道路網上での距離に基づく *k*-NN 経路探索

橋本　知宜・Aye Thida Hlaing・藤野 和久・大沢 裕・曽根原 登

A Study on k-NN Route Search Based on Road-Network Distance

Tomonori Hashimoto, Aye Thida Hlaing, Kazuhisa Fujino, Yutaka Ohsawa and Noboru Sonehara

Abstract: Efficient k-NN query methods have been studied actively for LBS (location based services). k-NN query finds POIs (point of interest) located at neighboring to the query point q. When the distance is measured based on the road network distance, the query cost becomes huge. Then, this paper proposes efficient algorithms based on road network distance, adopting A* algorithm and bi-directional search. We do not require preprocessing for these methods and achieve several times calculation with higher speed comparing with conventional methods.

Keywords: OSR(Optimal Sequential Route)，POI(Point of Interest)，経路探索(Path Search)，道路網(Road Network)，移動体(Moving Object)

1. はじめに

ある位置が指定され，その位置からの距離が短い空間オブジェクト(POI:point of interest)を最短のものから順に任意個(*k*個）求める探索は，*k*-NN(nearest neighbor)探索と呼ばれ，活発に研究されてきた．また，距離の尺度として道路網などのネットワーク上を移動する距離を用いた*k*-NN探索アルゴリズムも各種提案されている．

k-NNアルゴリズムは，*k*の値を固定した探索のほか，*k*の値を順次増加させる探索(インクリメンタル探索）も重要となる．例えば，SharifzadehらのOSR(optimal sequenced route)探索[1]では，PNEアルゴリズムにおいてこのインクリメンタル探索を必要としている．一般に，*k*-NN探索をPOI探索に用いる場合，距離以外の属性が探索条件を満たさない場合に，次に探索条件を満たす次に隣接するPOIの探索が必要となる．例えば，近傍のレストラン探索において探索結果に定休日のものが含まれる場合，次に隣接するPOIの探索を繰り返す必要がある．

ネットワーク上の距離に基づく*k*-NN探索においては，ネットワーク上での距離で探索点に隣接する*k*個のPOIを決定する他に，その各々のPOIへの探索点からの経路が必要となる場合も多い．例えば，カーナビやマンナビにより目的のPOIへ誘導する場合に，この経路が必要となる．

また，POIの*k*-NN探索において，POIの位置以外の属性が指定される場合が多い．例えばレストランの探索において，イタリアンレストランが指定されたり，その予算額が指定されたりする．後に述べるように道路網上での*k*-NN検索には前処理により道路網上での

大沢　裕 〒338-8570 埼玉県 さいたま市下大久保 255
埼玉大学大学院理工学研究科数理電子情報部門
Phone: 048-858-3717
E-mail: ohsawa@mail.saitama-u.ac.jp

距離に基づく索引を作成しておく方式と汎用的な空間索引のみを用いる方式がある．特別な索引を作成しておく方式を採用する場合，索引作成に際してPOIの種別を細分化しておく必要がある．また道路網上での距離に基づく索引のために多くのデータ量を必要とする傾向がある．

そこで，本稿では以下の目標の基に開発した道路網上の距離に基づく高効率な*k*-NN探索アルゴリズムについて提案する．

- インクリメンタル検索が可能なこと
- 道路網上での位置に依存した検索のためにR木などの空間索引のみを用いる
- 特殊な索引構造を必要としないことから，道路網の時間的変化(時間による一方通行規制や工事による閉鎖)に対応可能

2. 関連研究

道路網上の距離を尺度とした *k*-NN 探索に関しての研究は，筆者らの知る限りにおいては Papadias らの研究が最初であると思われる．そこでは *k*-NN 探索のために，INE(incremental network expansion)および IER(incremental Euclidean restriction)アルゴリズムが提案されている．これらは R 木などの空間インデックス以外には特殊な構造を用いない方式であり，またインクリメンタル探索も可能である．従って，求まる POI 毎に非空間属性が探索条件を満たすか確認することが可能である．また，道路網が時間的な変化の情報を有していれば，時間的に変わる(一方通行などの)規制に対応した経路を求めることができる．

一方，道路網上での距離に基づき何らかの索引構造を構築しておく方式も提案されている．これらの代表例として Kolahdouzan ら[3]のネットワークボロノイ図(以下 NVD と略記）や，Hu ら[4]の距離記号(network signature)に基づく方式や Sankaranarayanan・Sament[5]による距離神託(distance oracle)による方式がある．

3. A*アルゴリズムと Dijkstra 法による *k*-NN 検索アルゴリズム

3.1 基本アルゴリズム

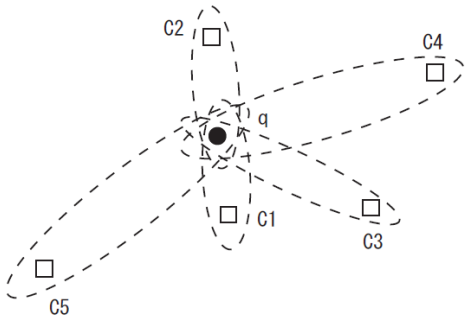
提案方式を述べる前に，その基本となるBA(basic algorithm)を説明する．BAの基本はユークリッド距離に基づく対象候補の生成過程とA*アルゴリズムにより実際の道路網上での距離を求める検証過程からなる．検索点をqとするとき，qにユークリッド距離で近接するk+1 (但しkは求めるNNの数）個のPOIを求める．この検索は, R木上での最適優先探索[6]により求まる．道路網での距離に基づくNNの候補集合をCとし，k個のNNを$c_1 \ldots c_k$としてCに入れる．一方， k+1 番目のNNはc_{nxt}とし，Cとは別に保持する．

各c_iからはqに至る最短路をA*アルゴリズムを同時並行的に実行して探索する．即ち，A*アルゴリズムの開始ではc_i ($c_i \in C$)とqのユークリッド距離($d(c_i,q)$)をコストとしてPQに投入し, そのコストが最小のものから順に取り出しノード展開を行う. PQから取り出されたノードをnとするとき，nに隣接する全てのノード集合Mを求め，その要素毎にVal=$d(c_i,m)$+$h(m,q)$ ($m \in M$)を求める．ただし，$d(c_i,m)$はc_iからmに至る道路網上の距離を表し， $h(m,q)$はmとqの間のユークリッド距離を表す. そしてValを尺度として各ノードmをPQに投入する．この処理をPQから取り出されたノードがqに一致するまで繰り返す．A*アルゴリズムによる探索がqに達したc_iはqに至るパスとともに，結果集合Rに登録する．これをk個のPOI からのパス全てが求まるまで繰り返す.

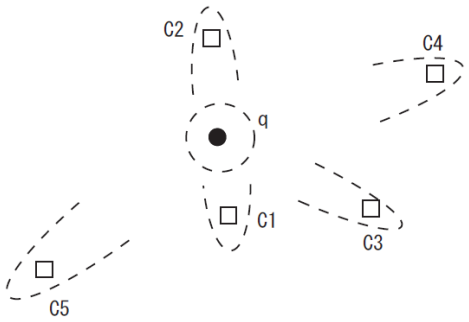
一方, PQから次のノード展開のために取り出したノード(n)のValの値が$h(c_{nxt},q)$より大きくなったとき，c_{nxt}がk-NNに含まれる可能性がある．そこで， c_{nxt}をCに含め，新しいc_{nxt} ($|C|$+1 番目のNN)を得る．即ちCは探索途中で増大する．

以上で述べた探索の概要を図 1(a)に示す．図中破線

で囲んだ楕円領域中のノードがA*アルゴリズムにより展開されることになる．最初はk個のc_iがPQに入れられた状態で探索が開始する．その際に，A*アルゴリズムの性質によりqに近いc_iからのパスがPQから取り出されてノード展開が進むことから経路長が短い経路のノード展開が先に進むことになる．またこの図に見られるように，BAではqの周辺のノードが複数回展開されることになる．これは各c_iからの探索が独立して実行されるためである．これが探索全体でのノード展開数を増大させることになる



(a) BA



(b) BDA

図１　基本アルゴリズムと双方向探索

3.2　基本アルゴリズムと双方向探索

q周辺のノード展開を抑制すれば，探索時間を短縮できる．そのために，Dijkstra法とA*アルゴリズムの併用を考える．即ち，各c_iからはqを目指したA*アルゴリズムによる探索を行い，一方qからはDijkstra法による探索を行うことである．これをBDA(bi-directional search by Dijkstra and A* algorithm)と呼ぶ．

BDAでは2種類のPQ，即ちqからDijkstra法によるノード展開を管理するPQ，PQ_Dと，c_iからA*アルゴリズムによるノード展開を管理するPQ，PQ_Aを用意する．PQ_Dには展開したノードをqからそのノードへの距離を尺度として投入する．一方，PQ_Aには各c_iからqを目指すノード展開について，Valを尺度として投入する．このとき，次に展開するノードは，PQ_DとPQ_Aをピークし，PQ_Dの$r(q,n)$とPQ_Aの$r(c_i,n)$を比較し，その値が小さい側のノードを取り出し展開する．このノード展開法により，qを中心とする展開と全てのc_iからのノード展開が並行して進行する．

PQ_Dから取り出したノードは展開後，終了集合(CS_D)に登録し，以降qから開始されたノード展開では2度とそのノードが展開の対象にならないように制御する．一方，PQ_Aから取り出したノードは展開後，c_iそれぞれに対応する終了集合(CS_i)に登録し，同じc_iから開始したノード展開では2度と展開の対象にならないように制御する．即ち，c_iからのノード展開では別のc_iからの展開におけるノード展開の対象となり得る．これは，各c_iからqに向かう最短路を探索するために必要である．

BDAもBAと同様に，PQ_Aから取り出したノードのValが$h(c_{nxt},q)$より大きくなった時点で，c_{nxt}をCに追加し，ユークリッド距離で次のNNをc_{nxt}として得る．

BDAにおけるノード展開の概要を図 1(b)に示す．c_iからのA*アルゴリズムによるノード展開とqからのDijkstra法によるノード展開が進行するが，qの近傍領域はDijkstra法により 1 度ノード展開されるのみとなり，多重のノード展開を抑制できることから，探索効率の向上が望める．

以上で述べた，BDAをアルゴリズム 1 に示す．まず結果集合Rを空集合で初期化し，PQ_Dには検索開始点q

を入れる. 2 行目ではユークリッド距離を尺度としてqのk-NNを探索し, 結果を集合Cに入れ, 検索対象のPOI数mにkを代入する. 3 行目ではk+1 番目のNNを探索し, それをc_{nxt}に代入している.

4 行目以降を繰り返す. まず 5 行目では, PQ_AとPQ_Dの先頭の値を調べ, ca<cdであれば$c_{n.i}$からのA*アルゴリズムによるノード展開を行う. 取り出したノードnの探索が開始したPOI(n.i)を調べ,そのPOIの終了集合にnを追加する(8 行目). 9 行目では, qからDijkstra法で展開した終了集合中にnが含まれているか調べ,もし存在すれば$c_{n.i}$からとqからのパスが出会ったことになるため, 10 行目でそのパスを得て,それをRに追加している. 11 行目では結果集合Rの要素数がk以上となったとき, Rを返し終了している.

15 行目では, PQ_Aから取り出したノードのコストがc_{nxt}とq間のユークリッド距離より大きいか調べ, その場合にはc_{nxt}を候補集合Cに加えている.また, c_{nxt}を次のNNで更新している. 19 行目では, nのノード展開を行い, nに隣接するノードを全てPQ_Aに追加する.

22 行目以降は, q側からのDijkstra法によるノード展開である. まずPQ_Dからノードnを得て, q側からの終了集合CS_Dにそれを追加する. 24 行目から 31 行目は, そのノードにPOI(c_i)側からのノード展開が到達しているか
調べている. もし, 到達していれば, qとc_iを結ぶパスを得て, それをRに追加する. その結果, 得られたパスの数がkを超えれば, 結果集合を返して終了する.
32 行目は, nに隣接するノードを展開し, その結果全てをPQ_Dに追加する.

3.3　探索点からの A*アルゴリズム探索

前節で述べた BDA アルゴリズムは, 2 種類の PQ と多数の終了集合(CS)を必要とした. 本節で述べる INEH(incremental node expansion with heuristics)は, 検索点から各 POI に向けた片方向探索を行うため, PQ と CS を 1 つずつに抑えることができる.本アルゴリズムの概要を図 2 に示す.

Algorithm 1 BDA

```
1: R ← ∅, PQ_D ← {q}
2: C ← NN(k),m ← k
3: c_nxt ← nextNN()
4: loop
5:   ca ← peekPQ_A(),cd ← peekPQ_D()
6:   if ca < cd then
7:     n ← deleteMinPQ_A()
8:     CS_n.i ← CS_n.i ∪ n
9:     if n ∈ CS_D then
10:      R ← R ∪ getPath(n)
11:      if |R| ≧ k then
12:        return R
13:      end if
14:    else
15:      if n.cost > h(c_nxt, q) then
16:        C ← C ∪ c_nxt,m ← m + 1
17:        c_nxt ← nextNN()
18:      end if
19:      expandNodeA(n)
20:    end if
21:  else
22:    n ← deleteMinPQ_D()
23:    CS_D ← CS_D ∪ n
24:    for i = 1 to m do
25:      if n ∈ CS_i then
26:        R ← ∪getPath(n)
27:        if |R| ≧ k then
28:          return R
29:        end if
30:      end if
31:    end for
32:    expandNodeD(n)
33:  end if
34: end loop
```

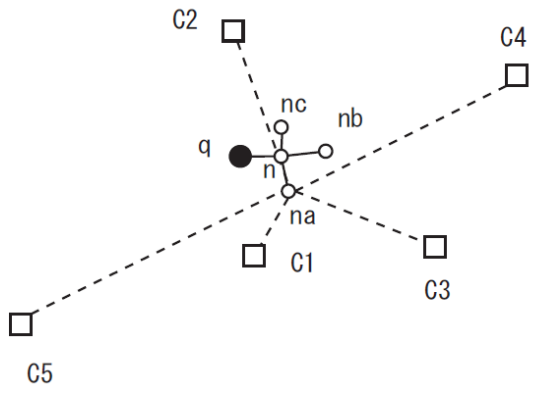


図 2　INEH アルゴリズム

Algorithm 2 INEH

1: $R \leftarrow \emptyset$, $PQ \leftarrow \{q\}$
2: $C \leftarrow NN(k)$,$m \leftarrow k$
3: $c_{nxt} \leftarrow nextNN()$
4: **loop**
5: $n \leftarrow deleteMin()$
6: $CS \leftarrow CS \cup n$
7: **if** $n \in C$ **then**
8: $R \leftarrow R \cup getPath(n)$
9: **if** $|R| \geqq k$ **then**
10: **return** R
11: **end if**
12: $C \leftarrow C - c_i$
13: **end if**
14: **for all** $nn \in neighbor(n)$ **do**
15: deside c_i which gives minimum $h(n, c_i)$
16: $PQ \leftarrow PQ \cup < d(q, nn) + h(nn, c_i), nn, c_i >$
17: **end for**
18: **end loop**

4. 実験結果

本稿で述べたk-NN探索アルゴリズムの性能を評価するために，実際の道路地図と擬似乱数により発生させたPOIを用いて実験を行なった．ここで用いた道路地図は，数値地図 25000 に含まれるさいたま市全域である．図 3 に実験で用いた道路地図を示す．また，母点は全て道路上に発生させた．母点の密度(p)は，道路セグメントあたりの存在確率で表している．即ち，p=0.001 とは 1000 本の道路セグメント(主として交差点間)に 1 個の割合で存在することを示している．本実験では，INE，VN^3，および提案方式を比較した．

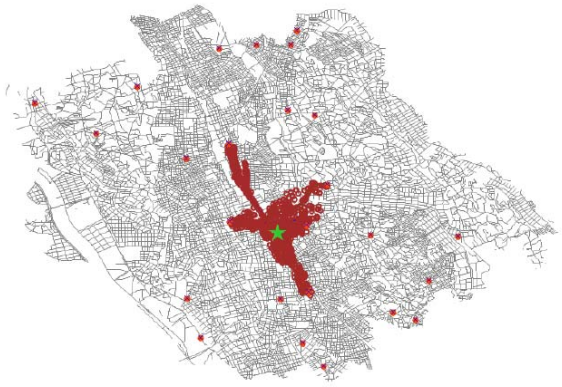


図 3　提案方式におけるノード展開範囲

図 4 は，k=5 の場合におけるPOI密度とノード展開の数の関係を示している．この図に見られるように，POI密度が低いとき，PNEは大量のノード展開を必要とする．一方，VN^{3}は常に少ないノード展開数にとどまっている．提案方式は，両者の中間の性能を示しているが，INEに比べて 1/4 以下のノード展開数にとどまっている．

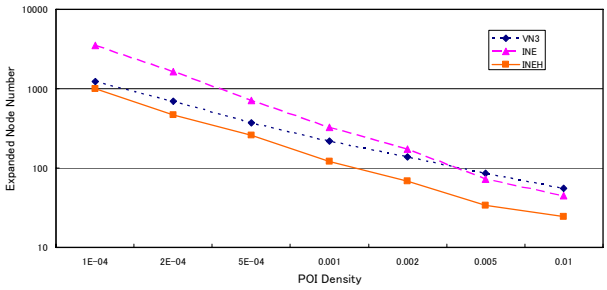


図 4　ノード展開数の比較(k=5)

図 5 は，POI密度を 5×10^{-4}とし，kの値を変えたときのノード展開数を示している．Kの増加により，全ての方式で展開ノード数が増加している．しかし，ここでも提案方式はINE, VN^3 の中間的な性能を示している．

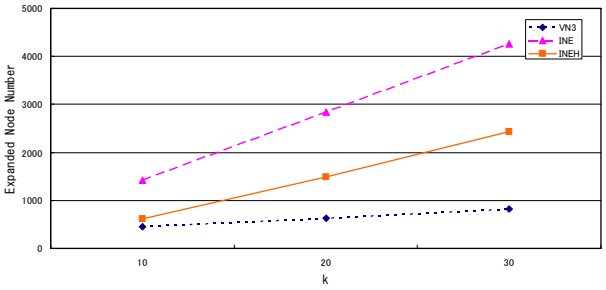


図 5　ノード展開数と k との関係

図 6 は，探索終了時にヒープ中に存在する要素数と，POI密度の関係を比較している．VN^{3}はINEや提案方式に比してk-NN探索中にヒープ中の要素数が増大する．一方，提案方式とINEではVN^{3}に比して 1/8 程度に納まっている．

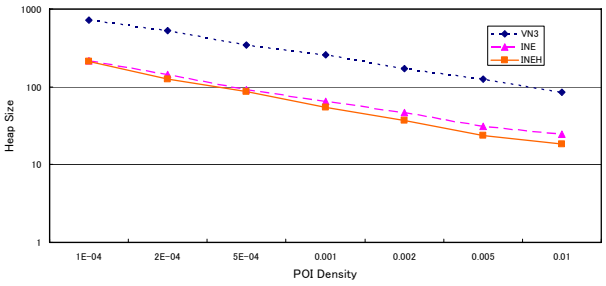


図 6　ヒープ中の要素数と POI 密度の関係

5. おわりに

本稿では，VN^3 構造を簡約化した構造を提案し，その構造を用いた*k*-NN検索のアルゴリズムを示した．また実験により，提案方式の有効性を実証した．

近年，LBS での利用を目的として，移動経路を含む POI 探索方式や OSR など旅行計画アルゴリズムの研究が活発に行なわれている．本方式のそれらの探索への適用は今後の課題である．

謝辞　本研究は，文部省科学研究費補助金（基盤研究（C）21500093）及び日本ディジタル道路地図協会の補助を得た．

参考文献

[1]M.Sharifzadeh, M.Kolahdouzan, C.Shahabi: “The Optimal sequenced route query” technical report , Computer Science Department , University of Southern Calfornia (2005)

[2]D.Parpadias, J.Zang, N.Mamoulis, Y.Tao : “Query processing in spatial network databases”,Proc.29th VLDB,pp.790-801(2003)

[3]M.Kolahdouzan, C.Shahabi: Voroni-Based K Nearest Neighbor Search for Spatial Network Databases, Proc. 30th VLDB Conf., pp.840 -851 (2004)

[4] H.Hu, D.L.Lee, V.C.Lee: “ Distance indexing on road networks, VLDB’06, pp.894-905 (2006)

[5]J.Sankaranarayanan, H.Samet: “Distance oracles for spatial networks”, IEEE International Conference on Data Engineering, pp.652-663 (2009)

[6]G.Hjaltason, H.Samet: “Ranking in spatial databases”,Proceedings of the 4th Symposium on Spatial Databases, pp.83-95(1995).