木構造生成 参照座標系によるノードの位置座標を用いてトップダウンに領域を分割し，二分木を構成する (図 3). すなわち二分木の根ノードを G の存在する全領域 S とした時, S を鉛直線・水平線で交互に 2 等分し, ノード集合 V を領域内ノード数が 1 になるまで分割を繰り返す. 1 となったとき， 各ノード ID を自身が所属するユニークな領域コードに置換する.

分割された領域 A_i のインスタンスはリストに格納され，i は図 3 のような z 曲線 [11] を用いた z コードで表現される. 生成される二分木の高さは，$o(log_2 n)$ となる.

(2) 最短経路計算と境界エッジ生成 最下層レベルから順にボトムアップに各レベルで領域内・領域間の全対最短経路と境界を求め, 境界エッジのデータを格納する.

(2-1) 最短経路計算と境界エッジの番号付け 隣接領域間の全てのノードの組合せに対して最短経路計算を行い，それぞれに関して境界エッジ b，コスト c を求める.

(2-2) 最短経路索引の格納 b, c と v_s, v_t を $(v_s, [b_{j_1}, \ldots b_{j_m}], v_t, c)$ の形式とし，これを最短経路索引として，部分木に格納する.

(3) 全領域の最短経路索引生成及び格納 全ての領域において (2) を実行し，最短経路索引を生成，格納する.

[Note] 木構造は (1) で生成し，(2)(3) で全対最短経路及び境界エッジデータを小さな領域からボトムアップに生成・格納する.

[Note] 全対最短経路データが既に得られている場合に本手法を適用する場合は，最短経路が領域分割線 l を交差する境界を求める処理が必要になる.

4.2 2 点間の最短経路の抽出手順

G 上の任意の 2 点 v_s, v_t が与えられた場合の経路索引を用いた最短経路抽出手順は，次の通りである.

(1) 部分木の指定と最短経路索引の抽出 v_s, v_t 間に共通な領域を求め，最短経路索引を抽出する.

(1-1) v_s, v_t 間の共通部分木の指定 ノード ID が z 曲線コードで表現されていることを用い，v_s, v_t 間に共通な部分木のコードを v_s, v_t の ID から求める.

[例 4] v_s のノード ID が ”0000001”，v_t のノード ID が ”0001000” であるとき，共通部分木のコードは ”000” である.

(1-2) 最短経路索引の抽出 (1-1) で求めた共通部分木内の最短経路索引集合から，v_s, v_t に対応する最短経路索引を抽出する. 抽出された最短経路索引のノードを 2 つずつの組 (v'_s, v_t) とし，各組に対して，(1-1) の処理を行う.

(2) 最短経路ノード列の抽出 再帰的に二分木を辿り，v_s, v_t 間の最短経路ノード列を抽出する. (1) の処理を再帰的に行い，組の 2 ノードが同一ノードとなったとき，そのノードを確定ノードとし，最短経路ノード列に追加する. 全てのノードが確定ノードとなったときに生成された最短経路ノード列が最短経路を示す.

[Note] 前節のうち, 再帰的に木を辿る処理の呼び出し回数を制限することによって，詳細度に応じた最短経路のノード列を抽出できる.

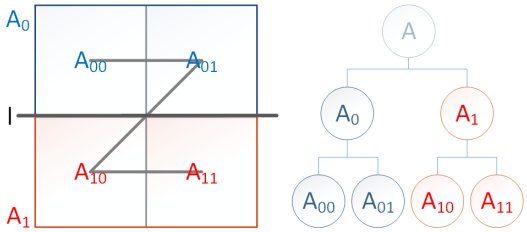

図 3 領域分割と二分木

Fig. 3 Region Splitting and Binary Tree Representation (z-code)

表 1 評価用国土地理院道路地図データ

Table 1 Road Map Data issued by GSI

	$\|N\|$	$\|M\|$	全対最短経路数
GSI1	960	1,325	921,600
GSI2	5,601	7,509	31,371,201

表 2 原データと経路索引のデータサイズ比較

Table 2 Size Comparison of Binary Tree and Original Graph Data

地図	原データ	全対経路	経路索引	二分木高さ	索引生成時間
GSI1	180KB	222MB	39.2MB	25	63,325ms
GSI2	1,592KB	13.8GB	1.43GB	32	22,646,154ms

4.3 評 価

[対象地図データ] 表 1 に示す国土地理院数値地図 2,500(領域データ基盤，1/2,500 縮尺) [14] を用いた.

[実行環境] SunOS 5.10 Generic 144500-19 32 sparcv9 プロセッサ (2530 MHz，浮動小数点プロセッサ付), 主記憶 32GB，Java version 1.5.0-20 である.

[評価方法]

ダイクストラ法との比較によって評価を行う.

- 経路索引生成 : 提案手法では，原地図データを読み込み，全対最短経路索引を生成し，中間ファイルに出力するまでを測定した.
- 最短経路抽出 : 任意の (v_s, v_t) を指定し，最短経路を返すまでを測定した. ダイクストラ法は索引生成処理が必要ないため，直接探査を行う.

4.3.1 経路索引生成に関する評価

表 2 で経路索引生成時間とデータサイズを示している. 生成時間，データ量ともにノード数 N の増加にしたがって増加する. 経路索引のデータサイズに関しては，原地図データサイズより大きくなるが，完全な全対最短経路データサイズより小さくなる.

4.3.2 最短経路抽出に関する評価

それぞれの地図データに対し，各ホップ数で到達可能な (v_s, v_t) を 100 組用意し，提案手法とダイクストラ法で処理を行い，平均実行時間及び展開数を計測した.

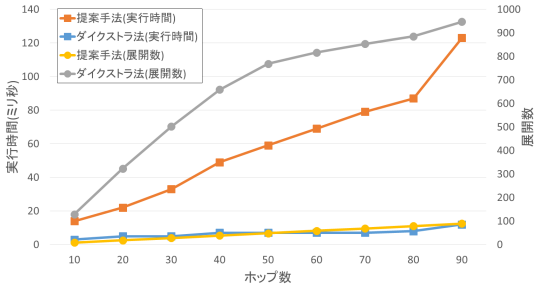

図 4　ホップ数に応じた最短経路抽出時間と展開数 (GSI1, N = 960)

Fig. 4　Extraction Time and Number of Expansions vs. Hops for (v_s, v_t) (GSI1,N = 960)

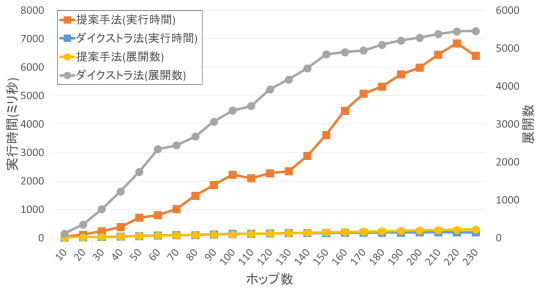

図 5　ホップ数に応じた最短経路抽出時間と展開数 (GSI2, N = 5601)

Fig. 5　Extraction Time and Number of Expansions vs. Hops for (v_s, v_t) (GSI2,N = 5601)

(1) 抽出及び探査時間に関する評価

図 4，図 5 で，ホップ数ごとの各手法による最短経路探査及び抽出時間について示す．本手法ではホップ数に大きな影響を受け，実行時間が長くなる傾向にある．それに対して，ダイクストラ法による処理時間はホップ数の影響を受けるものの，軽微なものとなっている．

(2) 展開数に関する評価

図 4，図 5 で，ホップ数ごとの各手法による展開数について示す．本手法では，v_s から v_t に対して最短経路上のノードのみを展開するため，ホップ数と展開数はほぼ同じ値となる．それに対して，ダイクストラ法では，全方位に展開するため展開数はホップ数に大きく依存し，ホップ数が増加すると，展開数も大きく増加することがわかる．

5.　データベースの利用

4 章で生成した全対経路索引集合をデータベース (MySQL) 利用効果を確認する. ここでは JDBC を用いて接続する.

5.1　ス キ ー マ

ここでは全対最短経路索引を格納する場合と, 全対経路集合を格納する場合のスキーマを示す．ここでは全対最短経路索引と全対経路集合を，データベースへ格納する前にあらかじめ最大境界数・経路長を調べ，それぞれ固定長のテーブルを作成している．

[全対最短経路索引に対するスキーマ] 表 3 に提案手法で生成した全対経路索引をデータベースに格納する際に用いたスキーマを示す. フィールドは, 始点ノード (StartN), 終点ノード (GoalN), 距離 (Cost), 境界ノード (Eg_i) とし, 始点と終点ノードの組合せによって境界ノード数が異なるため、境界ノード数が短い場合には NULL を用いる. ここでは, 最短経路索引の数だけテーブルを作成, テーブル名を最短経路索引とし, フィールド StartN,GoalN に索引を付与する. また, 最短経路問い合わせをする際, ノード ID を z コードで表したものをを利用ため,StartN,GoalN,Eg_i では, z コードを用いる.

表 3　全対最短経路索引に対するスキーマ

Table 3　Relational Schema for Indices of All-pairs Shortest Paths

StartN	GoalN	Cost	Eg_1	Eg_2	...	Eg_i	...	Eg_{14}
000100	01100	5841.48	000100	0110011	...	NULL	...	NULL
000100	01101	5735.17	000100	01100011	...	NULL	...	NULL

[全対最短経路集合を直接格納するスキーマ] 表 4 は, ダイクストラ手法でもとめた全対経路集合をデータベースに格納する際に用いたスキーマを示す. フィールドは, 始点ノード (StartN), 終点ノード (GoalN), 距離 (Cost), ホップ数 (Rt_m) とし, 始点と終点ノードの組合せによって, ホップ数が異なるため, ホップ数が短い場合には NULL を用いる. ここでは, 一つのテーブル (allpairs) に全対経路集合を格納し, フィールド StartN,GoalN に索引を付与する. また, 全対経路集合は生成する際, 経路結果をノード ID で出力しているため,StartN,GoalN,Rt_m では, ノード ID を用いる.

表 4　全対最短経路集合を直接格納するためのスキーマ

Table 4　Relational Schema for All-Pairs Shortest Paths

StartN	GoalN	Cost	Rt_1	Rt_2	...	Rt_m	...	Rt_{300}
347881	347899	5841.48	347881	347923	..	347926	...	347899
347881	347918	5735.17	347881	347924	..	NULL	...	NULL

5.2　最短経路問合せ

[全対最短経路索引を用いた問合せ手順] G 上に任意の 2 点 v_s, v_t が与えられた場合の最短経路の問合せ手順を次に示す.

(1) 最短経路索引の抽出

データベース内に v_s, v_t が格納されているテーブルを指定するため,v_s と v_t の z コードを用いて，最短経路索引を求める.

(1-1) v_s, v_t を含む最短経路索引の指定とデータベース問合せ

ノード ID が z コードで表現されていることを利用して,v_s, v_t の最短経路索引を求める．求めた最短経路索引を用いて, データベース内のテーブルを指定し, フィールド StartN と GoalN から v_s, v_t に対応するフィールド全てを問合せる.

(1-2) v_s, v_t の最短経路を構成する境界列の再帰的抽出

再帰的にデータベースに問合せることで, 任意の 2 点 v_s, v_t 最短経路を構成する境界列を抽出する. 抽出した最短経路索引のノードを組 ([$StartN$,Eg_1],...,[Eg_{14},$GoalN$]) とし，各組に対してそれぞれ (1-1) の処理を行いデータを問合せる.

(2) 最短経路ノード列の抽出

組の 2 ノードが同一ノードとなったとき，そのノードを確定ノードとし，最短経路ノード列に追加する．全てのノードが確定ノードとなったときに生成された最短経路ノード列が最短経路を示す.

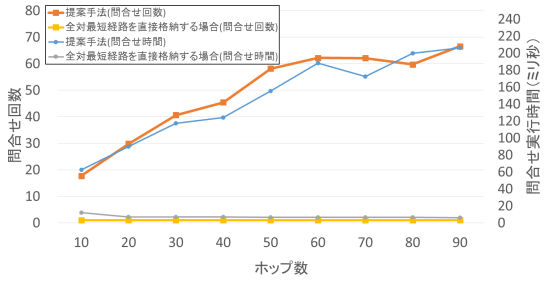

図 6　ホップ数による問合せ回数, 問合せ実行時間 (GSI1,N = 960)

Fig. 6　Access Count and Query Time vs. Hops for (v_s, v_t),(GSI1,N = 960)

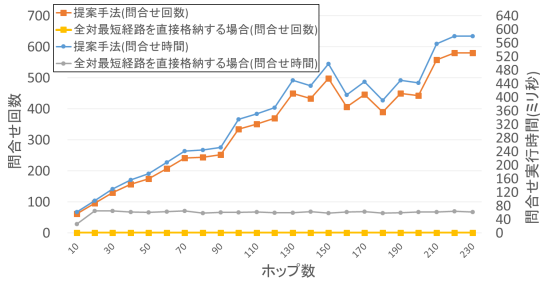

図 7　ホップ数による問合せ回数, 問合せ実行時間 ,(GSI2,N = 5601)

Fig. 7　Access Count and Query Time vs. Hops for (v_s, v_t),(GSI2,N = 5601)

[全対最短経路を直接格納する場合の問合せ手順] 最短経路の問合せ手順を次に示す.

- allpairs テーブルを指定し, フィールド StartN,GoalN から v_s, v_t に対応する最短経路を問合せる.

5.3　評　　価

[対象データ] 4 章で生成した全対最短経路集合と全対経路集合をデータベース内に格納したものを利用.

[評価方法] それぞれの地図データに対し，各ホップ数で到達可能な組合せ (v_s, v_t) を 100 組用意し, 問合わせ実行時間と問合せ回数を測定する.

[問合せ実行時間に関する評価]　図 6 に，960 ノードにおけるホップ数と問合せ回数, 問合せ実行時間の関係，図 7 に，5601 ノードにおけるホップ数と問合せ回数，問合せ実行時間を示す. 全対最短経路索引問合せでは，経路索引を用いて探査するため, ホップ数の増加と共に問合せ回数が増加し，問合せ実行時間も増加する．一方，全対最短経路を直接格納する場合の問合せでは，最短経路が格納されている部分を問合せればよく，アクセス数が一定となり, ホップ数による問合わせ実行時間のばらつきが少ない.

6.　お わ り に

本稿では, 二分木を用いた全対最短経路の格納及び抽出方法について議論した. これは，探査処理の代替として問合せ処理により最短経路を抽出する方法である. 提案手法では経路索引生成時間が長く，大規模グラフに適用する場合には，生成時間を短縮する方法を検討する必要がある．また，最短経路抽出処理の実行時間のほとんどを索引ファイルの読み込み時間が占めているため，ファイル編成を再検討することが必要である．次に，データベースに全対最短経路索引を格納する場合，最短経路問合せ処理を再帰的に行っているため，実行時間の増加の原因となっており，問合せ回数を抑える方法を検討する必要がある.

文　　献

[1] A. Moffat, T. Takaoka, *An All Pairs Shortest Path Algorithm with Expected Time $O(n^2 \log n)$*, SIAM Journal on Computing, 16(6), 1023–1031 (1987)

[2] A. Botea, D. Harabor, *Path Planning with Compressed All-Pairs Shortest Paths Data*, Proceedings of the 23rd International Conference on Automated Planning and Scheduling(ICAPS 2013), pp.1–5, (2013)

[3] G. J. Katz, J. T. Kider Jr, *All-Pairs Shortest-Paths for Large Graphs on the GPU*, Proceedings of the 23rd ACM SIGGRAPH/EUROGRAPHICS symposium on Graphics Hardware (GH'08) pp.47–55 (2008)

[4] T.Okuyama, F. Ino, K.Hagihara, *A Task Parallel Algorithm for Finding All-Pairs Shortest Paths Using the GPU*, International Journal of High Performance Computing and Networking, Vol. 7, No. 2, pp. 87–98, (2012).

[5] GraphCREST(代表：藤澤克樹), 科学技術振興機構, `http://www.graphcrest.jp/jp/fujisawa-g.html`

[6] J. Sankaranarayanan, H. Alborzi, H. Samet, *Efficient Query Processing on Spatial Networks*, Proceedings of the 13th ACM International Symposium on Advances in Geographic Information Systems(ACM-GIS2005), pp.200–209 (2005)

[7] H. Samet, J. Sankaranarayanan, H. Alborzi, *Scalable Network Distance Browsing in Spatial Databases*, Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp.43–54 (2008)

[8] J. Sankaranarayanan, H. Samet, H. Alborzi, *Path Oracles for Spatial Networks*, Proceedings of the 35th International Conference on Very Large Data Bases(VLDB), pp.1210–1221 (2009)

[9] J. Sankaranarayanan, H. Samet, *Distance Oracles for Spatial Networks*, Proceedings of the 25th IEEE International Conference on Data Engineering (ICDE), pp.652–663 (2009)

[10] L. Wu, X. Xiao, D. Deng, G. Cong, A. D. Zhu,. S Zhou, *Shortest Path and Distance Queries on Road Networks: An Experimental Evaluation*, Proceedings of the 38th International Conference on Very Large Data Bases(VLDB), pp.406–417 (2012)

[11] H.Samet, *Foundations of Multidimensional and Metric Data Structures*, Morgan Kaufmann (2006).

[12] N.Jing, Y.-W.Huang, E.A.Rundensteiner, *Hierarchical Encoded Path Views for Path Query Processing: an Optimal Model and its Performance Evaluation*, IEEE Transactions on Knowledge and Data Engineering,Vol.10,No.3, pp.409–32 (1998)

[13] 蒲原智也，上島紳一, 道路網応用のための空間索引木の提案と最短経路探査への応用，情報処理学会論文誌データベース Vol.2, No.2, pp.10–28 (2009)

[14] 国土地理院:数値地図 (空間データ基盤), `http://sdf.gsi.go.jp/`