D-024

道路網上での ANN 探索法
An Aggregate Nearest Neighbor Query Method on Road Network Distance

トウトウ †　**大沢　裕** †　**曽根原　登** ‡
Htoo Htoo　Yutaka Ohsawa　Noboru Sonehara

1. まえがき

ANN(aggregate nearest neighbor) 検索は，検索対象となる POI(point of interest) の集合 P と，複数個の検索点の集合 Q が与えられたとき，ある集約関数 f の基で (距離などの) コストが最少となる $p \in P$ を求める検索である．以下 1 個の p を求める検索を 1ANN，コスト最少のものから k 個求める演算を kANN と呼ぶ．この検索は，Papadias ら [1] により最初 group nearest neighbor と名付けられ提案された．その後，道路網上での距離に基づく ANN 検索法が Yiu ら [2] により提案されている．Yiu らは複数のアルゴリズムを比較し，その中で IER(incremental Euclidean restriction) 法が検索速度の観点から優れていることを示している．

この IER は，まず空間索引構造 R 木を用いてユークリッド距離に基づく ANN を検索し，その結果を A*アルゴリズムを用いて道路網上での距離により検証する方式である．この検証の際に対象点 p と Q の各々の要素間の最短経路を探索し，その距離を求める必要があるため，A*アルゴリズムを $|Q|$ 回起動する必要がある．道路網上ノードの隣接関係が隣接リストで表現されているとき，あるノードに直接隣接するノードを隣接リストを参照して求める処理をノード展開と呼ぶことにすると，この繰り返しにおいては p の近傍ノードが複数回ノード展開の処理対象となる．

本稿では，1 つの検索点から複数の点集合の各要素への最短路を同時に求めることができるアルゴリズム SSMTA*(single-source multi-target A*) を用いて，IER による ANN 探索を高速化する方式について述べる．

2.SSMTA*アルゴリズム

ANN 検索で対象とする POI は常に道路網のノードに存在するとは限らない，また検索点 Q もノード上に指定されるとは限らない．しかし，以下では説明を単純化するため，POI と Q は全て道路網上のノードに存在するものとする．但し，この制約を除くことは容易である [2]．

先ず，SSMTA*の概要を図 1 を用いて説明する．q を検索点とし，p_1 から p_4 までが集合 P の要素とする．現在 q から n までの経路が求まっており，現在の注目ノードを n とする．n には n_a から n_c までの 3 つのノードが隣接している．まず n_a に注目する．q から n を経由して n_a に達する道路網上の経路長は，$d_N(q,n)+d_N(n,n_a)$ である．また，n_a から P 中の各要素へのユークリッド距離の最小値を $d_E^{min}(n_a,P)$ と表わすことにすると，この図では p_1 までの距離が最も短いことから $d_E^{min}(n_a,P)=d_E(n_a,p_1)$ となる．そこで，q から n を経由していずれかの p に至る経路長のコストとして，$Cost=d_N(q,n)+d_N(n,n_a)+d_E^{min}(n_a,P)$ を用いる．ノード n に隣接する他の n_b，n_c に対してもこのコストを求め，優先順位付きキュー (PQ) に追加する．

A*アルゴリズムでは，$Cost$ が小さい順に取り出される PQ を用いる．探索を制御する PQ のレコードを以下に示す．

$$< Cost, N_C, N_P, d_N(q,N_C), RLink(N_P,N_C) > \qquad (1)$$

ここで，N_C は現在の注目ノードであり，N_P は q から N_C に至る経路上で N_C の直前のノードである．また，$d_N(q,N_C)$ は q から N_C に至る経路の道路網上の距離を表わす．$RLink(N_P,N_C)$ は N_P と N_C を結ぶ道路セグメント (へのポインタ) である．例えば，図 1 において N_C は n_a，N_P は n，$d_N(q,N_C)$ は $d_N(q,n_a)$，$RLink(N_P,N_C)$ は n と n_a を結ぶリンクがそれぞれ対応する．

†埼玉大学
‡国立情報学研究所

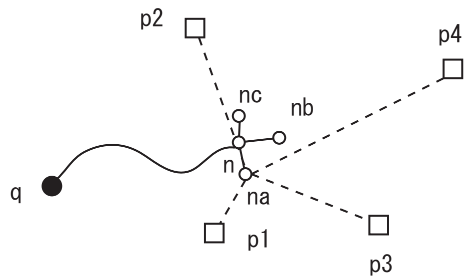

図 1: SSMTA*アルゴリズム

SSMTA*アルゴリズムの開始時には，

$$< d_E^{min}(q,P), q, -, 0, - > \qquad (2)$$

を PQ に入れる．

SSMTA*アルゴリズムの動作は，基本的には 2 点間の A*アルゴリズムと同じである．PQ から $Cost$ 最少の要素を取り出し，その要素が既に処理したノードでないか調べ，まだ処理されていないノードであれば，それに隣接する全てのノードに対して，式 (1) を作成して PQ に追加する処理を繰り返す．既に処理されたことを判断するため，クローズドセット (CS) に PQ から取り出されたレコードを追加する．レコードが PQ から取り出される度に，そのレコードの CS 中での存在が確認される．もし CS 中に存在すれば，そのノードは既に処理されている為，次のレコードを PQ から取り出す．CS では，N_C の存在確認が重要となる為，式 (1) の N_C をキーとする探索が高速になる構造（Hash など）で構成される．

PQ から取り出されたレコードを e とするとき，$e.N_C$ が $p \in P$ と一致すれば，1 つの POI が求まったことになる．そこで，p を P から取り除く．また，p から q までの経路の探索は，Dijikstra 法などと同じである．CS でまず $e.N_C$ が探され，そのレコードの $e.N_P$ を $r.N_C$ として持つレコード r を探す処理を，$r.N_C$ が q に一致するまで繰り返す．

ここで述べたアルゴリズムをアルゴリズム 1 に示す．

Dijkstra アルゴリズムや，2 点間の A*アルゴリズムでは PQ から取り出されたノードが CS に入れられた時点で，始点 q からそのノード n への最短路が確定し，後にその距離が変更されることはない．しかし，SSMTA*アルゴリズムでは CS 中のノード n の距離が変更される場合があり得る．アルゴリズム中の 7 行目はそのような状況に対応するためのものである．

3.ANN 探索への適用

道路網上での距離で ANN を求めるアルゴリズムとして，Yiu ら [2] は各種アルゴリズムを提案し，その中で IER が最も性能的に優れることを示している．基本的な考え方は，Papadias ら [3] の MBM(minimum bounding method) によりユークリッド距離での ANN を求め，その結果を道路網上での距離で検証する処理を繰り返す方式である．ユークリッド距離での ANN は R 木をたどることにより求められる．

求まった候補に対する道路網上での距離による検証は，次のように行われる．

ANN の探索点の集合を Q とし，ユークリッド距離での探索で求まった ANN を p とする．集合関数に sum（距離の総和）を用いた場合，各探索点から p への距離の和は，$\sum_{i=1}^{|Q|} d_N(p,q_i)$ である．Yiu らは $d_N(p,q_i)$ の計算に 2 点間の距離を求める A*アルゴリズムを用いている．以下，この

Algorithm 1 SSMTA*

$R \leftarrow \emptyset$
$d_{min} \leftarrow min(d_E(q, p_i), p_i \in P)$
$enqueue(< d_{min}, q, -, 0, - >)$
loop
  $e \leftarrow deleteMin()$
  if $CS.Contain(e.N_c)$ **then**
    $CS.renew(e.d_N(q, e.N_P))$
  end if
  $CS.add(< e.N_C, e.N_P, e.d_N, e.RLink >)$
  if $e.N_C \in P$ **then**
    $R \leftarrow R \cup < e.N_C, getPath(e.N_C) >$
    $P \leftarrow P - e.N_C$
    if $P = \emptyset$ **then**
      return R
    end if
  end if
  for all $nn \in neighbor(e.N_C)$ **do**
    decide $p(p \in P)$ which gives $d_E^{min}(nn, P)$
    $d_N \leftarrow d_N(q, e.N_C) + d_N(e.N_C, nn)$
    $enqueue(< d_N + d_E^{min}(N_C, P), nn, e.N_C, RLink(N_c, nn) >)$
  end for
end loop

アルゴリズムを ANN0 と呼ぶ．もしある POI(p_j) への Q の各要素からのユークリッド距離による総和が，ANN 候補として得られている POI(p_i) への道路網上での距離による総和より小さい場合，p_j は ANN 候補となり得る．そこで，このような場合には p_j も道路網上での距離による検証に加えなければならない．つまり，ANN 探索法はインクリメンタルであるため，次の候補を求めて道路網上での距離で検証する過程を，PQ から得られる距離の総和が道路網上での距離の総和より大きくなるまで繰り返す．即ち 1ANN を求める場合でも，ユークリッド距離による 1ANN を求め，それを道路網上での距離で検証するのでは十分ではなく，多数の ANN 候補に対する検証を必要とする．

そこで，前述の SSMTA*を用いることにより多数の検索点から道路網上での距離の総和を求める計算は高速化できる可能性がある．具体には，SSMTA*の適用法として次の 2 つの方式が考えられる．

(1) p を探索開始点として，Q の全ての要素を目的地とする SSMTA* を適用する．(以下，ANN1 と呼ぶ)

(2) Q の各々の点を探索開始点として，新たな p が求まる度に，各々の q_i から新しいターゲットとして，その p を追加した SSTMA*を適用する．(以下 ANN2 と呼ぶ)

ANN は評価に用いる集約関数の値が最小の結果を 1 つ求める (1ANN) だけではなく，値が最小のものから k 個の結果を得たい (k-ANN) 場合も多い．この状況で，上記の 2 つの方式には適不適がある．まず, (1) の方式では特に k の値が大きい場合に，多数の異なる p を開始点として，道路網上の同じノードがそれぞれの p に対して多数回たどられることになる．一方，(2) の方式では $|Q|$ が大きい場合，それぞれを開始点として同じ p までの距離を計算するため，やはり同じノードが多数回展開される場合がある．これらの特質を次章で実験により検証する．

4. 実験結果

本稿で述べた k-NN 及び ANN 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．POI の密度 (d) は，道路セグメントあたりの POI 存在確率で表している．即ち，$d = 0.001$ とは 1000 本の道路セグメント (隣接する交差点間) に 1 個の割合で存在することを示している．隣接リストは，Peano-Hilbert スキャンで順序づけられ，2KB 単位にブロック化した．また読み込みには 50 スロット分の LRU バッファを用いた．

図 2 は，$|Q| = 7$ の場合に，kANN 検索における k の値を変えたとき，各方式の実行時間を示している．また，図 3 は $|Q| = 3$，$k = 5$ のときの d と実行時間の関係を示している．

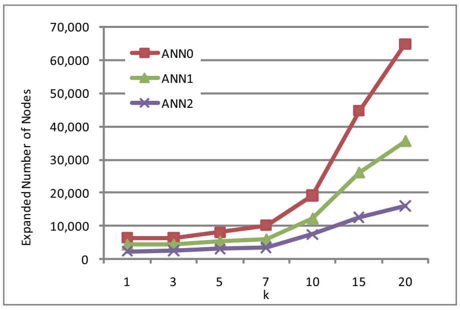

図 2: $|Q| = 7$ の場合の k と展開ノード数の関係

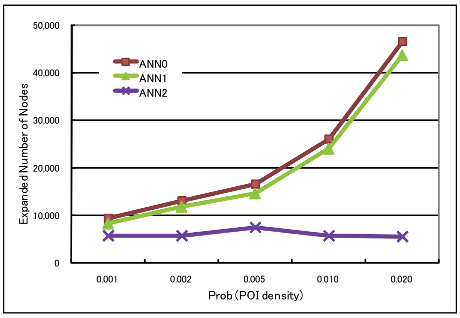

図 3: ノード展開数と $Prob$ の関係 ($|Q| = 3$,k=5)

5. おわりに

本稿では，1 つの探索点から道路網上での移動距離による k-NN を探索する際に，A*アルゴリズムを多数の目的地に拡張したアルゴリズム SSTMA*を提案し，ANN 検索に適用した．また，実際の道路地図，及び疑似乱数により発生させた POI を用いて実験を行い，提案方式の優位性を示した．

参考文献

[1] D. Papadias, Q. Shen, Y. Tao and K. Mouratidis: “Group nearest neighbor queries”, Proceedings of the 20th International Conference on Data Engineering, pp. 301–312 (2004).

[2] M. L. Yiu, N. Mamoulis and D. Papadias: “Aggregate nearest neighbor queries in road networks”, IEEE Transactions on Knowledge and Data Engineing, **17**, 6, pp. 820–833 (2005).

[3] D. Papadias, Y. Tao, K. Mouratidis and C. K. Hui: “Aggregate nearest neighbor queries in spatial databases”, ACM Transactions on Database Systems, **30**, 2, pp. 529–576 (2005).