道路網上におけるC-ＯＳＲ探索法

西潟　耕治・Htoo Htoo・大沢　裕・曽根原　登

Continuous Optimal Sequenced Route Query on Road Network

Koji Nishikata • Htoo Htoo • Yutaka Ohsawa • Noboru Sonehara

Abstract: Several types of trip planning query methods have been proposed. Most of them search the optimal trip route for the fixed query point. When the query is invoked from a moving object (eg. a car), however, the position always changing and sometimes the moving object may ignore the instructions. In the case, re-calculation becomes necessary. This paper proposes a method named C-OSR query for speeding up the re-calculation.

KeyWords: 旅行計画，道路ネットワーク，ＯＳＲ探索，ＰＯＩ探索

1. はじめに

近年，LBS(location based services) の普及に伴い，各種旅行計画の効率化アルゴリズムが活発に研究されている．レストランや，コンビニ，ホテル，ガソリンスタンドなど，旅行時に訪れる対象を POI(point of interest) と呼ぶ．旅行計画では，目的地と，その旅行の途中で訪れる POI の種類が指定されたとき，現在位置から目的地に到達する途中で，指定された POI 種の中から 1 つずつ訪れる最適な経路を求める演算である．この最適化の尺度としては時間や費用，距離など多数考えられるが，以下では旅行の総距離を最短化することを考える．

本稿で扱う旅行計画は Sharifzaded ら [3] により提案された OSR(optimal sequenced route) である．OSR は途中に訪れる POI の種類とその訪問順序が指定され，現在位置からその順序で POI を１つずつ訪れ，目的地に達する経路を求める．Shrarifzaded らは道路網上での OSR 探索アルゴリズムとして PNE(progressive neighbor exploration) 法を提案している．これは現在位置から目的地に向けて POI を探索していく方式である．しかし，PNE は多大な演算量を必要とする．そこで，藤井ら [4] は，VNG(visited node graph) という構造と双方向探索を用いることにより，その演算量を大幅に減少させる方式を提案した．しかしこれらの方式では，最初に求められた最適経路を無視して車が進んだ後，別の位置を始点としてでもう一度同じ条件での検索が求められたとき，最初から再度探索を行いなおさなければならないという欠点を有していた．

本稿では，そのような状況で直前に計算された結果を再利用し，高速な検索を可能とする OSR 探索アルゴリズム（C-OSR: continuous OSR）探索アルゴリズムを提案する．基本的な処理は，最初の探索で目的地から現在位置に向けて逆方向に OSR 探索を行い，その結果の VNG やヒープの内容を保存しておき，現在位置が移動した状況で再計算が求められたとき，それらのデータを利用することにより高速な探索を行うものである．

2. 関連研究

始点と終点が与えられ，その 2 点を結ぶ最短路を求めるアルゴリズムの代表例に Dijkstra 法や A*アルゴリズムがある．また，この演算を高速に実行するための前処理 (マテリアライズ化）の方式も各種提案されている．

一方，k-NN 検索，CkNN 検索，ANN 検索，空間

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

的スカイライン検索など，多様な状況と目的に合わせて道路網上での距離に基づく近接 POI を探索するアルゴリズムも各種提案されている．

Li ら [2] は TPQ(trip planning query) という旅行計画法を提案した．これは，途中に訪れる POI の種類のみが指定され，それぞれの POI 種を 1 つずつ訪れて目的地に達する経路を求めるものであるが，途中経由する POI 種の数について NP 困難な問題となる．Sharifzadeh らは，この問題を単純化し，先に述べた OSR を提案した．しかし，そこで提案されているアルゴリズムの多くはユークリッド距離による巡回路の最適化であり，それを直接道路網上での距離に適用することは困難である．道路網上での距離に適用可能な OSR 探索アルゴリズムは INE のみである．

INE は，現在値から目的地に向けて，指定された順番で 1 つずつ POI を探索する方式であるが，道路網上での最短路を求めるために Dijkstra 法を用いており，また探索の途中で見つかった全ての POI から次の POI に向けての探索が開始されるため，同じノードが何度もノード展開されることになり，訪問する POI 種の増加に伴い実行速度は大幅に遅くなる．

藤井ら [4] はこの問題に対応するため，VNG と呼ぶ一度発見された POI 相互の関係を記述するグラフを用い，始点と目的地からの双方向探索を行い，最短路の探索は A*アルゴリズムを用いることにより大幅な速度向上を図った OSR 探索法を提案している．

一方，Chen ら [1] は，TPQ と OSR を一般化した旅行計画を提案している．これは，MRPSRQ(multi-rule partial sequenced route query) と呼ばれ，途中に訪れる POI 種の順序をルールとして指定するものである．例えば，銀行とレストランと映画館を訪れる際に，まず銀行を訪れる必要があり，レストランと映画館の訪問順序はどちらが先でもよい，という関係のみを指定する旅行計画である．

3. 提案方式

3.1. 準備

OSR 探索の概要を図 1 に示す．ここで，C は移動体の現在位置，D は目的地である．また，C から星印，四角，△の POI をこの順序で一つずつたどり，D に至る OSR を求めるものとする．文献 [4] で提案されている OSR 探索法は，図中の破線で示すグラフ，VNG(visited node graph) を作成しながら探索することにより，PNE の欠点である同じ部分ルートの重複探索を無くすことにより高速化を達成するものである．経路探索には A*アルゴリズムを用いる．

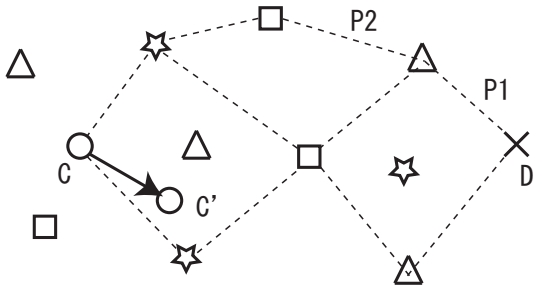

図 1: OSR 探索の例

目的地 D は固定されており，現在位置 C は常に変化する．そこで，OSR を D から C に向けて逆順にたどる．具体的には，A*アルゴリズムを用いて D から C への最短路の探索から開始し，徐々に探索範囲を広げながら最初の POI 種 (図では△) を探索していく．△の POI が見つかったとき，D からその POI へのパス (図では P1) を VNG に登録し，次の 2 つの探索を行う．1 つは，△から四角の POI を目指す探索であり，他の 1 つは現在見つかった△を無視して，更に別の△を目指す探索である．これらの処理を繰り返し，D から C に至る途中で指定された順序で POI を 1 つずつ訪れる OSR を最短のものから k 個 (指定された任意個) 求める．

本稿で扱う C-OSR 探索は，先の検索時の現在位置 C から，その検索結果に従わずに新しい位置 C' に移動したとき，その C' からの OSR 探索を先の検索時に得られたデータを用いて高速に再計算するものである．ここで，最初の検索時に得られるプライオリティーキュー (PQ) の内容と，VNG を以降の検索で利用する．これらを利用して，再計算を抑え効率よく新しい OSR を求める．

3.2. アルゴリズム

PSR(pertial sequenced route) とは，部分的なルートである．例えば，D から最初の△の POI が見つかったとき，その間のルート (P1) が PSR になる．更に△から更に探索を進め四角の POI が見つかったとき，

D から△ (P1)，更に四角 (P2) を結ぶ経路が PSR となる．

アルゴリズム 1 で用いる VNG は，先に述べた訪問パスグラフ (visited node graph) であり，各 POI 間の最短経路を格納している．$R.length$ は結果集合 R における最長の経路の経路長であり，R に指定本数の k 本格納されていない場合は $R.length = \infty$ である．$n.cost$ は，探索開始点 (D) から n までの経路長と，n から探索終了点 (C) までのユークリッド距離の和である ($n.cost = n.length + n.heuristic$)．

アルゴリズムの詳細について述べる．まず，プライオリティキュー $\mathcal{PQ}$，結果集合 R，訪問経路グラフ VNG を初期化する．$\mathcal{PQ}$ は初期探索の場合は目的地 D(但し，C-OSR 探索の開始点) を挿入する．二回目以降の再探索の場合は，前回で使用された $\mathcal{PQ}$ を引き継ぐ（これは前回の探索において一度は $\mathcal{PQ}$ から取り出された n を格納したもの保存しておき，それを次回の探索において $\mathcal{PQ}$ に代入する）．$\mathcal{PQ}$ の優先度に，$n.cost$ を用い，その値が最小のものから取り出される．

$\mathcal{PQ}$ からノードを取り出す．もしこの値が，結果集合 R の k 番目のコスト ($R.length$) よりも大きければ，R を返しアルゴリズムを終了する．取り出したノードに対し，以下の処理を行う．

(1) 取り出したノードが POI であれば，直前に経由した POI から到達した POI までの部分経路を VNG に登録する．

  (1.1) POI が現在地であれば，目的地から現在地へ至る経路が確定したことになる．VNG を辿り，条件（$PSR.length < R.length$）を満たすすべての経路を結果集合 R に格納する．

  (1.2) 一度訪れたことのある POI であれば，目的地から取り出したノードを経由して現在地へ至る経路が存在するか確かめる．存在していれば，VNG を辿り，条件（$PSR.length < R.length$）を満たすすべての経路を結果集合 R に格納する．

  (1.3) 上記二つに当てはまらない場合は，到達した POI から，次に訪れるカテゴリ（本来の順序的には一つ前の POI カテゴリ）の最も近い POI への探索と，一つ前に経由した POI から，到達した POI と同じカテゴリーに属する別の POI への探索を開始する．

(2) 取り出したノードが POI でない場合は，そのまま展開を続ける．

$\mathcal{PQ}$ に格納されたものがなくなったら，R を返しアルゴリズムを終了する．

アルゴリズム中の関数 NE($node, POI_i$) は現在のノード $node$ から i 番目もカテゴリの POI を目指してノード展開を行う関数である．

Algorithm 1 Continuous OSR

```
1: R ← ∅
2: VNG ← first search ? ∅ : VNG_{Last used}
3: PQ ← first search ? {E} : PQ_{Last used}
4: loop
5:     n ← deleteMinCost(PQ)
6:     if n.cost > R.length then return R
7:     end if
8:     if n is POI then
9:         経路を VNG に追加
10:        if n は探索終了点 or n は一度訪れたことのある POI then
11:            PSR ← 出発地から n を通って現在地に至るすべての経路
12:            for all PSR do
13:                if PSR.length < R.length then
14:                    R ← R ∪ PSR
15:                end if
16:            end for
17:        else
18:            NE(n, n.POI_{i−1})   ▷ 一つ前のカテゴリの POI
19:            NE(n.prevPOI, n.POI_i)  ▷ 同じカテゴリの別の POI
20:        end if
21:    else
22:        NE(n, n.POI_i)        ▷ 目標の POI は変わらず
23:    end if
24: end loop
```

4. 実験

本実験では，埼玉県さいたま市の地図データ（数値地図 25000）を使用した．POI は 2 次元擬似乱数を用いて作成した．POI の密度は，1 道路セグメント当たりに存在する POI 数で表現している．例えば POI 密度 0.001 とは，1000 道路セグメント当たり 1 個の POI が存在する密度である．

本実験で比較対象としたのは，PNE[3] と高速 OSR[4] である．高速 OSR では，境界カテゴリによ

る展開抑制を行ったアルゴリズムによる経路探索を用いた．また，C-OSR は，高速 OSR を基にして構築した．これにより，高速 OSR による計算時間の高速性と，C-OSR による効率の良さの両方の実現を図る．

実験は，まず POI 密度が同一な 3 カテゴリ分の POI 配置を 3 パターン，始終点の組を 4 組用意．これらのすべてのパターンに対し，始点 S が探索により求まった経路以外の移動（ここでは現在位置と目的地とを結ぶ，ユークリッド上の最短距離に沿った，最短距離の 10%の移動）に伴う試行（例えば，図 1 における D–C 間の探索は試行 1 回目，D–C' 間の探索は試行 2 回目）を合計 5 回行った．

図 2 は，POI の密度を 0.002 とした場合の探索 1 回あたりの平均計算時間を示している．横軸は試行回数，縦軸は対数軸で平均計算時間としてある．図 3 は，各手法の 1 回目の試行の計算時間を 100 と置いた時の，2 回目以降の試行の計算時間の推移（以後，増減率と呼ぶ）の平均をとったものである．横軸は試行回数，縦軸は増減率としてある．なお，1 回の試行で最短路から短い順に 3 本の経路を求めている (kC-OSR における $k = 3$ である)．

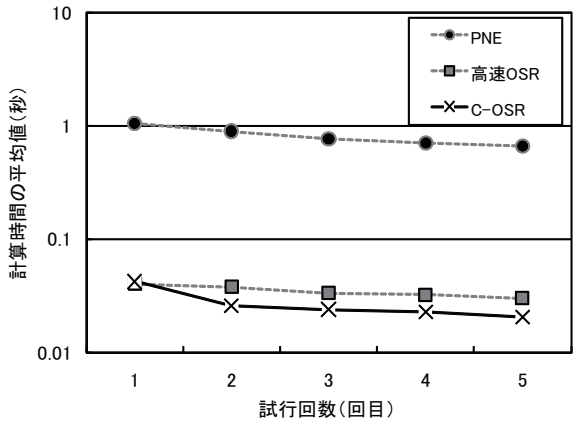

図 2: 実験結果 1

図 2 から，提案方式が従来法に比して再計算時に高速な探索が行えていることが分かる．また，図 3 から，既存手法では再計算時の処理時間の変化はないが，提案方式では 2 回目以降の増減率は 50–60%の処理時間に低減している．

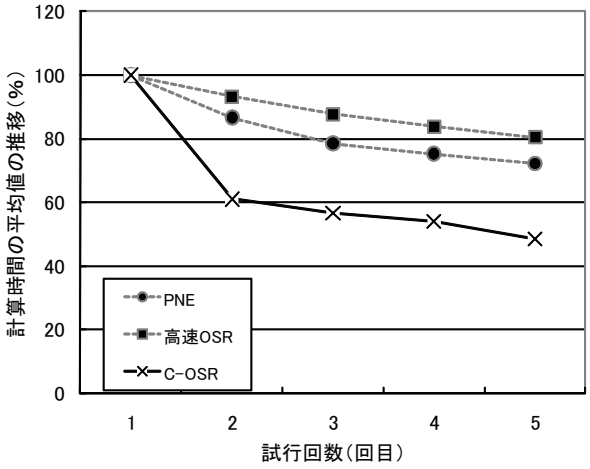

図 3: 実験結果 2

5. まとめ

本研究では複数の POI カテゴリを経由する旅行計画問題である OSR を求める際に，探索点が移動する場合の再計算を高速化するアルゴリズム，Continuous OSR 探索法を提案した．現在位置の移動量に対する計算コストの評価は今後の課題である．

参考文献

[1] Haiquan Chen, Wei Shin Ku, Min Te Sun, and Roger Zimmermann. The multi-rule partial sequenced route query. In *ACM GIS '08*, pp. 65–74, 2008.

[2] Feifei Li, Dihan Cheng, Marios Hadjieleftheriou, George Kollios, and Sang-Hua Teng. On trip planning queries in spatial databases. In *Proc. SSTD 2005*, pp. 273–290, 2005.

[3] M.Sharifzadeh, M.R.Kalahdouzan, and C.Shahabi. The optimal sequenced route query. Technical report, Computer Science Department, University of Southern Calfornia, 2005.

[4] 藤井健児, Htoo Htoo, 大沢裕. 境界カテゴリーを設定した双方向探索による高速 OSR 探索法. 電子情報通信学会論文誌　D, Vol. J93-D, No. 12, pp. 2587–2596, 2010.