1K-07

連続最適巡回路検索方式

菊地 弘樹　　Htoo Htoo　　大沢 裕
埼玉大学大学院理工学研究科

1 はじめに

位置に関連した情報サービスやカーナビゲーションシステムでの利用を目的として，様々な経路探索が研究されている．この分野は，近年，スマートフォンやタブレットの普及や進化によって，地図やナビゲーションシステムなどを手軽に利用することができるようになり，私達の生活にとってより身近になった．

経路探索の一種として，旅行計画問題がある．これは，現在地から目的地までに，銀行や映画館，レストランなど，いくつかの種類の points of interest(POI) を 1 つずつ経由する最短経路を求める検索のことである．訪れる順序が一意に指定されているものは，最適巡回路検索 (Optimal Sequenced Route query)(OSR) (Sharifzadeh ら [1]) と呼ばれている．

また，実生活では，道路工事や事故，不注意などの理由で，求めた旅行経路から逸れる可能性がある．OSR 検索は多大な処理時間を要するため，求めた旅行経路から逸れる度に再検索を行うのは効率が悪い．OSR 検索以外でも，経路探索において一般的に，ユーザが求めた経路から逸れる場合の検索は連続検索と呼ばれており，従来から検索効率向上のために，検索時に safe-region と呼ばれる領域を生成する方式が採られている．

safe-region とは，移動体がその領域から出ない限り結果が変わらないような領域のことであり，safe-region 生成方式は，単純な検索では方式提案がなされているが，連続 OSR 検索のような複雑な検索では研究されていない．よって，本稿では，連続 OSR 検索のための safe-region 生成方式を提案する．

2 OSR 検索

まず，OSR 検索の定義を述べる．

定義 1 (OSR 検索) M 種類のデータ点カテゴリ C_i $(1 \leq i \leq M)$ と現在地 q，目的地 d が与えられたとき，q から各データ点 p_i を経由して d に至る，最もコストが低い旅行経路を求める．

また，本稿では OSR 検索アルゴリズムに，Htoo らによって提案された A^* 法に基づくアルゴリズム [2] を用いる．そのアルゴリズムは，現在地 q から訪問順序通りに A^* 法を用いて探索を広げていく．最初に経由するカテゴリ C_1 に属すデータ点を見つけたとき，次に経由するカテゴリ C_2 のデータ点の探索と，同じカテゴリ C_1 の別のデータ点の探索を並行して行い，それを繰り返し，目的地 d に辿り着いたとき探索を終了する．

3 safe-region 生成

図 1 は，連続 OSR 検索における safe-region の概念を示しており，ここでは $q \rightarrow \bigcirc \rightarrow \triangle \rightarrow d$ という順に経由する．safe-region 内の q' へ移動しても経由点は変化しないが，safe-region 外の q'' へ移動すると経由点が変化するため，q が領域外へ移動したとき新しい safe-region を生成する．また，最初に経由するカテゴリ (○) に属す p_1 に到達したら，p_1 から p_2 を経由して d へ至る OSR のための safe-region を生成する．

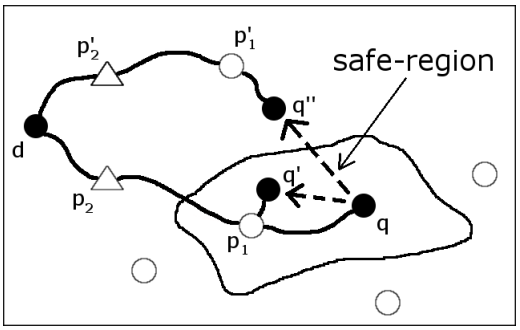

図 1: 連続 OSR 検索における safe-region

3.1 Basic Algorithm

まず，safe-region を生成する単純なアルゴリズム Basic Algorithm(BA) を提案する．現在地 q から検索した旅行経路の最初に経由するデータ点 p_1 から Dijkstra 法と同様に探索を広げていく．現在の探索ノード n から OSR 検索を行い，最初に経由するデータ点が p_1 である場合，n を safe-region に含め，C_1 に属す p_1 以外のデータ点である場合，safe-region に含めない．

この方法では，safe-region 内の全てのノードに対して OSR 検索を行う必要があり，膨大な処理時間を要する．そのため，3.2 節，3.3 節で改善した 2 つのアル

Copyright ©2017 Information Processing Society of Japan.
All Rights Reserved.

ゴリズムを提案する．

3.2 Preceeding Rival Addition Algorithm

まず，safe-region の形状は，p_1 付近にある最初に経由するカテゴリ C_1 に属す POI によって影響を受ける．ここで，safe-region に影響を及ぼす POI をライバルオブジェクト (以後，RO) と呼ぶ．

Preceeding Rival Addition Algorithm(PRA) では，q から Dijkstra 法と同様に探索を広げていき，現在の探索ノード n において，以下の条件を満たす p'_1 の集合を RO とする．また，$L_{2..M}(p_1)$ は，p_1 から d までに 2 番目〜M 番目のカテゴリを経由する旅行経路の道路網距離での長さ，$d_N(p_1, n)$ は，p_1 と n の間の道路網距離，$L^E_M(p'_1)$ は，p'_1 から d までに全てのカテゴリを経由する旅行経路のユークリッド距離での長さを表す．

$$L_{2..M}(p_1) + 2d_N(p_1, n) \geq L^E_M(p'_1)$$

n から d までに p'_1 を経由する旅行経路の中で最短のものと p_1 を経由する旅行経路を比べて，p_1 を経由する旅行経路の方が短くなるとき，n を safe-region に含める．このアルゴリズムは，safe-region の形状を決定するのに必要十分な RO 集合に比べて，多大な RO を探索してしまう．この RO の探索と，n から各 RO への道路網距離の計算に長い処理時間を要するという欠点がある．

3.3 Tardy Rival Addition Algorithm

PRA の問題を解決するために，Tardy Rival Addition Algorithm(TRA) というアルゴリズムを提案する．PRA とは RO の決定方法のみが異なる．現在の探索ノード n から最初に経由するカテゴリ C_1 の p_1 以外で最も近い POI を RO とする．PRA に比べ，RO の数が少ないため，処理時間が短いが，正確な safe-region を生成するために必要十分な数の RO が見つからない場合があり，生成する safe-region の大きさが数%広がってしまう可能性がある．

4 実験結果

提案した safe-region 生成アルゴリズムを比較するため，ノード数 16,284，リンク数 24,914 の実際の道路地図を用いて実験を行った．また，アルゴリズムは Java で実装した．

図 2 は，カテゴリ数 $M = 4$ の場合にデータ点密度を変化させて BA，PRA，TRA の処理時間を比較した結果を示している．BA は，密度が低いほど safe-region が大きくなるため，OSR 検索の回数が多くなり，処理時間が長くなる．PRA は，密度が高いほど、RO の数が多くなるため，処理時間が長くなる．TRA は，安定しており，密度に依らず処理時間が短い．

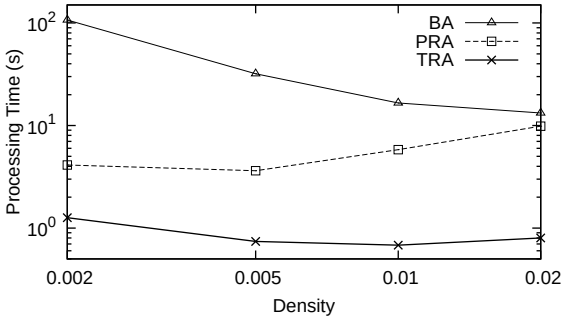

図 2: $M = 4$

図 3 は，$M = 5$ の場合の結果を示している．カテゴリ数が大きくなると，全てのアルゴリズムの処理時間が長くなるが，PRA と TRA は BA より処理時間が短い．

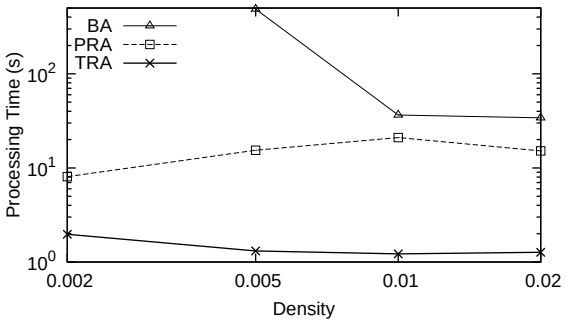

図 3: $M = 5$

5 結論

本稿では，連続最適巡回路検索方式のための safe-region 生成アルゴリズムを 3 種類提案した．最初に BA を提案したが，このアルゴリズムは処理時間が長い．次に提案した PRA は，BA より処理時間が短いが，特にデータ点の密度が高いときに多大な RO を探索することから，処理時間が長くなる．最後に提案した TRA は，PRA より RO の数が少ないため，処理時間が短いが，生成する safe-region の大きさが数%広がってしまう可能性がある．

参考文献

[1] M. Sharifzadeh, M. Kolahdouzan, C. Shahabi, “The Optimal Sequenced Route Query”, The VLDB Journal, Vol.17, pp.265-287, 2008.

[2] H. Htoo, Y. Ohsawa, N. Sonehara, M. Sakauchi, “Optimal Sequenced Route Query Algorithm Using Visited POI Graph”, WAIM, pp.198-209, 2012.

Copyright ©2017 Information Processing Society of Japan.
All Rights Reserved.