ユークリッド距離での制約をベースとする単純な旅行計画の為の高速アルゴリズム

トウ　トウ† 　大沢　裕†

† 埼玉大学大学院理工学研究科　〒 338-8570 さいたま市桜区下大久保 255
E-mail: †htoohtoo2@gmail.com, ††ohsawa@mail.saitama-u.ac.jp

あらまし　本稿では道路網上で，旅行開始点 (s) と目的地 (d)，及び旅行の途中で訪れる施設 (POI) の種類が与えられたとき，s から出発して，与えられた種類の POI を 1 つ訪れ，d に至る最小コストの経路を求める STP (simple trip planning) 検索の高速アルゴリズムを提案する．方式の基本は，まずユークリッド距離に基づく検索により候補集合を絞り込み，その結果を道路網上の距離で検証する枠組み，即ち IER に基づくものである．本稿では，まず R 木上で STP を高速に求めるアルゴリズムを提案する．次に，1 つの探索開始点から複数の POI までの経路探索を並列的に実行する方式を提案する．

キーワード　LBS, 道路網, 旅行計画, 経路探索

Fast Algorithms for Simple Trip Planning Queries Based on Euclidean Restriction

Htoo HTOO† and Yutaka OHSAWA†

† Grad. School of Science and Technology Saitama University
Shimo-okubo 255, Sakura-ku, Saitama 338–8570 Japan
E-mail: †htoohtoo2@gmail.com, ††ohsawa@mail.saitama-u.ac.jp

Abstract　This paper proposes a fast algorithm for simple trip planning (STP) queries which find the shortest detour route via a POI belonging a specified category from a trip start point (s) to a final destination (d). The proposed method bases on the incremental Euclidean restriction (IER) framework which generates candidate route by Euclidean distance search, and then verifies it on road network distance. This paper proposes a fast search algorithm based on Euclidean distance referring R-tree,then an efficient concurrent shortest paths search algorithm for verification on road network distance.

Key words　LBS, Road Network, Trip Planning Query, Path Search

1. はじめに

現在，道路網上を移動する際の各種旅行計画法が提案されている．ここで言う旅行計画とは，現在位置と最終目的地，及び途中に訪れる施設の種類が与えられたとき，現在位置から旅行を開始して，指定された種類の施設の中から 1 つずつをたどり，最終目的地に達する際の最適経路を求める問題である．経路の最適性にはいくつかの尺度が用いられるが，一般的には旅行経路の総距離や旅行時間を最小化する経路が求められる．また，ユーザが好みに併せて選択可能なように，経路長が最短のものから順に複数の経路 (以下では k 個の経路) の提示が求められる．

本稿では，現在地から目的地に達する途中で 1 つの種類の施設を訪れる際の最短経路を求める，効率化アルゴリズムを提案する．本方式の基本は，ユークリッド距離により検索で訪れる施設候補を選び，それを道路網距離で検証する枠組みにより探索する．ユークリッド距離での探索と道路網上での距離の探索の大きな違いは，前者の計算は高速であるのに対して，後者の演算は距離に応じて実行時間が増大する点にある．

本稿の主張点を以下に示す．

- POI の分布密度に寄らず高効率な旅行計画アルゴリズムを提案する．本稿で扱っている対象は最も単純な旅行計画であるが，ここで提案する基本的な考え方はより複雑な旅行計画に直接適用可能である．
- ユークリッド距離に基づく，R 木上での最適優先探索による旅行計画候補生成アルゴリズムを提案する．このアルゴリ

ズムはインクリメンタルな検索が可能である．

2. 関連研究

本研究には，大きく分けて 2 種類の研究が関連する．1 つは旅行計画に関する研究であり，他の 1 つは，ユークリッド距離に基づく検索で候補を生成し，それを道路網上での距離で検証する枠組みの研究である．

道路網上での距離に基ずく各種旅行計画の研究は，単純な最短経路探索を除けば，Shekhar ら [1] による In-Route Nearest Neighbor(IRNN) の提案が最初と思われる．これは，現在地から目的地に向かって車が移動する際に，その車が日頃良く通るルートからの逸脱が少ないという条件の下で，経路長が短い経路を求める問題である．訪れる施設の種類が 1 種類であることと，良く通るルートが予め与えられていることにより，旅行計画の中では最も単純な問題である．

Li ら [2] は，TPQ(trip planning query) という旅行計画問題を提案した．これは $m(m \geqq 1)$ 種類の施設が与えられ，現在地から最終目的地に達する途中で，与えられた種類の施設を 1 つずつ訪れる際の最短巡回経路を求める問題である．例えば，施設の種類として，銀行，レストラン，映画館が与えられたとき，訪れる順番にはこだわらずに，必ずそれらの種類に属す施設を 1 つずつ訪れることが求められる．

TPQ では与えられた施設カテゴリーに対して，訪れる順番に制約が設定されていないが，実用上は訪問順序に制約が与えられる場合も多い．例えば，銀行でお金を引き出し，レストランで食事をし，最後に映画を見たいという場合である．このように訪問順序が定められた旅行計画は，SR(sequenced route) と呼ばれる．Sharifzadeh ら [3] は，ユークリッド距離による最適 SR，即ち OSR (optimal sequenced route) を求めるアルゴリズムを提案した．その後，道路網上での OSR を求めるアルゴリズム PNE(progressive neighbor exploration) [4] が提案されている．また，藤井ら [5] は，道路網上での距離に基づく OSR の高速化アルゴリズムを提案している．

施設の訪問順序に拘束が無い TPQ と，順序が指定される OSR の中間的な状況に対応するため，Chen ら [6] は MRPSRQ(multi-rule partial sequenced route query) という旅行計画法を提案している．これは，銀行は最初に訪れる必要があるが，レストランと映画館の順番はどちらでもよいという状況に応えるものである．訪問順序の拘束条件を AOV(activity on vertex) ネットワークという半順序関係を表現するグラフで表わし，それを満たす順序で最適経路を探索するものである．

一方，道路網上での距離に基づく各種検索の枠組みとして，ユークリッド距離で対象の候補を生成し，それを道路網上での距離で検証するというアプローチがある．Papadias ら [7] は IER(incremental Euclidean restriction) を提案し，それを範囲検索，k-NN 検索，空間ジョイン検索などに応用している．また，Yiu ら [8] は，この IER の枠組みを ANN 検索に適用している．更に，Deng ら [9] は，道路網上での距離を A*アルゴリズムを用いて検証するアプローチとして LBC(lower bound constraint) を提案している．

これらの仮説検証をベースとする方式は，PNE などで採られているノード展開のみによる方法と比較して次の性質を持つものと予測できる．まず，単純なノード展開法は，POI が密度高く存在する場合に高速な処理が可能である．仮説検証の方式は，逆に POI 密度が低い場合に効率が良い．一方，本稿の目的は，POI 密度に寄らず，他方式に比べて効率良い仮説検証をベースとした方式を提案することである．

3. STPQ

3.1 STPQ の定義

本章では，TPQ や OSR の最も単純な形態を取り上げる．即ち，車両の現在位置と最終目的地が与えられ，目的地に到達する前に 1 つの種類の施設を訪れる経路を最短のものから任意個 (k 個) 求める検索である．この検索を本稿では STPQ(simple trip planning query) と呼び，その検索の結果得られる最短寄り道経路を TP(trip path) と呼ぶ．この検索を高速に実行するアルゴリズムや計算の枠組みが得られれば，それを応用することにより TPQ や OSR も高速化可能である．

3.2 経路探索の基本

本稿で扱う方式は，道路網の隣接リストが与えられ，その隣接リストを参照しつつ道路網上での探索範囲を拡大していくものである．隣接リストには道路網上のノード（交差点や道路の終点）に対して，それに直接隣接するリンク（道路セグメント）の集合が記述されている．隣接リストは空間的に近接するノード同士が固定長サイズのブロックに分割されている．探索の際には，これを LRU バッファーに読み込み，参照する．

探索全体は優先順位付きキュー (PQ) で管理される．優先度の付け方は探索アルゴリズム毎に異なるが，Dijkstra 法では探索開始点から現在の注目ノード (n) までの道路網上での距離が，また A*アルゴリズムでは始点から n までの道路網上での距離と n から終点までのユークリッド距離の和が用いられる．PQ からは上記のコストが最小の要素が取り出され，探索アルゴリズム毎に必要な処理が行われる．PQ から取り出したノード (n) に対して，隣接リストを参照することにより，それに直接隣接するノード (及びリンク）が得られる．これらのノードに対してコスト（処理アルゴリズム毎に異なる) が計算され，その結果が PQ に追加される．この，隣接リストを参照し，n に隣接するノードに対してコストを計算し，それらを PQ に入れる処理をノード展開と呼ぶ．また，一度展開されたノードは，再度展開の対象とならないように終了集合 (CS) に入れられる．

以上の処理が，探索目的毎に異なる終了条件を満たすか，または PQ のサイズが 0 になるまで繰り返される．これが，後に述べる全ての探索アルゴリズムで用いられている基本処理である．

3.3 STPQ の従来方式

STPQ の為には，Li ら [2] が TPQ の目的で提案している MDQ(minimum distance query)，Sharifzaded ら [4] が OSR の為に提案している PNE(progressive neighbor exploration) を直接適用可能である．これら 2 つのアルゴリズムは STPQ, 即ち途中に訪れる POI 種が 1 種類の場合には同じである．

MDQ と PNE の概要を以下に示す．現在位置を s, 最終目的地を d，途中に訪れる POI 集合を P とするとき，s から Dijkstra アルゴリズムを用いて P に属す要素を最近接のものから順に探す．これを $p_i \in P(1 \leqq i \leqq |P|)$ と表わすことにする．そして，p_i が見つかる都度，その p_i から Dijkstra アルゴリズムを用いて d を目指す検索を開始する．p_i から d を目指す検索が d に達したとき，s から，その p_i を経由して d に至る経路が得られる．即ち，TP(trip path) は s と p_i を結ぶ経路と p_i と d を結ぶ経路を連結したものである．k-STP 検索は経路長が最小の POI を経由するものから順に k 番目の経路長を与える POI を経由するものまで求めればよい．

このアルゴリズムは単純であるが，次の問題点がある．p_i が見つかる都度 Dijkstra 法による新たな探索が始り，その探索では s からの探索とは別のクローズドセット (CS) を用いる必要があるため，同じ道路網上のノードが複数回展開されることである．特に POI 密度が高い場合に，同じノードが多数回展開され，処理時間が長くなる．また，POI 密度が低い場合にも広い領域の探索が必要となり，展開ノード数が増加する．

大沢ら [10] は，STPQ の為のアルゴリズムを数種類提案しており，その中で AIA が最も優れた性能が得られることを示している．AIA では A*アルゴリズムにより s から d を目指す探索と，d から s を目指す双方向の探索を 1 つの PQ を用いて同時に実行する．s からの探索が d に到達しても，または d からの探索が s に到達してもそこで探索を終了せず，探索範囲を拡大していく．この探索の途中で，何れかの側からの探索がある POI(p_i) に達したとき，それを記録しておく．そして，p_i に両方向からの探索が達したとき，s から p_i を経由し d に達する 1 つの TP が確定する．k-STP 探索では以上の処理を k 個のパスが求まるまで繰り返す．

この AIA は MDQ に比して大幅な高速化を達成できる．また道路網上の各ノードは 1 方向からの探索により高々1 回，双方向の探索で高々2 回の展開に抑えることができる．

4. IER に基づく STP 探索

本章で提案する IER に基づく TP 探索では，まずユークリッド距離による TP 候補を生成し，その TP を道路網上での距離で検証する．ユークリッド距離での TP 候補生成はインクリメンタルに行われる．その結果を，道路網上の距離で検証する際に，STP 結果に含まれ得ない候補を目的地とする探索を抑制する検証方式を提案する．

4.1 ユークリド距離での STP 探索

Sharifzadeh ら [4] は，ユークリッド距離での OSR を求めるアルゴリズム LOAD を提案している．また，この効率化方式として，R 木を用いる R-LOAD を提案している．この R-LOAD は深さ優先に基づく探索方式であり，本稿で目的とするインクリメンタルな探索には適していない．ここでは，R 木を対象として最適優先探索により TP 候補をインクリメンタルに生成可能なアルゴリズムを提案する．このアルゴリズムを単純に発展させることにより，OSR や TPQ にも適用可能である．

POI 集合は R 木で管理されているものとする．旅行計画における出発地を s，最終目的地を d とするとき，ユークリッド距離での TP は R 木中の MBR を参照しつつ，最短の経路長が得られる可能性のあるノードをたどることにより得られる．以下に，具体的な探索方式を述べる．

図 1 は，s との位置関係が特徴的な 3 種類の目的地 d_1〜d_3，及び R 木中のある MBR(m) の位置関係を示している．s と d_1 の関係のように，2 点を結ぶ線分が m の内部を通過するとき，この MBR 中に存在する POI を途中で訪れる経路長の下限値は，この線分長自身となる．なぜならば，POI が線分上に存在する可能性があるためである．

d_2 と d_3 は s と結ぶ線分が MBR を通過しない場合である．まず，d_2 の場合には，寄り道路の距離の下限値は s と d_2 を結ぶ線分から m の最も近い頂点を通過する場合の距離となる．また，d_3 の場合には，s から発せられた光が m のある辺で反射して d_3 に到達する場合の距離，即ち辺上の点 A を通過する距離となる．

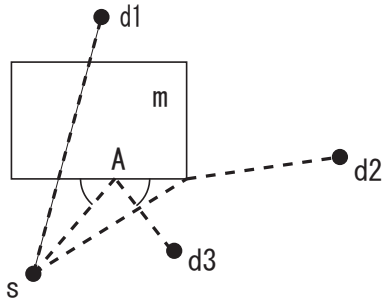

図 1　STP 経路の最小距離

以上で述べた 3 種類の場合に対する TP の距離の下限値を求める方式を以下に示す．s と d を通過する直線を L とし，s，d を両端点とする線分を L_{sd} と表わす．また，距離を求める対象の MBR を m とし，その 4 頂点を C_1〜C_4 とする．

Case1:　L_{sd} と m が交わる場合．寄り道路の距離の下限値は線分 L_{sd} の長さ，即ち $|L_{sd}|$ である．これは図 1 の d_1 の場合に相当する．

Case2:　線分 L_{sd} が X 軸に平行な辺の延長線，及び Y 軸に平行な辺の延長線に共に交わる場合．寄り道路の距離の下限値は s からある頂点 (C_i) を経由し d に至るパスの長さの最小値，即ち $\min(|L_{sC_i}| + |L_{C_i d}|) : \{i = 1, \ldots, 4\}$ である．

Case3:　それ以外の場合．L_{sd} は，m のある辺 (b) に対して m の外側に共に存在している．そのとき，図 2 に示すように，b を含む直線に対して d と対称な点 d' と s を結ぶ線分と b との交点を求め，その点を A とする．もし A が辺 b の範囲の外側であれば，Case2 と同様に最小距離を求める．一方，A が b の範囲に含まれれば $|L_{sd}| = |L_{sA}| + |L_{Ad}|$ とする．

以上で述べた処理により得られる，s から R 木のあるエントリー (MBR または点) e を経由して d に至る経路の距離を，以下では $L^E_{sd}(e)$ と表わすことにする．即ち，この関数は e が MBR のときには s から MBR 中の POI を経由して d に至る経路長の下限値を表わし，e が POI の場合にはユークリッド距離での実際の TP の距離を表わす．R 木の探索はヒープ (PQ)

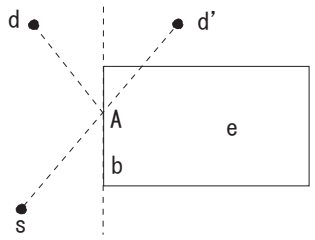

図 2　Case3 の場合

を用いた最適優先探索で行う．ここで，PQ は次のレコードを管理するものとする．

$$< L^E_{sd}(e), e > \quad (1)$$

この準備のもと，ユークリッド距離で最短のものから順に k 個の経由する POI 集合を求めるアルゴリズム (ESTP) をアルゴリズム 1 に示す．

このアルゴリズムではユークリッド距離での TP 経路長が短いものから順に求まる．1 行目の n は求まった経路の数を計数する変数である．また $RSLT$ は，途中に通過する POI からなる k 個の結果集合である．2 行目では，s，d 間のユークリッド距離 $d_E(s,d)$ と R 木の根ノード ($root$) からなるレコードを PQ に入れている．3 行目から 12 行目までのループを優先順位付きキュー PQ が空になるか，または指定された個数 (k) の POI が求まるまで繰り返す．

4 行目では，PQ から距離の下限値が最小の要素 r を取り出している．5 行目から 8 行目では，r が MBR の場合，その全ての子 ($e.c$)(ノード，又は POI) について PQ のレコードを作成し，それを PQ に投入している．10 行目は，e が POI の場合の処理であり，そのときは m を結果集合 $RSLT$ に加え，n の値をインクリメントしている．

このアルゴリズムでは，STP の距離が短いものから k 個求めているが，PQ の内容を保存しておき，3 行目以降を更に繰り返すことによりインクリメンタルな候補の探索が行える．

Algorithm 1 ESTP

```
1: n ← 0, RSLT ← ∅
2: PQ.poll(< d_E(s,d), root >)
3: while PQ.size() > 0 and n < k do
4:    r ← PQ.poll()
5:    if r.e instance of MBR then
6:       for all m ∈ e.c do
7:          PQ.offer(< L^E_sd(m), m >)
8:       end for
9:    else
10:      RSLT.add(m),n ← n + 1
11:   end if
12: end while
13: return RSLT
```

4.2　道路網上の距離での検証

ユークリッド距離で得られた候補に対して，道路網上での移動距離を求めることにより，その候補が道路網上での距離による STP であるかを検証する．即ち，候補 POI を p とするとき，s と p 間，及び d と p 間の距離の和を求め，全ての候補 POI の中で最小距離のもの (距離最小な上位 k 個) を検索結果とする．この検証では始点と終点が定まっている為，A*アルゴリズムを用いることができる．これを PWA*(pair-wise A*) 方式と呼ぶ．また，候補集合の全ての要素に対して，SSMTA*(single-source multi-target A*) アルゴリズム [11] により s 及び d から同時に距離を求めることもできる．これを SSMTA*方式と呼ぶ．以下の説明では，簡単の為，s，d，及び POI は道路網のノード (交差点や道路端点) に置かれているものと仮定するが，この制限を取り除くことは単純である．

以下に，これらの 2 つの方式による道路網上での距離での検証アルゴリズムを述べる．

(1)　ユークリッド距離で移動距離が最短となる POI を k 個求め，それらの集合を C，$(c_i \in C, i = 1, \ldots, k)$ とする．

(2)　s と C の全ての要素間，及び d と C の全ての要素間の道路網上での距離 ($L^N_{sd}(c_i)$) を求める．この為に PWA*または SSMTA*を用いる．

(3)　求まった距離の最大長 ($L^N_{sd}(c_k)$) 以下の POI を，再びユークリッド距離での検索で全て求め，それらの POI を新たに $C(c_i, i = 1, \ldots, j)$ とする．

(4)　(2) と同様に C の各要素に対して $L^N_{sd}(c_i)$ を求める．

(5)　$L^N_{sd}(c_i)$ の値が小さなものから k 個を結果として返す．

(3) では，道路網上での距離で求まった $L^N_{sd}(c_k)$ 以下の候補を全てユークリッド距離で再度求めている．これは，ユークリッド距離では k 番目までの候補に入らなかった POI の中にも，道路網上の距離では c_k よりも移動距離が短いものが存在する可能性があるためである．しかし，ユークリッド距離においてこれより距離が大きなものは候補から完全に除くことができる．

両方式とも，POI 密度が低いとき AIA に比して高い効率が達成できる．しかし，POI 密度が高いときには多数の候補に対して距離を求める必要があることから，特に PWA*方式において効率が劣化する．また，これらの方式では s と d，双方からの道路網上での距離が求まった後でないと，k 個の結果に入るか否かを決定できない．

これらの問題を解決するため，SSMTA*方式をベースとして，探索の際に全体の経路長を考慮した方式を提案する．これは POI 候補のある要素を c とし，現在の注目ノードを n とするとき，ヒープで管理するコストとして

$$d_N(s,n) + h(n,c)$$

但し，$h(n,c) = d_E(n,c) + d_E(c,d)$

を用いるものである．即ち，C を k-STP の候補 POI 集合とするとき，SSMTA* アルゴリズムにおいて，ノード展開の際に以下のコストを求め，その値でヒープを管理する．

$$Cost = \min_i \{d_N(s,n) + h(n,c_i)\}, c_i \in C$$

d からも上式の s と d を入れ替えた式で得られるコストを求め，それらを 1 つの PQ で管理する．PQ から上記 $Cost$ が最小のものを取り出し，そのノード n を展開する．n が c_i に到

達したとき，s または d から c_i までの距離が確定する．確定後は，c_i から s または d へのユークリッド距離の代わりに，道路網上の実際の距離を用いて $Cost$ を計算する．

以上の処理により，全体の距離が長い POI に向けた探索は進まず，探索範囲を狭めることができる．この方式を BDDC(bi-directional distance constraint) 方式と呼ぶ．Algorithm 2 にその疑似コードを示す．

Algorithm 2 BDDC

```
1: C ← ESTP(k, s, d)
2: c_n ← NextESTP(s, d)
3: nextDist ← d_E(s, c_n) + d_E(c_n, d)
4: PQ.offer(d_E(s, d), 0.0, s, null, s)
5: PQ.offer(d_E(s, d), 0.0, d, null, d)
6: while PQ.size() > 0 do
7:    v ← PQ.poll()
8:    if v has a POI then
9:       makePartialPath()
10:      modifyPQ
11:   end if
12:   if v.cost > nextDist then
13:      C ← C ∪ c_n
14:      c_n ← NextESTP(s, d)
15:      nextDist ← d_E(s, c_n) + d_E(c_n, d)
16:      modifyPQ
17:   end if
18:   if v.cost > maxPathLength then
19:      break
20:   end if
21:   for all nn ∈ neighbor(v, N_c) do
22:      decide c_i which gives minimum h(nn, c_i)
23:      d_N ← d_N(q, v.N_C) + d_N(v.N_C, nn)
24:      PQ.offer(< d_N + h(nn, c_i), d_N, v.N_C, v.N_S >
25:   end for
26: end while
```

1 行目では Algorithm 1 を用いて，ユークリッド距離での STP 候補を k 個求めている．2 行目ではインクリメンタル検索により，次に STP 経路が最短となる POI(c_n) を求めている．3 行目の $nextDist$ は c_n を経由する経路長の下限値である．以下の探索の途中で PQ から得られたレコードのコストが $nextDist$ を超えたとき，c_n を候補集合に加える．4 行目と 5 行目は s からと d からの探索の初期レコードを PQ に入れている．

6 行目からのループを PQ が空でない内繰り返す．7 行目では PQ からコスト最小の要素を取り出す．8 行目〜11 行目は，そのレコードが POI に達した場合，探索開始点 (s または d) から $v.N_C$ までの部分パスを得る (9 行目)．その発見された POI を s または d からの候補集合から取り除き，PQ の内容を更新する (10 行目)．12 行目から 17 行目は，$v.cost$ が $nextDist$ より大きくなったことにより，候補集合に新たな候補点が追加される場合の処理である．

18 行目は処理の終了条件であり，$v.cost$ が現在見つかっている k 番目のの TP の長さ ($maxPathLength$) より大きくなったとき，処理を終了する．

21 行目から 25 行目はノード展開であり，現在ノード $v.N_C$ に直接連結するノードを隣接リストから得て，それらの全てのノードに対してコストを求め，PQ のレコードを作成し，それらを PQ に追加している．

5. 性能評価

さいたま市の道路地図 (道路セグメント数，25,586) と疑似乱数により発生させた POI を用いて，MDQ，AIA，PWA*,SSMTA*,BDDC の各方式に対して性能比較実験を行った．この実験では，探索開始点，目的地，及び POI は全て道路網のノード上に置いた．実験に用いた POI の密度は，POI の個数と全ノード数との比で示している．従って，POI の密度が 0.01 とは，100 個のノードに 1 個の POI が存在する密度を示している．全てのプログラムは Java により記述した．また実験に用いた計算機は，Intel Corei7CPU(3.2GHz)，主記憶容量は 9GB である．

図 3 と図 4 は k=1 の場合，即ち経路長の最も短い STP 経路を 1 つ求める際に，上記の各方式により展開されたノード数と実行時間を示している．この中で MDQ は展開ノード数，計算時間ともに他の方式に比して膨大となるため，図からは除外している．MDQ は，BDDC に比して，実行時間，展開ノード数とも 100 倍以上となった．PWA*は POI 密度が高くなるに伴い展開ノード数，CPU 時間共に増加する．これは POI 密度が高い場合に，ユークリッド距離で求めた STP の候補数が増大し，それらに対して道路網上での距離での検査を必要とするためである．その際に，同じノードが何度も展開されることになる．一方，計算時間の増加率が低いのは，POI 密度が高い状況では多くの経路が位置的に近接しており，隣接リストの LRU バッファーでのヒット率が上昇するためと考えられる．一方 AIA は POI 密度が低い状況で，実行時間が急激に増加する傾向が見られる．これは POI 密度の低下と共に，AIA で探索する範囲が拡大することによる．一方 BDDC は展開ノード数及び実行時間共に他の方式に比して優れた性能を示している．

図 5 と図 6 は k=5 の場合の展開ノード数と実行時間を比較している．一般に，どの方式も k の値が増加するにつれ，展開ノード数，及び実行時間共に増加する．特に POI 密度が低い場合の AIA の実行時間の増加が顕著である．これは，AIA では k の増加につれ道路網上の広い範囲の探索が行われ，LRU バッファのヒット率が低下するためと考えられる．PWA*方式では展開ノード数の割に実行時間が短いが，これは 1 つのペアに対して最短路を求める際には道路網上の狭い範囲に探索が限定され，ヒット率が高く保たれている為である．

以上に示したように，BDDC は POI 密度の広い範囲において安定した性能が示されている．

6. おわりに

本稿では，道路網上の移動を想定した単純な旅行計画の為の高速アルゴリズム BDDC を提案した．実験により，提案方式が他の全ての既存方式に比べて，展開ノード数及び実行時間共

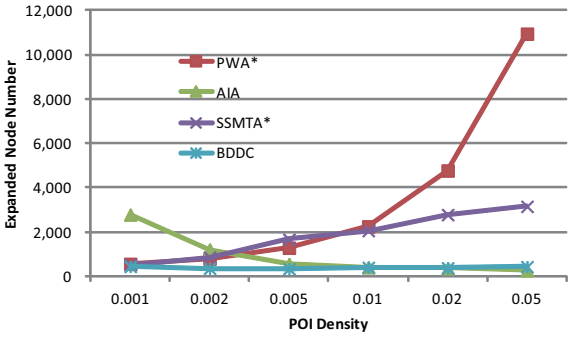

図 3　展開ノード数 ($k = 1$)

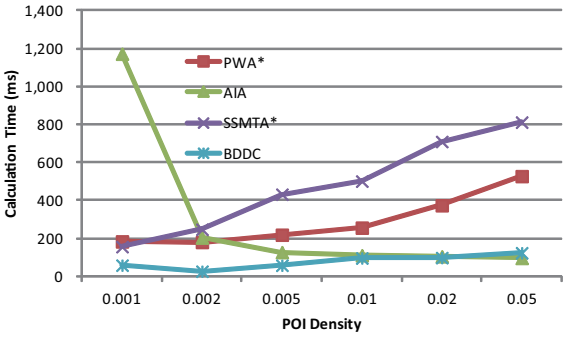

図 4　実行時間 ($k = 1$)

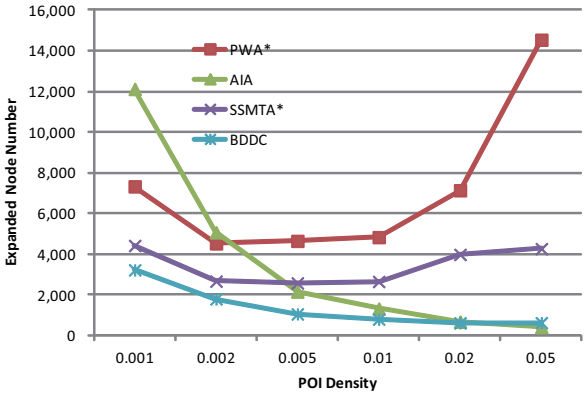

図 5　展開ノード数 (k=5)

に優れることを示した．特に，POI 密度が低い状況においては，ユークリッド距離により候補を生成し，その結果を道路網上での距離で検証する枠組みの有効性が高い結果が示された．

また，R 木を用いてユークリッド距離により STP 候補を高速に求めるアルゴリズムを示した．このアルゴリズムは，最適優先探索を採用しており，距離が短いものから順にインクリメンタルに候補を求めることができる．ここで述べたアルゴリズムは，より複雑な旅行計画，例えば TPQ や OSR に適用可能である．

今後，本稿で提案した IER に基づくアルゴリズムを OSR や TPQ に適用する．

謝辞 本研究は科研費 (21500093) 及び (2300337) の補助による．

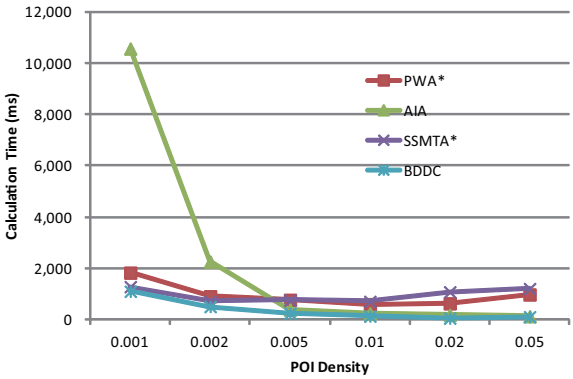

図 6　実行時間 (k=5)

文　　献

[1] S. Shekhar and J. S. Yoo: “Processing in-route nearest neighbor queries: A comparison of alternative approaches”, Proc. ACM GIS’03, pp. pp.9–16 (2003).

[2] F. Li, D. Cheng, M. Hadjieleftheriou, G. Kollios and S.-H. Teng: “On trip planning queries in spatial databases”, 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] M. Sharifzadeh, M. Kolahdouzan and C. Shahabi: “The optimal sequenced route query”, The VLDB Journal, **17**, pp. 765–787 (2008).

[5] 藤井, H. Htoo, 大沢：“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, **J93-D**, 12, pp. 2587–2596 (2010).

[6] H. Chen, W. S. Ku, M. T. Sun and R. Zimmermann: “The multi-rule partial sequenced route query”, ACM GIS ’08, pp. 65–74 (2008).

[7] D. Papadias, J. Zhang, N. Mamoulis and Y. Tao: “Query processing in spatial network databases”, Proc. 29th VLDB, pp. 790–801 (2003).

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

[9] K. Deng, X. Zhou, H. T. Shen, S. Sadiq and X. Li: “Instance optimal query processing in spatial networks”, The VLDB Journal, **18**, 3, pp. 675–693 (2009).

[10] 大沢, 藤野：“前処理を必要としない道路ネットワーク上での最短寄り道経路探索アルゴリズム”, 電子情報通信学会論文誌 D, **J93-D**, 3, pp. 203–210 (2010).

[11] H. Htoo, Y. Ohsawa and N. Sonehara: “Single-source multi-target A* algorithm for POI queries on road network”, WGIM 2011 Workshop, LNCS 7142, Splinger-Verlag, pp. 51–62 (2011).