D-021

軽量マテリアライズドパスビューによる道路網距離での最短路探索
A Fast Verification Method for Trip Planning Query Based on Incremental Euclidean Restriction

山田　皓大 *　　**大沢　裕** *
Akihiro Yamada　　Yutaka Ohsawa

1. まえがき

現在位置 (s) と目的地 (d) が与えられたとき，sd 間の道路網上での最短路を求めるアルゴリズムとしては，従来 Dijkstra アルゴリズムや A*アルゴリズムが用いられてきた．これらは，道路網のグラフを隣接リストで表現しておき，その隣接リストを参照しつつ最短路探索を行うアプローチである．しかし，sd 間の距離が離れているとき，隣接リストの参照回数が増大することから sd 間の距離の増大につれ演算時間も増大する．

一方，道路網上のあらゆる 2 点間の距離を予め算出して表に記録しておき，その表を参照して道路網距離を求める方式も古くから提案されている．この方式は一般にマテリアライズド・パス・ビュー (materialized path view : MPV) と呼ばれる．sd 間の距離を求めるとき，もし s と d が共に道路網上のノードにあれば表を 1 回参照することにより sd 間の距離が求まる．

しかし，この MPV には次の問題点がある．通常，道路網グラフには膨大な数のノードが存在し，MPV の表のサイズはノード数 (n) の 2 乗に比例するためデータ量が膨大になること，隣接するノード間を移動する距離や時間に変更が生じたとき，その変更は MPV 表の大きな範囲に影響すること，MPV 表の作成に際しては道路網グラフの全ての 2 点間の距離を求める必要があることから，構築コストや変更コストが膨大になる，などの点である．これらの欠点に対応するため，階層的な MPV 構築法が各種提案されている．これらの方式は上記の問題点を緩和しているものの，基本的には上述の問題を抱えている．

一般の LBS，即ち kNN 検索や範囲検索，ANN 検索，旅行計画などの応用では，探索範囲が比較的狭い領域（例えば検索点を中心とした半径 50km 程度の範囲）で大量の最短路探索が必要となる．また，検索対象の POI(point of interest) は一般道路上に存在することが多いことから上記のような道路属性による階層化を利用した探索法をとることが難しいという面がある．

本稿では，道路網を多数のサブグラフに分割し，そのサブグラフ内での距離のみをマテリアライズ化した軽量データを用い，最短路探索には最適優先探索を用いる方式を提案する．これらの方式は，2 点間の道路網距離長い場合，通常の A* アルゴリズムに比して大幅な速度向上が得られる．また，従来提案されている階層的な距離のマテリアライズ化方式に比して，大幅なデータ量の減少を達成しており，かつ大きな探索時間の増加がないことを実験で検証する．

2. 軽量マテリアライズドパスビュー

ここでは，それぞれ SPFLM，SPFMM，SPFFM と呼ぶ 3 種類の軽量 MPV 方式を提案する．これらの違いは，どの程度にマテリアライズ化されたデータ（その程度によりデータ量が異なる）を用いるかの違いである．

まず，道路地図をタイル分割により多数のサブグラフに分割する．この分割は道路網のノードに注目して行う．即ち，ノードには，1 つのサブグラフのみに属すノードと複数の隣接するサブグラフにそれぞれ含まれるノードがある，前者を内部ノード，後者を境界ノードと呼ぶ．各サブグラフについて，境界ノード間の距離を示す表 (BBDT)，内部ノードから境界ノードへの距離を示す表 (IBDT)，サブグラフ内の全てのノード間の距離を示す表 (NNDT) を作成する．提案方式の内，隣接リスト (Adj. List) を必要とするものもある．各方式により，これらの内用いられるものが異なる．それぞれの方式で用いるマテリアライズ表を表 1 にまとめる．

図 1 に方式の概要を示す．ここで，s と d 間の道路網上での最短路が探索されるものとする．上で述べた構造を用いた最短路探索の制御には優先順位付きキュー (PQ) を用いた最適優先探索で行う．この PQ は，次のレコードを管理する．

表 1: SPF で用いるテーブル

Data table	SPFLM	SPFMM	SPFFM
BBDT	✓	✓	✓
IBDT		✓	
NNDT			✓
Adj. List	✓	✓	

$$< p, Cost, dfn, phase >$$

ここで，p はこのレコードが対応する境界ノードである．$Cost$ は s から d までの道路網距離の下限値であり，PQ はこの値が小さいレコードから順に返す．dfn は，s から現在の注目ノード p に至る道路網上の距離である．最後の $phase$ は，処理進行の状態を表わす変数であり，初期状態 (PHASE0) から終了状態 (PHASE3) 間の値をとる．

まず，図 1(a) に示すように，s が属す部分グラフを決定し，それを SG_s と呼ぶ．次に，SG_s の全ての境界ノード $b_i(\in BV_s)$ に対して，$Cost = d_E(s, b_i) + d_E(b_i, d)$ を計算し，優先順位付きキュー PQ に入れる．ここで $d_E(x, y)$ は 2 点 x,y 間のユークリッド距離を表わしている．s が属す部分グラフの決定法に関しては後述する．即ち，この段階では以下のレコードが PQ に入れられる．

$$< b_i, d_E(s, b_i)+d_E(b_i, d), 0, PHASE0 > \quad for\ all\ b_i \in BV_s$$

次に，PQ から $Cost$ の値が最小の要素を取り出す．図 1(b) は，PQ から取り出されたレコード (e) の $phase$ の値が PHASE0 の場合を示している．$e.p$ に対して，s,p 間の道路網距離 (これを $d_N(s, p)$ と表わす) を求める．更に，$e.p$ に対して，$Cost = d_N(s, e.p) + d_E(e.p, d)$ の値を付け，$phase$ の値を PHASE1 として PQ に投入する．この状態で作成されるレコードを以下に示す．

$$< e.p, Cost, d_N(s, e.p), PHASE1 >$$

PQ から取り出されたレコード (e) の $phase$ 値が PHASE1 の場合 (図 1(c))，既にそのレコードの p ($e.p$) に対しては，s からの道路網距離が計算され，$e.dfs$ に入れられている．このとき，$e.p$ に隣接する全ての部分グラフ SG_n の全ての境界ノード (BV_n) に対して，次のコストを計算して PQ に投入する．

$$Cost = e.dfs + d_N(p, b_i) + d_E(b_i, d) \quad (b_i \in BV_n)$$

但し，BV_n は SG_n の境界ノードの集合の 1 つを表わしている．この段階で PQ に投入されるレコードは以下の様になる．

$$< b_i, Cost, e.dfs + d_N(p, b_i), PHASE1 >$$

この過程を続けていき，PQ から取り出された境界ノード ($e.p$) が，d を含む部分グラフ (SG_d) の境界ノードの 1 つと一致したとき，次のレコードを作成して PQ に投入する．

$$< e.p, e.dfs + d_E(e.p, d), e.dfs, PHASE2 >$$

*埼玉大学

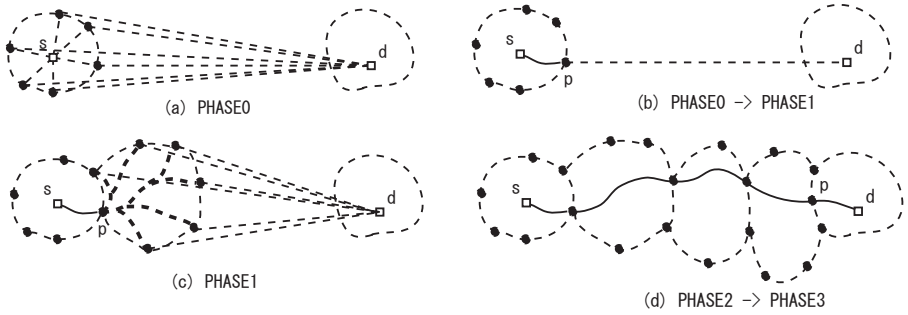

図 1: SPF の処理フロー

即ち，この場合には，コスト値に sp 間の道路網距離 ($e.dfs$) と p から d までのユークリッド距離を代入し，また $phase$ を PHASE2 に 1 つ進める．

もし，PQ から取り出されたレコード e の $phase$ 値が PHASE2 の場合，p と d との道路網距離 ($d_N(e.p, d)$) を求める．また，次のレコードを作成して PQ に投入する．

$$< e.p, e.dfs + d_N(e.p, d), e.dfs, PHASE3 >$$

最終的に，PQ から取り出されたレコードの $phase$ 値が PHASE3 の場合，sd 間の最短路距離が $e.dfs$ として求まったことになる．そこで，最短路探索処理を終了する．

3. 実験結果

本稿で提案した 3 つのアルゴリズム (SPFLM,SPFMM,SPFFM) の性能を評価するために，A*アルゴリズムとの比較実験を行った．それぞれのアルゴリズムは Java により実装した．使用計算機環境は，ハードウエアは Intel Core i7CPU 960，3.2GHz，9GB メモリー，OS は Windouws7 である．また，実験に用いた道路地図は 168km^2(MapS)，284km^2(MapM)，3,797km^2(MapL) の範囲の 3 種類である．これらの地図での各表のデータ量を表 2 に示す．

表 2: Data size (MB)

地図名	SPFLM	SPFMM	SPFFM	HEPV
MapS	2.6	6.7	14.5	30.1
MapM	11.3	28.7	70.1	376.1
MapL	65.8	166.6	400.0	8,287.6

図 2 は，MapS を用いて 2 点間の最短路を求めたときの処理時間を示している．HEPV，SPFFM は同様に高速な最短路検索が行える．また，SPFMM も同様な高速性を達成している．

図 3 は，MapM を用いて IER の枠組みで kNN 検索を実行したときの処理時間を示している．横軸は検索対象の POI の存在密度を示している．またこの図は $k = 10$ の結果を示している．SPFLM は s および d から，それぞれの境界ノードまでの最短路を A*アルゴリズムで決定するため POI の密度が下がると処理時間が増大する．一方，SPFMM 及び SPFFM ではそれを IBDT や NNDT を参照して決定するため，密度が低い時の処理時間も抑えられている．

4. むすび

本稿では，LBS での利用を目的に，部分的な距離のマテリアライズ化データを利用した最短路探索方式を提案した．この方式で利用するデータは，対象とする道路ネットワーク全域に対するマテリアライズ化や階層的なマテリアライズ化データより，大幅に少なく押さえることができる．また，分

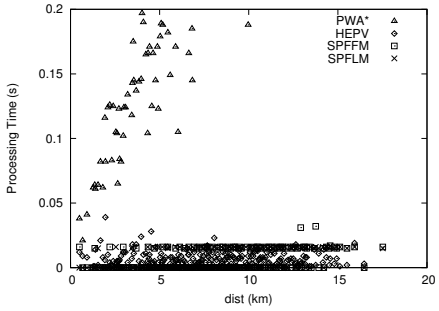

図 2: Processing time when s and d are placed on nodes (MapS)

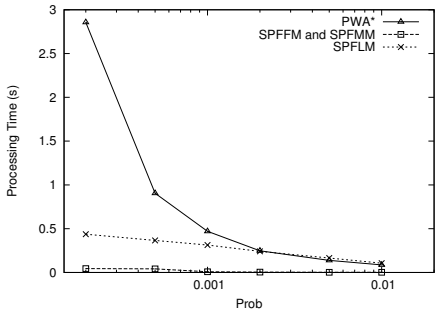

図 3: 10NN 検索の処理時間

割されたセル内の 2 点間距離をマテリアライズ化したデータのみを用いた方式 (SPFLM) でも，2 点間距離が大きい場合に A*アルゴリズムの 20 倍以上の高速化を達成している．

本稿で提案した 2 方式のうち，SPFLM は少ない前処理データを用いた最短路探索を行える．この方式を単に 2 点間の最短路探索に適用したとき，特に 2 点間の距離が大きい場合に，A*アルゴリズムに対して大幅な処理速度向上が見られる．一方，LBS で必要となる検索では多数の最短路を繰り返し求める場合が多い．その際には，隣接リスト管理の為の LRU バッファーのヒット率が向上することから，A*アルゴリズムの処理速度も相対的に向上する．本稿で示した kNN 検索は，そのような例であり，POI が密に存在する状況では A*アルゴリズムと SPFLM との速度差は小さくなる．一方，SPFMM ではそのような状況においても優位性が保たれている．