D-033

多様な移動体連続検索の為の道路網距離での safe-region 高速生成方式
Fast Safe-region Generation Method for Continuous Queries in Road Network Distance

大沢　裕 *　　Htoo Htoo*
Yutaka Ohsawa　　Htoo Htoo

1. はじめに

車や人などの移動体が，移動しながら各地点において同じ種類の検索を繰り返す検索形態は連続検索と呼ばれる．一方，ある検索点から一回行われるデータ点の検索はスナップショット検索と呼ばれる．スナップショット検索を繰り返し実行することにより連続検索を行うことが可能であるが，本稿では連続検索を効率よく実行する方式について述べる．

連続検索の効率化方式は，主としてユークリッド距離を対象にして研究されてきた．しかし，車や人などは道路網上を移動するため，各種検索においてはその道路網上での距離や移動時間が近接性の尺度として用いられる．また，ユークリッド距離で開発された多くの検索方式は，直接道路網距離には適用できない．本稿では，移動体からの道路網距離での近接性に基づく多様な連続検索を高効率に行う方式を提案する．

移動体からの連続検索には，多くの場合にクライアント–サーバモデルが用いられる．ある移動体から連続検索がサーバに対して発行されると，サーバは個々の移動体毎にスレッドを立ち上げ，そのスレッドが移動体の検索要求に応える．移動体が連続検索を行う場合の単純な形態として，(1) 一定時間毎に検索を実行する，または (2) 移動体が一定距離移動する毎に検索を実行する，ことが考えられる．しかし，これらの方式には無駄或いは検索もれを生じるという欠点がある．

そこで，無駄な検索を行わず，かつ検索結果が異なる場合にそれを見逃さない方式として，safe-region 方式が提案された [1]．safe-region とは検索結果が同じになる範囲である．サーバー側でこれを求め，クライアント側に送る．クライアント側では，現在位置が safe-region の内部か外かを確認する．移動体が safe-region の内部の場合には現在位置における検索結果は先に求めたものと変わらない．一方，外に出た場合にはサーバに対して新しい検索結果と，その safe-region を要求する．サーバはクライアントの現在位置を基に新たな検索結果と safe-region を求め，それらをクライアントに送る．

この safe-region の考え方は，前述の (1) や (2) の方式に比して，再計算のタイミングに必然的な意味が存在する．従って，連続モニタリングに有効な方法としてユークリッド距離や道路網距離での連続検索に対して safe-region の考え方に基づく多くの方式が提案されてきた．本稿で対象とする，道路網距離での連続検索に対しても，範囲検索，kNN 検索，RkNN 検索，Skyline 検索などの連続検索方式が提案されている．しかし，これらの方式は特定の検索を対象としたものである．また，各検索方式は処理速度改良の余地が大きい．

本稿では，近接性に基づく検索，例えば距離範囲検索，kNN 検索，RkNN 検索，拡張ボロノイ領域検索，などを対象とした汎用的で高速な連続検索の枠組みを提案する．

2. 方式の概要

本稿では，道路網のような大規模重み付グラフ (V, E, W) を対象とする．ここで，V はノード，E はエッジの集合を表す．また，W は各エッジに付けられた重みの集合であり，$w(\in W) \geq 0$ と仮定する．以下では，W としてリンクの距離を用いる．

まず，本章で検索例に挙げる距離範囲検索とその safe-region を定義する．

定義 1 (距離範囲検索)．距離 D と検索点 q が与えられたとき，q から距離 D の範囲内に存在するデータ点を全て求める検索を距離範囲検索と呼ぶ．

定義 2 (距離範囲検索の safe-region)．検索点が q から移動するとき，q における検索結果と同じ検索結果が得られる道路網上の領域を *safe-region* と呼ぶ．

*埼玉大学

検索対象とするデータ点の集合を S とし，検索点を q とする．本章では，q における距離範囲検索結果と一致する道路網上での範囲を求める検索を考える．この結果が，連続検索における safe-region となる．図 1 は道路網距離における距離範囲検索の一例を示している．ここで，$p_1 \sim p_3$ はデータ点であり，グラフ中の全ての枝の長さを 2 とする．検索範囲を q から距離 6 の範囲とするとき，p_1 と p_2 はその検索範囲に含まれる．

q を含む safe-region は，距離範囲検索結果の集合が変わらない道路網上の範囲である．図 1 に太線で示される範囲が距離 D の範囲に対する safe-region である．

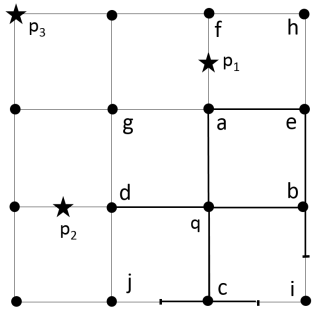

図 1: 範囲検索の safe-region

本稿で述べるアルゴリズムの基本は，以下の通りである．

Step 1 検索点 q から Dijkstra 法と同様に，近接ノードから遠方に向け徐々に探索範囲を拡大する．

Step 2 上のステップで訪れたノードにおいて，検索条件を満たすかのチェックを行う．

safe-region は，グラフ上のエッジ，またはエッジの一部の集合で構成される．従って，検索結果に含まれるべき全てのエッジは 1 度訪れられる必要がある．上記の Step 1 はエッジの端にあるノードを全て訪れるために必要である．Step 2 ではエッジの端点において q からこのノードを超えて更に探索が必要であるかのチェックを行う．もし，必要があれば更に範囲を拡大して探索し，もし必要がなければ，現在訪れているノードを超えての探索を行わない．safe-region の周辺ノード全てにおいて範囲の拡大が行われなくなった時，周辺ノードに隣接するエッジを分割し，safe-region に含まれるべき部分を含めた後，処理を終了する．

この枠組みは，距離範囲検索の他，kNN 検索にも直性適用可能である．本研究では，kNN 検索結果が集合として一致する領域を safe-region に設定する場合 (SkNN) と，kNN 検索結果が順番も含めて一致する場合 (OkNN) について検討した．

3. 高速化

2 で述べたアルゴリズムは 2 つの面から高速化できる．1 つは，検索候補点を得る場合のユークリッド距離での検索であり，他の 1 つは検索結果の道路網距離での検証である．本章では後者の面からの高速化について述べる．

前章で述べた 2 つのステップの内，Step 2 では各ノードにおいて検索条件が満たされているかチェックする必要がある．この検索は IER の枠組みで行うことができるが，道路網距離の検証に A*アルゴリズムを用いた場合，繰り返し似

た経路の最短路が探索されることになり，その処理に要する合計時間は膨大となる．safe-region の高速生成の為には，この部分の効率化が必要となる．以下ではこの処理の高速化について述べる．

道路網上に 2 点 s と e が与えられ，2 点間の道路網距離 $d_N(s,e)$ を A*アルゴリズムで求める場合を考える．A*アルゴリズムでは s，e 間の最短路を求める際に，途中の道路網上のノード n において，$c = d_N(s,n) + d_E(n,e)$ が小さなものから順次探索処理を進める．但し，$d_E(n,e)$ は n，e 間のユークリッド距離を表す．c 値が最小のノードを得る為に，優先順位付キュー（PQ）を用いる．また，PQ で管理されるレコードの書式を，$< c, n, d >$ とする．但し，c は上に示したコスト値，n は現在注目している道路網上のノード，d は $d_N(s,n)$ である．

一度 PQ から取り出され, 処理されたノードは終了集合 CS に入れられる．PQ に初期レコード $< d_E(s,e), s, 0 >$ を入れ，処理を開始する．最短路の探索処理が進み，PQ から c 値が最小のレコード r を取り出したとき，$r.n$ が e に一致すれば s, e 間の道路網距離が求まったため処理を終了する．連続検索において一度検索対象となったデータ点毎に PQ と CS の内容を保存する．もし，$r.n$ が既に CS 中に入れられていれば, そのノードに対して既に処理が終わっているため，以下の処理を省略する. それ以外のとき，$r.n$ に隣接する全てのノードを隣接リストから得て, その各隣接ノードに対して新たなレコードを作製し，PQ に追加する．

safe-region の高速化を次の方式で実現する．ユークリッド距離での検索で候補となった点 (C_i) 毎に独立した PQ と CS を割り当てる．この PQ と CS の内容は，safe-region 生成の全過程で保持する．2 章で述べた Step 1 で q の近傍のノード n でも同じ検索結果が得られるかの検査が行われるとき，まず n が C_i から探索した際の CS に含まれているか調べる．n が CS 中に存在する場合，CS 中のレコードの d 値が C_i から n への最短距離である．一方，含まれていない時，PQ の全レコードの c 値を，n を目的地とするものに置き換える．その後，A* アルゴリズムで n への最短路探索を行う．即ち，n の直前に C_i から n の近傍への最短路を求めた際に得られた PQ の内容を再利用するため，A*アルゴリズムでの最短路探索は高速になる．以上で述べた処理は，SSMTA*アルゴリズム [2] と同じ原理に基づいており，正しさは文献 [2] で証明されている．

図 2 は，上記のアルゴリズムにより s から q への最短路探索を行い，更に q の近傍ノード $n1 \sim n4$ への最短路距離を求めた後の PQ の内容 (wave-front) を□で示している．

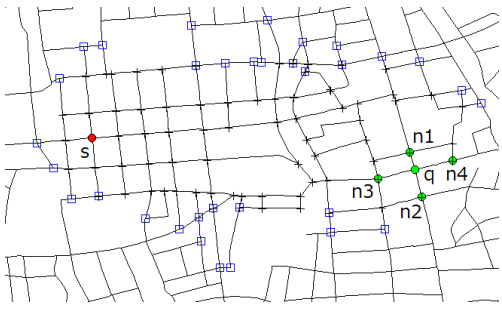

図 2: A*アルゴリズムの効率化

4. 実験結果

提案方式の性能を評価するため，実際の道路地図を用いた評価実験を行った．使用した道路地図はノード数 16,284，リンク数 24,914 である．移動体の移動軌跡は，実際の走行記録の他，生成した移動軌跡も用いた．提案方式は Java により実装し，計算機としては Intel Core i7 4770CPU (3.4GHz) を用いた．

まず，safe-region 作成に要する処理時間の比較実験を行った．比較対象は，検索点から領域拡大を行う際に，道路網上の各ノードが safe-region に入っているか否かの決定を IER と A*アルゴリズムで行う方式である．これを基本方式と呼ぶ．実験では，SkNN 検索，OkNN 検索，距離範囲検索の各々に対して，データ点の密度が異なる場合の処理時間を比較した．

図 3 は，k が 5 の場合に，データオブジェクト集合の存在密度を変化させたときの SkNN 領域作成の処理時間を示している．データオブジェクトの密度が上がるにつれ，SkNN 領域は小さくなることから，SkNN 領域作成に要する時間が短くなっている．提案方式は元方式に比べて 10 倍以上高速であることがわかる．

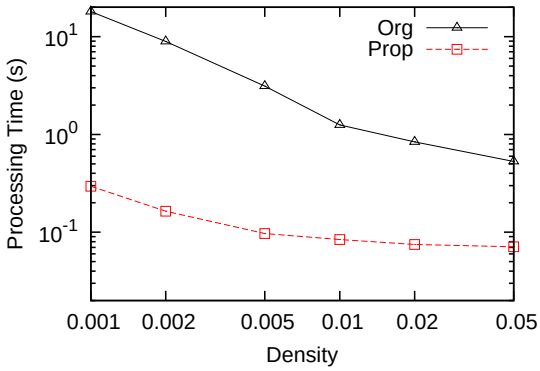

図 3: S5NN 領域の作成時間

safe-region どの程度の距離を走行するたびに新しい結果を得る必要があるか．SkNN，OkNN，距離範囲検索について比較した．

図 4 は，データ点の存在密度を変化させた場合に，SkNN 検索と OkNN 検索において，safe-region を移動可能な平均距離を示している．

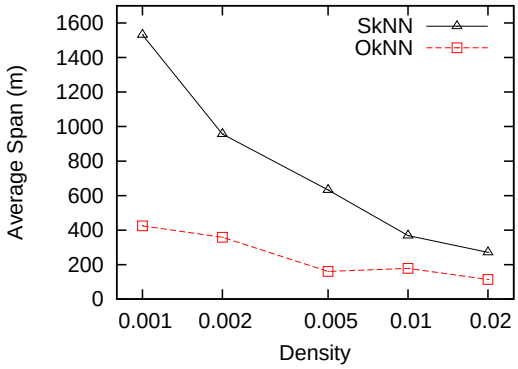

図 4: 連続検索における更新距離 (k=5)

参考文献

[1] M. A. Cheema, X. Lin, W. Zhang and Y. Mhang: “Influence zone: Efficiently processing reverse k nearest neighbors queries”, Proceeding ICDE, pp. 577–588 (2011).

[2] H. Htoo, Y. Ohsawa, N. Sonehara and M. Sakauchi: “Incremental single-source multi target A* algorithm for LBS based on road network distance”, IEICE Transactions on Information and Systems, **E96-D**, 5, pp. 1043–1052 (2013).