交通ネットワークに沿った最近接検索アルゴリズムの効率化

油井　真斗，　大沢　裕

An efficient nearest neighbor search algorithm on road network

Masato YUI，Yutaka OHSAWA

Abstract: Network distance is more practical for human activity analysis than Euclidean distance. For LBS (location based service), POI (point of interest) and road network are usually managed independently. Then, to calculate the distance along road network between two POIs, the operations to find a road segment which contains POI and to find POIs contained in a road segment are necessary. The algorithm named INE (incremental network expansion) has been proposed for several kinds of spatial operations based on road network distance according to this situation. This paper proposes data structure and algorithm suitable for INE.

Keywords: k-NN 検索(k-NN search)，位置情報サービス(location based service)，道路ネットワーク(road network)，空間インデックス（spatial index）

1. はじめに

位置情報サービス(LBS: location-based services)は，ユーザの現在位置とその位置に関連する様々な情報とを統合することによって，ユーザに付加価値を提供するサービスである[1]．LBS への関心は携帯電話の普及とともに高まっており，商業や緊急サービスの分野において普及しつつある．このサービスを実現するためには，携帯端末の測位技術や各種位置情報のデータベース化，経路探索や最近点検索などの各種空間演算が不可欠である．

最近点検索のためのアルゴリズムとして，Rousspoulos[4]や Hjaltason[2]らはユークリッド距離に基づく手法を提案している．しかし，LBS ユーザは多くの場合，目的地までの直線距離よりも移動する際の道のりや移動時間を重要視すると考えられる．なぜなら，人の移動はほとんどの場合道路や鉄道などの交通ネットワークに依存するためである．そのため，交通ネットワーク上での道のりが最短となる地物を検索するためのアルゴリズムが求められる．この実際の検索例を図 1 に示す．ここでは検索点(中央よりやや左下の点)からの道のりが小さい地物を 3 件探し，その位置とそこに至るまでの経路を強調表示してある．

Papadias ら[3]は，交通ネットワークに主眼を置いた検索アルゴリズムとして，INE (Incremental Network Expansion)を提案している．これは，道路ネットワークと検索対象となる地物をそれぞれ別個の空間インデックスで管理し，検索点周辺の道路ネットワークを順次展開しつつ，展開した道路上にある地物を求める手法である．検索対象となる地物はガソリンスタンドやホテルといった施設の経緯度情報を持つ点データである．この点データを本稿では POI(point of interest)と呼ぶ．このアルゴリズムでは，道路ネットワークと POI が独立したデータとして管理される．これは，データの動的な変化に柔軟に対応するためであり，一方のデータセットに対する

油井：〒338-8570 埼玉県さいたまま市桜区
下大久保 225　埼玉大学大学院理工学研究科
TEL(FAX):048-858-9181
email:s07mm333@mail.saitama-u.ac.jp

変更がもう一方のデータセットに影響を与えないという利点がある．しかし，道路と POI との空間的な位置関係をその都度演算によって求めなければならないという問題が生じる．この演算には R 木などの空間インデックスを用いるため，処理コストが大きい．

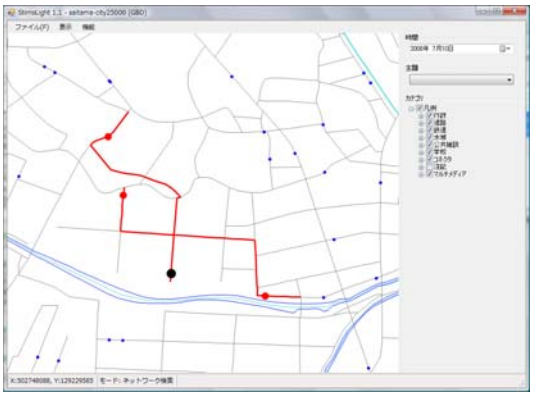


図 1．検索例

本稿では，POI と道路ネットワークとの対応関係を明示させることによってデータの管理方法を変え，全体の処理コストを低減する手法を提案する．現実世界において，POI 情報は時間の流れと共に常に変化し，その変更頻度は道路ネットワークのそれより遥かに高い．提案手法は，現実世界のこのような性質を考慮し，POI の変更が道路ネットワークのデータに影響を与えないように設計されている．

2. 関連研究

Papadias らは，道路ネットワークにおける最近点検索のためのアルゴリズムとして，INE(Incremental Network Expansion)を提案している．道路ネットワークと POI はそれぞれ独立したデータとして管理され，R 木などの空間データ構造によってそれぞれインデックス化される．

INE は，まず検索点 q がどの道路セグメント上にあるかを求める．求めた道路セグメント上に POI が存在すれば，それをすべて結果リストに入れる．発見した POI の数が検索個数 k に満たない場合は，道路セグメントの両端点を優先順位付きキューQ に入れる．このときの優先度は，q から端点までのネットワーク距離の小さい順である．

次に，優先度の最も高い端点を Q から取り出し，その点に接続しているすべての道路セグメントを求める．求めた各道路セグメント上に POI が存在すればそれらを結果リストに入れ，各道路セグメントのもう一方の端点を Q に入れる．そして Q から次の端点を取り出す，という処理を繰り返す．終了条件は，POI が k 個見つかっており，かつ k 番目の POI までのネットワーク距離が次に評価すべき端点までの距離以下となることである．

このアルゴリズムでは，道路セグメントと POI(q)との空間的な関連性を求めるための演算が必要である．以下に必要な演算を挙げる．

・*checkEntity(seg, p)*

点 p が道路セグメント *seg* 上に位置しているかどうかを判定する．まず，*seg* の MBR を用いてフィルタリングし，p がその MBR の内部にあれば，seg の形状を用いて厳密な判定を行う．デジタル化による誤差があるため，p が *seg* 上にある場合でも p と *seg* との距離は 0 にならない場合がある．そこで，ある閾値 *dT* を定めておき，p と *seg* との距離が *dT* 以内であれば，p は *seg* 上にあるとみなす．

・*findSegment(p)*

点 p がある道路セグメント上に存在している場合に，その道路セグメントを取得する．この検索は道路ネットワークを格納している R 木に対して行われる．判定に *checkEntity* を用いる．

・*findEntities(seg)*

道路セグメント *seg* 上にあるすべての POI を取得する．あらかじめ *seg* の MBR を用いて候補を絞り込むことによって，計算に要するコストを削減する．判定に *checkEntity* を用いる．

以下に INE アルゴリズムの流れを示す．$d_N(q,n)$ は点 q からノード n までのネットワーク距離である．

[アルゴリズム INE(q, k)]

Step1: $n_i n_j \leftarrow$ *findSegment(q)*

Step2: $S_{cover} \leftarrow$ *findEntities(*$n_i n_j$*)*

Step3: S_{cover} の中からネットワーク距離の小さいオブジェクトを順に k 個取り出し，これを $\{p_1,\mathrm{K}\ ,p_k\}$ とする．

Step4: $d_{N\max} \leftarrow d_N(q,p_k)$ (もし p_k が存在しない場合は∞)

Step5: $Q \leftarrow < (n_i, d_N(q,n_i)), (n_j, d_N(q,n_j)) >$ (Q は優先順位付きキュー)

Step6: Q からネットワーク距離最小のノードを取り出し，これを n とする．

Step7: $d_N(q,n) < d_{N\max}$ となる間，Step8 から Step13 までを繰り返す．

Step8: n と接続しており，まだ訪問していないすべてのノード n_x に対して，Step9 から Step12 までの処理を行う．

Step9: $S_{cover} \leftarrow$ *findEntities*($n_x n$)

Step10: S_{cover} をもとに $\{p_1,\mathrm{K}\ ,p_k\}$ を更新する．

Step11: $d_{N\max} \leftarrow d_N(q,p_k)$

Step12: $(n_x, d_N(q,n_x))$ を Q に追加する．

Step13: Q から次のノード n と取り出す．

3. 提案手法

INE アルゴリズムでは，道路ネットワークと POI は独立したデータとして管理され，両者の関係を明示する情報がない．

そのため，どの道路セグメントにどの POI が属しているかという情報をその都度空間演算によって求めなければならない．前節の *findEntities* がこれに該当する．この演算は，道路セグメントの MBR を用いて，POI を管理する R 木の検索により実行される．葉ノードにおいては，各 POI が MBR に包含されているかどうかを判定し，もし包含されていればポリラインを用いて厳密な判定を行う．この処理にはポリラインの頂点数に比例するコストがかかる．また，空間インデックスに対して問い合わせを行うため，道路セグメントが空間的な広がりを持っている場合に多くのノードを訪問しなければならない場合がある．

POI と道路セグメントとの関係を明示することによって，この演算に要するコストの削減を図る．各 POI に対して，その POI が属している道路セグメントの ID と，その道路セグメントの始点および終点からの道のりを補助情報として持たせる．管理する POI の形式は以下のとおりである．

$$(x, y, roadID\ , dist_S , dist_E)$$

x,y は POI の座標値，*roadID* は POI が属する道路セグメントの ID，$dist_S , dist_E$ はそれぞれ道路セグメントの始点および終点からの道のりである．この変更によって，*x* 座標，*y* 座標の 2 次元データとして表されていた POI を，*roadID* をキーとする 1 次元データとして管理することができる．そこで，POI を B+木を用いて管理する．R 木から B+木に変更することの利点は次のとおりである．

- B+木は R 木と比較して 1 エントリあたりに必要なデータ量が少ない．そのため 1 ノードあたりのバイト数を統一した場合，1 ノードあたりのエントリ数を増やすことができ，木全体のノード数を低く抑えることができる．
- 道路セグメントに属する POI を取得する際に空間的な計算を行う必要がなくなり，ID の一致検索を行うだけで必要な POI を取得できるようになる．

このような管理方法を採ることによって，前節で述べた処理 *findEntities* を以下に示す処理 *findEntities'*に置き換えることができる．

・*findEntities' (seg)*

segID で示される道路セグメント上に存在するすべての点データを求める．POI を管理している B+木を用いて，キーが *segID* に一致するすべての POI を取得できる．この処理において *checkEntity* の呼び出しは不要である．

4. 実験

4.1. 実験方法

従来方式と提案方式との比較実験を行う．実験環境は，Core 2 Duo 2.66GHz CPU，2GB RAM,OS は Windows Vista である．使用した地図データは，268,245 個の道路セグメント(この数を $|N|$ とする)からなる数値地図である．POI データは擬似乱数を用いて作成した．

道路ネットワークを R＊木[1]で，POI データを

R＊木と B+木でそれぞれインデックス化した．R＊木のノードサイズを 50，B+木のノードサイズを 100 とし，1 ノードのデータサイズを約 1,200B に統一した．

検索個数 k を 1 から 20 まで増加させた場合の両方式の性能を，アクセスノード数，検索時間の観点から比較する．

4.2. 実験結果

図 2 は，$|S|=|N|$($|S|$は POI の個数)に固定し，k を 1 から 20 まで変化させた場合のアクセスノード数および検索時間を比較した結果である．同図(a)は POI および道路ネットワークのそれぞれのインデックスに対するアクセスノード数を個別に集計し，その合計を積み上げ表示したものである．両方式を比較すると，POI インデックスのみを見ると 40%程度，全体で 25%程度アクセスノード数が改善しており，この傾向は k が増加してもほぼ変わらないことが分かる．処理時間を比較した結果，全体を通して約 35%高速化されており，ノードアクセスの改善効果よりも大きな改善効果が得られていることが分かる．これは，従来方式では空間演算が必要であった処理が提案方式によって ID の一致検索に置き換わったことにより，CPU コストが削減されたことが要因であると思われる．

5. まとめ

本稿では，道路ネットワークにおける最近接検索アルゴリズムとして知られる INE を効率化するための手法を提案した．この手法は，現実世界における地物の変化に柔軟に対応できるという従来方式の利点を踏まえており，POI 情報の変化が道路データに影響を及ぼさないように設計されている．両方式の比較実験により，提案方式によって I/O コスト・CPU コストが低減し，検索の高速化が可能であることを示した．

参考文献

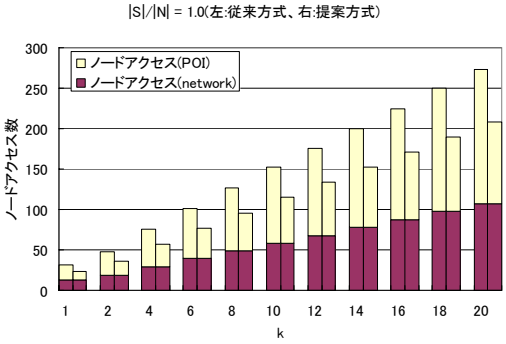
[1] Nobert Beckman, Hans-Peter Kriegel, Ralf Schneider and Bernhard Seeger. The R*-tree: an efficient and robust access method for points and rectangles. *SIGMOD Rec.*, 19(2):322-331, 1990.

[2] R. Hjaltason and Hanan Samet. Distance Browsing in Spatial Databases. *ACM Trans. Database Syst.*, 265-318, 1999.

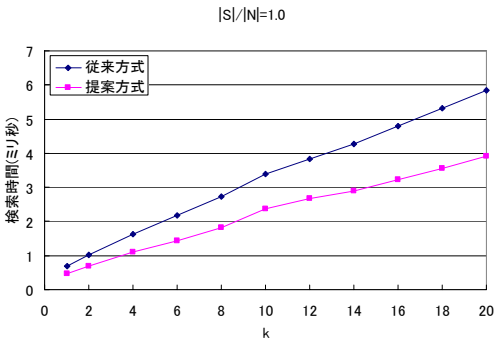
[3] Dimitris Papadias, Jun Zhang, Nikos Mamoulis and Yufei Tao. Query Processing in Spatial Network Databases. *Proceeding of the 29th VLDB Conference*, 2003.

[4] Nick Roussopoulos, Stephen Kelly and Frederic Vincent. Nearest Neighbor Queries. *Proceedings 13th International Conference on Very Large Data Bases*, pages 71-79, 1995.

[5] Jochen H. Schiller and Agnes Voisard. *Location-Based Services*. Morgan Kaufmann Pub., 2004.



(a) アクセスノード数(左：従来方式，右：提案方式)



(b) 処理時間

図 2. 実験結果($|S|/|N|=1$)