D-012

# 道路網距離の簡易マテリアライズ化を用いた空間的 Skyline 検索の高速化
# Fast Spatial Skyline Query Method Using SMPV on Road Network Distance

越澤 和樹 * Kazuki Koshizawa　Htoo Htoo* Htoo Htoo　大沢 裕 * Yutaka Ohsawa

## 1. はじめに

地理情報サービスにおいて、道路網上のノード間の最短距離の計算は重要である。道路網の分割および道路網距離のマテリアライズ化を行った Simple Materialized Path View(SMPV) 構造を利用することで、高速な距離計算が可能となる。本稿では、道路網において利用者の興味のあるオブジェクトに絞り込む検索である空間的 Skyline 検索に対し、SMPV 構造を適用する。まず SMPV 構造を用いた最近傍探索アルゴリズムを提案し、さらに最短経路探索である Shortest Path Finder と共に従来のアルゴリズムへ組み込むことで、高速化を実現する。

## 2. 関連研究

### 2.1.Skyline 検索

Skyline 検索 [1] とは、多くのオブジェクトから利用者が興味を持つものに絞り込む検索である。例えば「安い」かつ「ビーチに近い」ホテルを探しているとき、2 つの尺度で他のホテルよりも劣っておらず、利用者の興味を引くようなホテルを探す問題となる。これらのホテルを Skyline と呼ぶ。

利用者が興味を持つオブジェクト、すなわち POI(point of interest) の集合 $P$ が与えられたとき、$p \in P$ が $p' \in P$ よりも全ての尺度で優れている場合に「$p$ が $p'$ を支配する」と表現する。図 1 に 6 つのホテル $P = \{a, b, c, d, e, f\}$ を示す。各ホテルは「値段」と「ビーチまでの距離」という 2 つの尺度を持っている。他のホテルに支配されない $a, c, e$ のみが Skyline となる。

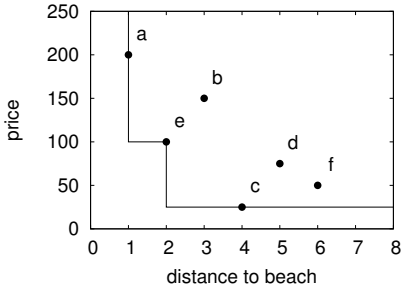


図 1: Skyline

### 2.2. 道路網における空間的 Skyline 検索

道路網における空間的 Skyline 検索 [2] とは、道路網上に POI 集合 $P$ と検索点集合 $Q$ があるときの Skyline を求める問題である。POI と検索点間の道路網距離を基に、他の POI に道路網距離で支配されない POI を探す。

空間的 Skyline 検索を解くアルゴリズムとして、Collaborative Expansion (CE)、Euclidean Distance Constraint (EDC)、Lower Bound Constraint (LBC)[2] が提案されている。CE は各検索点からのダイクストラ法により、並列に探索範囲を広げていく方法である。EDC はユークリッド距離での Skyline を最初に求め、その後 A*法で道路網距離により確認する方法である。LBC は道路網上での最近傍探索を基にした方法である。任意の検索点 $q$ から最近傍 POI を順次探索し、得られた POI から他の検索点までの道路網距離を A*法で計算する。

*埼玉大学

### 2.3.SMPV 構造

Simple Materialized Path View (SMPV)[3] とは、地図をサブグラフへ分割した後、サブグラフ毎に道路網距離のマテリアライズ化を行った構造である。従来のマテリアライズ化構造である Materialized Path View (MPV) や階層型の MPV と比較し、データ量を大幅に削減することができる。

SMPV と隣接リストを用いた 2 点間の最短経路探索アルゴリズムとして、Shortest Path Finder (SPF)[3] が提案されている。SPF は優先順位付きキュー (PQ) を用いた最適優先探索であり、A*法よりもはるかに高速に動作する。SMPV ではサブグラフ間に距離情報が存在しないため、実行時に A*法を用いて探索する。

SMPV において、サブグラフに含まれる各ノードは境界ノードまたは内部ノードに分類される。図 2 に地図を 3 分割した例を示す。黒い円は境界ノードであり、隣接する 2 つ以上のサブグラフに所属する。白い円は内部ノードであり、ただ 1 つのサブグラフに所属する。

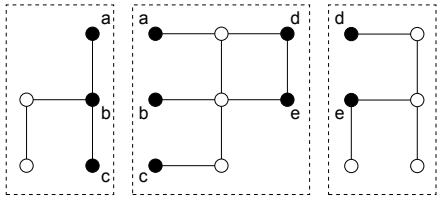


図 2: 3 分割後のグラフ

SMPV では BBDT 及び IBDT の 2 種類の距離テーブルを作成する。BBDT は境界ノード間の距離を、IBDT は内部・境界ノード間の距離を保持する。

## 3.SMPV 構造を用いた空間的 Skyline 検索

本稿では、道路網のような重み付き有向グラフ $G(V, E, W)$ を対象とする。$V$ はノード集合、$E$ はリンク集合、$W$ はリンクの重み (距離) 集合を表す。$SG_i(V_i, E_i, W_i)$ はグラフ $G(V, E, W)$ のサブグラフであり、$V_i \in V$、$E_i \in E$、$W_i \in W$ である。

従来の空間的 Skyline 検索アルゴリズムに対し SMPV を適用することで、検索の高速化が可能となる。EDC 及び LBC では距離計算に A*法を利用しているため、A*法を SPF[3] に置き換えることで達成できる。しかし、CE では A*法ではなくダイクストラ法を用いているため、2 点間の経路探索である SPF を適用することはできない。従って、SMPV を用いて全方向探索を行うアルゴリズム、NN on SMPV を提案する。NN on SMPV をダイクストラ法の代わりに CE の各検索点から実行することで、高速な検索を実現する。

### 3.1.NN on SMPV

NN on SMPV とは SMPV を用いた最近傍探索である。SPF が A*法を利用しているのに対し、NN on SMPV はダイクストラ法に基づいている。POI がいずれかのサブグラフに存在するため、NN on SMPV ではまず検索点から探索を開始し、近傍のサブグラフを徐々に展開していく。展開したサブグラフ内に POI を発見した際、POI までの道路網距離を計算し、検索点から近い順に POI を返す。

探索は PQ を使用した最適優先探索であり、以下に示すレコードを管理する。

$$< p, cost, dfs, prev, fSG, phase >$$

$p$ は現在のノードを表す。$cost$ 及び $dfs$ は検索点 $s$ から $p$ までの道路網距離を示す。ただし、探索中に POI を発見した際の $cost$ は POI までの道路網距離の下限を示す。$prev$ は 1 つ前に訪問したノードを表す。$fSG$ は $p$ の所属するサブグラフである。$phase$ は PHASE0 から PHASE2 までの処理の段階を表す。なお、PQ は $cost$ で昇順にソートされる。

探索処理の前に POI リスト $L$ を作成する。$L$ はサブグラフ毎に内部の POI を分類したものである。サブグラフの展開時に $L$ を参照し、POI が存在しない場合には内部を調べないことで処理を省略できる。例えば 5 つの POI$\{p_1, \ldots, p_5\}$ と 4 つのサブグラフ $\{SG_1, \ldots, SG_4\}$ が存在し、$SG_1$ に $p_1$ と $p_2$、$SG_2$ に $p_3$ と $p_4$、$SG_4$ に $p_5$ が含まれる場合、$L = \{L_1, L_2, L_3, L_4\}, L_1 = \{p_1, p_2\}, L_2 = \{p_3, p_4\}, L_3 = \{\}, L_4 = \{p_5\}$ と表される。

探索の初めに、検索点 $s$ の存在するサブグラフ $SG_s$ を展開する。$SG_s$ に含まれる全ての境界ノード $b_i$ において、$s$ からの道路網距離 $d_N(s, b_i)$ を IBDT から取得した後、以下のような PHASE0 のレコードを作成し PQ に追加する。

$$< b_i, d_N(b_i, s), d_N(b_i, s), s, SG_s, PHASE0 >$$

PQ から次に取り出したレコード $e$ が PHASE0 の場合、$e.p$ を境界ノードとして持つ各サブグラフ $SG_n$ を展開する。$e.p$ から $SG_n$ の各境界ノード $b_i$ までの距離 $d_N(e.p, b_i)$ を BBDT から取得し、以下のレコードを作成する。

$$< b_i, e.dfs + d_N(e.p, b_i), e.dfs + d_N(e.p, b_i), p, SG_n, PHASE0 >$$

ただし、サブグラフの展開時に POI リストを参照し、$SG_n$ に POI$p$ が存在すると判明した場合には、ユークリッド距離 $d_E(e.p, p)$ を $cost$ に加え、PHASE1 のレコードを作成する。

$$< p, e.dfs + d_E(e.p, p), e.dfs, e.p, SG_n, PHASE1 >$$

PQ から PHASE1 のレコードを取り出した場合、1 つ前に訪問した境界ノードから POI までの距離 $d_N(e.prev, e.p)$ を IBDT から取得し、PHASE2 のレコードを作成する。

$$< e.p, e.dfs + d_N(e.p, e.prev), e.dfs + d_N(e.p, e.prev), e.prev, e.fSG, PHASE2 >$$

最終的に PQ から PHASE2 のレコードを取り出した場合、次の最近傍として POI を返す。探索処理を続けることで、2 つ目以降の最近傍を順次得ることができる。

## 4. 実験結果

提案方式の性能評価実験には 2 種類の地図 (MapS 及び MapM) を使用した。MapS のノード数は 16,284、リンク数は 24,914、SMPV のサブグラフ数は 200 である。また、MapM のノード数は 81,233、リンク数は 109,373、SMPV のサブグラフ数は 1,000 である。実験環境は Intel Core i7 (3.40GHz)、16GB メモリ、実装には Java8 を用い、検索点数 $|Q|$ および POI 密度を可変パラメータとした。

MapS を用いた実験では従来方式と提案方式を比較する。図 3 は POI 密度を変更した場合の従来方式及び提案方式の比較結果である。いずれの提案方式も、10 倍以上高速化していることがわかる。

MapM を用いた実験では提案方式同士を比較する。図 4 は検索点数を、図 5 は POI 密度を変更した場合である。EDC on SMPV と LBC on SMPV の実行時間は指数関数的に増加するのに対し、CE on SMPV は検索点数や POI 密度が増加しても、処理時間の増加を小さく抑えることができ、特に POI 密度の影響を軽減できている。

## 5. 結論

従来の Skyline 検索アルゴリズム (CE、EDC、LBC) に SMPV を適用することで、10 倍以上の高速化を実現した。提案した 3 種の手法のうち、特に CE on SMPV は検索点数や POI 密度が増加しても、処理時間の増加を抑えることができる。

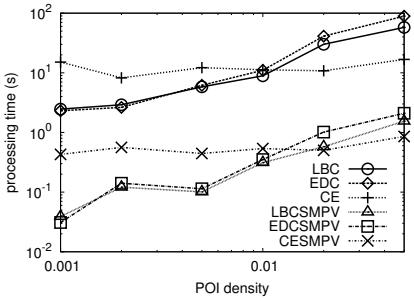


図 3: MapS における POI 密度と実行時間の関係

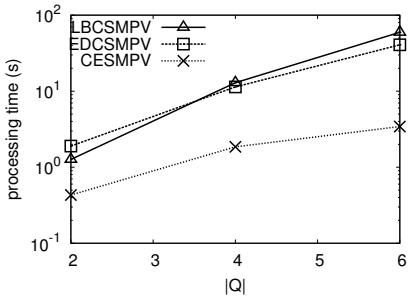


図 4: MapM における検索点数と実行時間の関係

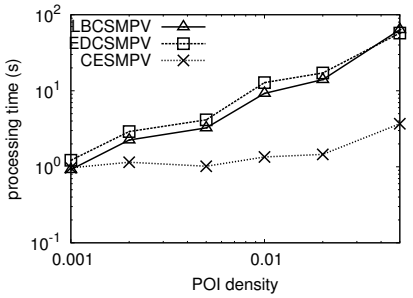


図 5: MapM における POI 密度と実行時間の関係

## 参考文献

[1] S. Borzsony, D. Kossmann, and K. Stocker. The Skyline operator. *Proceedings 17th International Conference on Data Engineering*, pp. 1–20, 2001.

[2] Ke Deng, Xiaofang Zhou, and Heng Tao Shen. Multi-source skyline query processing in road networks. *Proceedings - International Conference on Data Engineering*, pp. 796–805, 2007.

[3] Aye Thida Hlaing, Htoo Htoo, Yutaka Ohsawa, Noboru Sonehara, and Masao Sakauchi. Shortest path finder with light materialized path view for location based services. pp. 229–234, 2013.