DBSJ Journal, Vol.12, No.3
February 2014

部分領域の全対経路索引を用いた経路スカイライン問合せ処理法の提案

Route Skyline Query Processing Using Regional All-Pairs Path Indices

蒲原 智也 ♥　小池 実 ♦
壷内 貴弘 ♠　上島 紳一 ♣

Tomoya KAMBARA　Minoru KOIKE
Takahiro TSUBOUCHI　Shinichi UESHIMA

本稿では，多次元コストグラフに対する経路スカイラインの問合せ処理法について議論する．経路スカイライン問合せは，利用者の選好を表すコスト関数に対するパレート最適な経路集合を予め抽出する操作であり，経路計画，散策計画，施設配置，工程管理，グラフ航行等の様々な分野での応用が期待できる．経路スカイライン問合せは，多次元データ集合に対するスカイライン問合せと異なり，グラフ探査処理を必要とする．ここでは，経路スカイラインの任意の部分経路がスカイラインであることに着目し，グラフ分割で得られた部分領域の全対経路索引を生成して，グラフ探査における展開途中の経路の刈込み処理を行う 2 段階のアルゴリズムを提案する．事前処理では，グラフ分割により得られた部分領域に対する経路索引として全ノード対に対して経路スカイラインを求め，探査実行処理では経路索引を用いて経路スカイラインを構成的に計算する．提案手法を評価するため，国土地理院発行の数値地図を用いてシミュレーションを行い，事前処理時間，探査実行時間，属性数などの影響等について提案手法，ナイーブな探査法，距離索引を用いた探査手法と定量的に比較する．

In this paper, route skyline query processing is studied over multidimensional cost graphs. Route skyline query processing is to extract the pareto optimal routes between specified start/goal nodes, that minimize the given linear cost function with user-defined non-negative coefficients. It can be applicable to various problems in route plannings, city navigation, facility placement, process management, graph navigation, etc. Route skyline query processing requires route search on the target graph, while data set are explicitly given beforehand in skyline problems for multidimensional data points. Here the authors propose a two-stage algorithm that consists of preprocessing and search processing to extract route skyline, utilizing all-pairs path indices of subregions of the graph, based on the property that partial routes of skyline are also skyline. We employ spatial tessellation technique and generate all-pairs path indices to subgraphs to prune active search paths, and generate route skyline in a bottom-up fashion. Numerical simulations for digital road maps issued by GSI are provided to verify the efficiency of the proposed algorithm quantitatively and to compare it with a naive method and a distance-index based approach.

♥ 正会員　関西大学先端科学技術推進機構
♦ 学生会員　関西大学大学院総合情報研究科
♠ 学生会員　関西大学総合情報学部
♣ 正会員　関西大学大学院総合情報研究科
ueshima@res.kutc.kansai-u.ac.jp

1. はじめに

スカイライン問合せは，利用者の選好係数を持つコスト関数に対する最適な部分集合を予め抽出する操作であり，様々な分野で利用されている [1]．多次元データ集合に対するスカイライン問合せでは，対象集合が明示的に与えられ，データ間の支配関係を調査しながら，支配されないデータ集合を解として抽出する [2]．高次元データ集合に対しては，分割統治法などの効率的な計算方法や並列処理環境などを利用した問合せ処理法が提案されている [3, 4, 5, 6]．

これに対して，近年，多次元コストグラフに対して，経路コスト関数を定義し，他の経路に支配されない経路を抽出する経路スカイライン問合せの研究が進められている [7, 10, 11, 12]．経路スカイラインは，利用者毎の選好経路を求めることができるため，経路計画，散策計画，施設配置，工程管理，グラフ航行等の様々な分野で応用できると考えられる [8, 9, 12]．経路スカイライン問合せでは，多次元データ集合の場合と異なり，予め 2 地点間を結ぶ経路集合が明示的に与えられず，グラフの探査が必要である．

本稿では，経路スカイライン内の経路の任意の部分経路がパレート最適である点に着目して，部分領域に対する経路索引を生成し，グラフ探査に利用するアルゴリズムを提案する．提案手法は生成処理と探査実行処理の 2 段階から構成される．生成処理ではグラフを分割し，その部分領域内ですべてのノード間の全対経路スカイラインと隣接関係等の経路索引を予め求めておく．一方，探査実行処理では与えられた出発地点と目的地点に対して，まず目的地点からの経路見積もりを求め，それと経路索引を利用して最適な経路を組合せて延伸する処理を実行して経路スカイラインを構成的に抽出する．2 章で関連研究について述べ，3 章で経路スカイラインの定義と性質を述べる．4 章でナイーブな探査法，5 章で提案アルゴリズムについて述べ，6 章で提案手法を評価する．

2. 関連研究

Kriegel ら [7] は，多次元コストグラフ上で，利用者選好を係数とする経路コスト関数の最小化問題に対して経路スカイラインを議論している．参照点埋込み法としてグラフ上に参照点を複数配置し，参照点への距離見積り指標を作成する事前処理を行い，探査実行時における経路の刈込み処理の基準として用いる手法を議論している．[8, 9] では，利用者への利便性を高めた GUI を構築し実応用へ適応し易く工夫している．Yang ら [10] は，多次元コストグラフをクラスタ分割し，各クラスタの外部索引と内部索引を事前に計算しておき，経路の刈込みに用いる手法を提案している．外部索引としては，(i) 各クラスタの境界と他のすべてのクラスタの境界間の属性毎の最短経路コスト，また，内部索引としては，(ii) 各クラスタ内部を通過するすべての境界間のスカイライン経路集合と，(iii) 各クラスタ内の任意の 2 ノード間の属性毎の最短経路コストを求めている．これらの索引を用いて，ノードの展開処理で，目的点までの残存距離の見積もりと，探査における経路支配関係の調査に利用している．しかし，[10] は，利用者定義部分を含まない固定的な評価関数の最適化を目標としており，提案手法が目指す個人化手法へ適用等の面で利用しにくい．また，クラスタ分割法は，ノード次数が力の法則を満たすネットワークを対象としており，必ずしも提案手法と直接の比較は難しい．同様に Tian ら [11] も各属性の最小経路コストを用いた支配関係による刈込みを行う探査手法を提案している．Deng ら [12] は，複数の質問点から相対的なネットワーク距離を求め，協調的展開手法，経路距離の下限を用いる手法等を提案している．また，Mouratidis ら [13] は，予めグラフ上に施設集合を与え，候補集合点からそれらの施設への多次元コストに関して支配関係を定義し，漸進的なスカイライン抽出手法と，Top-k スカイラインの段階的な抽出法により解の部分集合を抽出している．また，ユークリッド空間内のオブジェクト集合を対象とした空間スカイライン抽出問題として，Sharifzadeh ら [14] は，複数の質問点集合からオブジェクト集合までの距離を用いてオブジェクト間の支配関係を定義し，空間のボロノイ分割を用いた解集合の抽出手法を提案している．

これらの関連研究に対し提案手法では，予めグラフを分割して，部分領域毎の全対ノード間の経路スカイラインと距離指標を

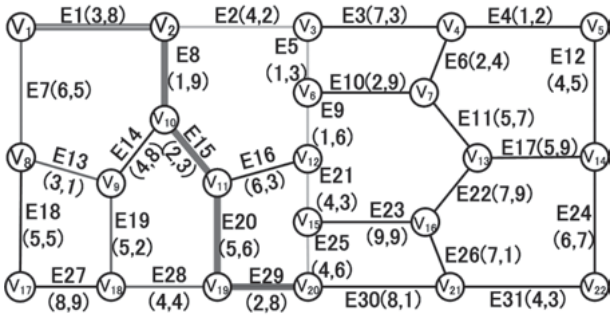

図 1　2 次元コストグラフ $G(V,E,W)$ の例 $(d=2)$

Fig. 1　2-dimensional Cost Graph $G(V,E,W)(d=2)$

抽出し解候補の部分となる経路集合を求めておくことで，事前処理と探査実行時の処理に負荷を分散させている．また，探査実行処理では経路を延伸することで，解経路集合を構成的に抽出している．これらの特徴により，グラフの部分的更新や規模の拡大へ対応しやすく，経路選択の個人化等への発展も期待できる．

3.　経路スカイライン

3.1　定義

[定義 1] (多次元コストグラフ) V をノード集合，E をエッジ集合 $(E \subset V \times V)$，$W \subset \mathbb{R}_+^d$ を $e \in E$ の非負の値を持つ d 次元コスト値 (属性値) とするグラフを多次元コストグラフ $G(V,E,W)$ という (単に，G とも書く)．本稿で G は無向グラフとする．

G を道路網とみなす場合は，交差点をノード，交差点を結ぶ道路片をエッジ，道路片のコストをエッジの属性値と考える．属性数 2 に対する G の例を図 1 に示す．G 上の経路コストを次のように定義する．

[定義 2] (経路と経路コスト) $G(V,E,W)$ 上の経路 p を連接するノードの列 $p=(v_1,v_2,\ldots,v_k), v_i \in V$ とし，構成エッジ (v_i,v_{i+1}) の第 l 番目のコスト $w((v_i,v_{i+1}))_l (1 \leq i < k, l = 1,2,\ldots,d)$ を用いて p の第 l 番目のコスト $cost^l(p)$ を次のように定義する．$cost^l(p) = \sum_{i=1}^{k-1} w((v_i,v_{i+1}))_l$ つまり経路 p は，経路の構成ノードを繋ぐ隣接エッジのコストの和である．この時，p の d 次元コストベクトルは，$cost(p) = (cost^1(p), cost^2(p), \ldots, cost^l(p))'$ となる．$'$ は転置を示す．本稿では，G 上で，より小さいコストを持つ経路が他より優れている経路とする．また探査を行う際に経路は，巡回しないものとする．

[定義 3] (利用者選好と選好関数) 経路に対する d 次元の利用者の選好ベクトル $\Pi \subseteq \mathbb{R}_{\geq 0}^d$ と，経路 p に関する選好関数 $Pref_\Pi(p)$ をそれぞれ定義する．$\Pi = (\pi_1,\pi_2,\ldots,\pi_d)' \in \mathbb{R}_{\geq 0}^d \backslash \{\vec{0}\}$，$Pref_\Pi(p) = \sum_{l=1}^d \pi_l \cdot cost^l(p) =< \Pi, cost(p) >$．

[定義 4] (経路間の支配関係) $G(V,E,W)$ 上で，出発地点 v_s と目的地点 v_t の 2 地点間を繋ぐすべての経路を $\mathcal{P}(v_s,v_t) = \{(v_s,\ldots,v_{i_j},\ldots,v_t)_j\}$ とする．$p,q \in \mathcal{P}(v_s,v_t)$ に対して，$cost^l(p) < cost^l(q) (\exists l, 1 \leq l \leq d)$ かつ $cost^j(p) \leq cost^j(q) (1 \leq \forall j \leq d \wedge j \neq l)$ が成り立つ時，経路 p が経路 q を支配しているという．上定義で経路 p は，第 l 属性について優れかつ他の属性 j に関しても同等か優れていることを意味する．

[定義 5] (経路スカイライン) $G(V,E,W)$ 上に出発地点 v_s と目的地点 v_t が与えられた時，経路集合 $\mathcal{P}(v_s,v_t)$ 内で他の経路に支配されない経路集合を**経路スカイライン** $RS(v_s,v_t)$ という．以下, 単に RS とも書く．

また，経路集合 $\mathcal{P}(v_s,v_t)$ から，経路スカイラインを抽出する問合せを経路スカイライン問合せという．本稿では，Π に対して選好関数の最小化を考える．

3.2　経路スカイラインの性質

[定義 2,3,4] より, 経路スカイラインが次の性質を持つことを容易に確かめることができる．

(性質 1)　G 上の 2 点 v_s, v_t に対する $RS(v_s,v_t)$ に属する任意の部分経路は，その部分に対する経路スカイラインである．

(性質 2)　G 上の 2 点 v_s, v_t に対して，$RS(v_s,v_t) = RS(v_t,v_s)$ である．

(性質 3)　v_s から出発した任意の長さの経路 r が，ある経路スカイライン $p_o \in RS(v_s,v_t)$ に支配される時，r を v_t まで延長しても p_o に支配される．

(性質 1,3) は経路展開で各ノードでの刈込み判定と, 目的地点での刈込み判定に用いることができる．(性質 2) は目的地点からの後向き展開の際に距離見積りに用いる．

[例 1] 図 1 の 2 点 v_1 v_{20} に対して，v_1 から展開すると，次の経路からなる経路スカイライン $RS(v_1,v_{20})$ を得る．

$p_1 = (v_1,v_2,v_{10},v_{11},v_{19},v_{20}),$　　$cost(p_1) = (13,34)'$
$p_2 = (v_1,v_8,v_9,v_{18},v_{19},v_{20}),$　　$cost(p_2) = (20,20)'$
$p_3 = (v_1,v_2,v_3,v_6,v_{12},v_{15},v_{20}),$　　$cost(p_3) = (17,28)'$

また，p_1 の部分経路 $p_1' = (v_1,v_2,v_{10})$(コスト $(4,17)'$) は，$\mathcal{P}(v_1,v_{10})$ 内の他経路に支配されず，経路スカイラインであること，すなわち $p_1' \in RS(v_1,v_{10})$ を容易に確かめることができる．同様に $p_1'' = (v_{10},v_{11},v_{19}) \in RS(v_{10},v_{19})$ 等も成り立つ．また,G は無向グラフのため $RS(v_1,v_{20}) = RS(v_{20},v_1)$ である．このように，2 ノード間の経路スカイラインとして複数の経路が存在する．□

4.　ナイーブな探査法

4.1　経路スカイライン探査の特徴

$RS(v_s,v_t)$ を求める場合，v_s から G を展開して経路を生成するが，エッジが複数の属性を持つため, 一度, ある経路から展開されたノードが別の経路のノードとして展開される．探査すべき経路は，アクティブ探査リストに格納する．

3.2 節により, 経路の探査を停止する場合は,

- 途中ノードで他経路に支配される
- 展開途中の経路が, 既に v_t に到達した経路に支配される

の 2 つに限られる．つまり，展開先ノードにおいて到達した経路の支配関係を調査し，経路を刈込む必要がある．また，v_t に到達した経路がある場合は，アクティブ探査リストの経路を刈込む必要がある．

4.2　経路属性空間と探査モニタリング

経路属性空間 (RAS)[7] で経路間の展開過程を確認する (図 2)．

- RAS 上の各点は，v_s から v_t への探査途中に生成されるすべての経路の経路コストを示す．
- G のエッジコストが正であるため，各点は探査の進行に従って RAS 上で右上方向に移動する．
- G の接続関係に依存して，RAS 上の 1 つの点がノードにおいて複数の点に分化する．

RAS 上で，ある経路 p が定める支配領域に含まれる経路群の探査を停止する必要がある．

これを実現するために, 各ノード v_i において RAS を持たせ, 到達した経路のコストをプロットして相互の支配関係を調査する．この方法を**ナイーブな探査法**と呼ぶ．

[例 2] 図 1 上で，$v_s = v_1$,$v_t = v_{20}$ とした場合の探査についてみる．図 2 は RAS 上での経路コストの動きを示し．対応する G の展開を図 3 に示す．展開については，5.2.2 の属性優先順序による．

まず，v_1 から隣接ノード v_2, v_8 を展開し RAS にプロットする．次に，v_2, v_8 から展開を継続し，第 7 ステップ目の RAS が図 2(i) である．この時，v_{12} に 2 つの経路が到達しており，紫経路 $(v_1,v_2,v_{10},v_{11},v_{12})$(12,23) は橙経路 (v_1,v_2,v_3,v_6,v_{12})(9,19) に支配されている．このため紫経路は展開を停止する．同図でアクティブ探査リストの中のノード (波頭ノード) を黒塗りしている．

更に展開を継続し，第 13 ステップ目に目的地 v_{20} に到達した赤経路 $(v_1,v_2,v_{10},v_{11},v_{19},v_{20})$(13,34) ができる．

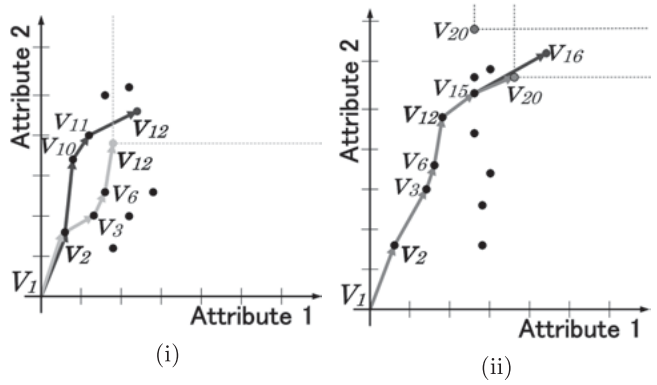

図 2　経路属性空間上での展開途中の経路コストの動き

Fig. 2　Behavior of Route Costs in Route Attribute Space

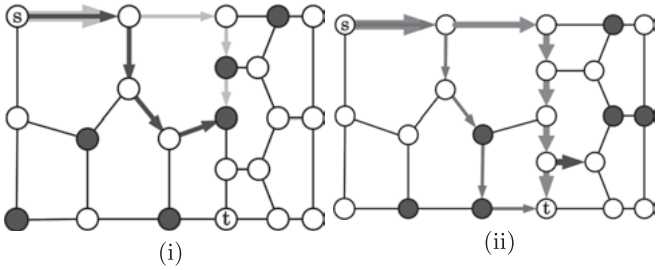

図 3　出発地点 v_1，目的地点 v_{20} に対する G の展開 (●:展開の波頭)

Fig. 3　Route Search on G from v_1 to v_{20}(●:wave front)

第 16 ステップ目に経路 $(v_1, v_2, v_3, v_6, v_{12}, v_{15}, v_{16})$(22,31) は，緑経路 $(v_1, v_2, v_3, v_6, v_{12}, v_{15}, v_{20})$ (17,28) に支配される．前者の経路 $(v_1, v_2, v_3, v_6, v_{12}, v_{15}, v_{16})$ は以後の展開を停止する．その後もう一つの経路が第 25 ステップ目に発見され，展開すべきノードは第 27 ステップ目で空になり，(v_1, v_{20}) 間の経路スカイライン探査は終了する．□

5.　提案手法

5.1　事前処理

3.2 節の性質を利用して, 次の (1),(2) の処理を行って部分領域毎の経路索引を生成する．

5.1.1　処理内容

(1) **グラフ分割**　まず G をホップ数 (エッジ数) を用いて互いに素な部分グラフ $\{G_1, G_2, ..., G_n\}$ に NV 分割し，G_i の境界と隣接関係を求める (図 4)．

(2) **全対計算**　次に G_i 内で全ノード対に対して経路スカイラインを計算する．ナイーブな探査法により，全ノード対 $(s, d)(s, d \in G_i)$ に対して，s から経路を展開して $RS(s, d)$ を求める (図 5)．

Note：(NV 分割) [16, 18] V の部分集合を母点集合として与え，母点以外のノードに対して最も近い母点を決定し，G のノード集合を分割した図である．並列 Dijkstra 法 [17] によって生成する．E に関しても，任意の地点から各母点への重みを考えれば，各エッジの所属母点を決定できる．NV 分割によって G を母点数の部分領域に分割できる．

分割については，母点からのホップ数（エッジ数）で分割する．この時，母点の選択は部分領域の処理の均一化を図るため，ランダムに行う．これによりグラフの粗密を考慮する必要がない．部分領域どうしを結ぶエッジを**境界**と呼ぶ．境界は両端のノードの所属母点が異なるエッジである．境界の両端のノードを**境界ノード**と呼ぶ．

5.1.2　経路索引

各部分領域 G_i の経路索引は次の通りである．

(1) 全対経路スカイライン：G_i 内の任意のノード s が, G_i 内の他のすべてのノードに至る経路スカイライン $\{RS(s, d)|d \in G_i\}$ を持つ．

(2) 領域間の隣接関係：G_i の母点は，境界を通して隣接する領域と, 各隣接領域との境界エッジの属性毎の最小値を持つ．

(3) 領域見積もり：G_i の母点が, G_i 内の境界ノードのすべての組合せの属性毎の最小値を持つ (**(1)** より参照する)．探査実行処理時に (v_s, v_t) が与えられた時に展開中ノードから v_t への距離見積もりに用いる．

グラフ分割は次の特徴を持つ．

- グラフ分割が経路スカイライン内の各経路を分割しており，3.2 節 (性質 1) を利用できる．
- 各 G_i 内の経路スカイラインを独立に計算できる．
- 事前処理で得られた経路索引を用いて G 全体の経路スカイラインを求めることができる．

[例 3] 図 4 に，図 1 の NV 分割例を示す．ここでは母点を $\{v_3, v_6, v_{13}, v_{15}, v_{18}, v_{20}\}$（色塗り円）としている．母点領域を同一色で，境界を黒実線と ⋈ で示す．

次に, 各部分領域内で，境界ノード間のすべての組合せに対して経路スカイラインを計算する (図 5)．但し，この例の v_6 に対する部分領域のように，1 つのノードからなる部分領域内での計算はしない．経路スカイライン 1 本には複数の経路が含まれる．事前処理は，G を部分領域単位のグラフ書換えに相当する．□

5.2　探査実行処理

5.2.1　処理内容

G 上の経路スカイラインは, 一般に, 複数の G_i を跨いだ経路になる．このため，探査実行処理では, G_i 上で求めた経路スカイラインを境界で延伸する．その時点で, 経路間の支配関係を判定して刈込み処理を行いながら，v_t まで経路の延伸を繰り返す．v_t への到達経路が発生すると，探査途中の経路との判定，アクティブ探査リスト内の経路との判定を行う．探査実行処理は，G 上で経路の延伸操作と探査停止判定の両方の役割を担う．

5.2.2　グラフ展開における属性優先順序

アクティブ探査リストにある複数の経路に対して, 第 1 属性について最小値判定を行ない, それらが同一の場合, 第 2 属性の最小値判定を行う．以下, 第 d 属性まで繰り返す．最後まで同一の場合は, 最初に選択した経路を展開する．

5.2.3　探査停止基準

探査停止基準は以下の通りである．事前処理ではエッジの展開の際に，探査実行処理では経路の延伸の際に用いる．

- 探査停止基準 **I**：目的地点へ到達した経路と探査途中経路間の支配関係による探査停止．
- 探査停止基準 **II**：複数の探査途中経路が同一ノードへ到達した場合の支配関係による探査停止．
- 探査停止基準 **III**：目的地点へ新たに到達した経路とアクティブ探査リスト内経路の支配関係による探査停止．

5.3　距離見積りを用いた経路探査

探査実行処理において, A*経路探査法 [15] を応用して，経路を延伸する際，展開に方向性を与える．ここでは探査中の経路の展開先領域から v_t 領域までの各属性に関して残り距離の見積もりを計算する．

[隣接関係グラフの重み付け] 図 5 の各部分領域をノードと見なし，各部分領域の隣接関係を維持したグラフを隣接関係グラフと呼ぶ (図 6 左)．

[エッジへの重み付け] 同図で, 部分領域間エッジに対して, 次のように属性値を与える．すなわち, (v_3, v_{18}) 領域間のエッジの属性値としては, v_3 領域,v_{18} 領域を繋ぐ境界間の経路の持つ最小値を属性毎に与える．

例えば，図で v_{18} 領域と v_3 領域間のエッジは 2 本あり，v_1, v_8(6,5) と v_2, v_{10} (1,9) のうち，各属性値において最小の (1,5) を領域間のエッジの重みとする．v_{18} 領域と v_{20} 領域間のエッジの重みは (2,3) である．他の領域間のエッジ属性も同様に決める．

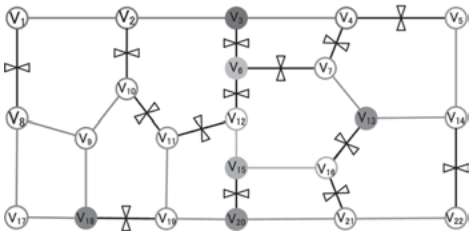

図 4　NV 分割, ⋈:境界, 色塗り円：母点
Fig. 4　NV Partition (⋈:boundary, painted circle : generator)

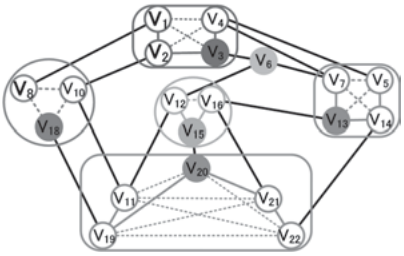

図 5　G_i の全対経路スカイライン
Fig. 5　G_i All-Pairs Route Skyline

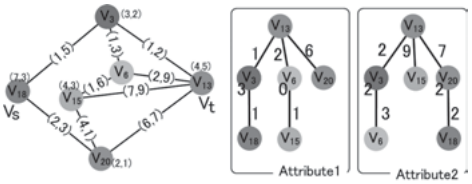

図 6　$\{G_i\}$ の隣接関係グラフの重み付けと v_t への距離見積もり
Fig. 6　Weighting Pattern of Adjacency Graph on G_i's and Attribute-wise Distance Estimation to v_t

[ノードへの重み付け] この時，領域見積もりとして，経路索引を参照して，領域内境界ノードからの他のすべての境界ノードに対する組合せを参照して，属性毎の最小の属性コストを母点に持たせる．これにより，当該領域通過のためのコスト見積もりを得る．

たとえば，v_3 領域において全対スカイライン経路は 6 本あり，その中の各属性の最小コスト (3，2) を v_3 領域の重みとする．

[距離見積り] 図 6 で v_s が v_{18} 領域に，v_t が v_{13} 領域に属する場合を考える．重み付けされた隣接関係グラフに対して，各属性値について Dijkstra 法を用いて v_t 領域から最短経路を求めて展開する．この時，v_3 領域を通過するための見積もりとして，前節のノードへの重み付けで述べた領域見積もり (3,2) を用いて，v_t 領域までを見積もる．つまり，v_t 領域から各属性に関して最短経路木を構成する（図 6 右）．

次に，v_s 領域（$=v_{18}$ 領域）から次の展開対象は，v_3 領域，v_{20} 領域である．展開順序は，第 1 属性の最小値，第 2 属性の最小値の順に見る．v_3 領域については，領域見積もりが (3,2) であるので，v_t 領域までの見積もりは (1,5)+(3,2)+(1,2)=(5,9) である．一方，v_{20} 領域については，v_{20} 領域の領域見積もりが (2,1) であるので，v_t 領域までの見積もりは (2,3)+(2,1)+(6,7)=(10,11) である．よって，v_3 領域の持つ境界点に対し延伸する．

5.4　提案手法の特徴

- **経路索引の生成と探査時の刈込みでの利用：**　事前処理で小規模グラフに分割し，全対ノード間の経路スカイラインと領域見積もり等を経路索引として計算している．次に探査実行処理として G 上の 2 点間に対する経路スカイラインを 2 段階に抽出している．
- **部分経路のパレート最適性の利用：**　経路スカイラインの任意の部分経路がパレート最適である点に着目し，分割された経路を延伸して，経路スカイラインを構成的に求めている．
- **探査実行処理での見積もり型経路探査：**　探査実行処理において，距離指標を用いた目的地点から後ろ向きの最短経路探査を属性毎に行い，前向きの延伸処理での刈込みに用いる．
- **境界での処理に帰着：**　G を予め分割して部分領域毎に経路探査をしている．事前処理で各部分領域内・部分領域間の処理を終えているため，探査実行処理では，部分領域の持つ境界での処理のみの実行でよく，部分領域内探査を省略できる．

6.　評価

6.1　対象データと評価方法

提案手法を評価するため数値シミュレーションを行う．

[対象地図データ] 国土地理院数値地図 2,500(空間データ基盤，1/2,500 縮尺)[19] を用いた．エッジ属性として，第 1 属性はノード間の実距離（m)，小数点以下 3 桁を用い，第 2 属性以降に 2 桁の独立な一様乱数を与えた．

[実行環境] SunOS 5.10 Generic 144500-19 32 sparcv9 プロセッサ (2530 MHz，浮動小数点プロセッサ付)，主記憶 32GB，Java version 1.5.0-20 である．

[比較対象] 次の 2 つの手法を用いた．

表 1　部分領域 G_i の数と領域あたり境界ノード数 ($|V| = 960$)
Table 1　$|\{G_i\}|$ and ♯Boundaries of G_i ($|V| = 960$)

p	$\lvert\{G_i\}\rvert$	♯Bound	♯Bound/G_i
1/16	59.79	367.97	5.71
1/32	29.96	254.07	8.12
1/64	15.20	172.92	11.25
1/128	7.60	113.07	15.08

(I) グラフ分割を用いず，(v_s, v_t) に対してノード毎に RAS を用いた支配関係調査のみを行うナイーブな探査法
(II) 参照点埋込み法 [7] による距離索引を用いた探査法 (ARSC)

[評価方法]

- **事前処理：**提案手法と (II) では，原地図データを読み込み，経路索引と距離索引を作成するまでを計測した．これらは中間ファイルに出力する．
- **探査実行処理：**中間ファイルを読み込んだ後，(v_s, v_t) を指定してから，すべての $RS(v_s, v_t)$ を返すまでを測定した．(I) は事前処理がないため直接探査処理を行っている．

Note：(参照点埋込み法 ARSC）[7] G 上のノード上に予め参照点を配置し，各ノードから参照点までの最短経路を属性毎に求めて距離索引とする方法である．探査実行時に三角不等式を用いて展開先ノードから目的地点までの属性毎の最低コスト値を見積もる．

6.2　事前処理に関する評価

・経路索引と距離索引の生成時間：図 7 に，母点選択確率 p と経路索引の生成時間 (msec) の関係を示す．提案手法は，属性数 d の増加と共に，生成時間は増加し，$p = 1/128$ の時，約 17,468msec である．また p 値が小さくなるほど，1 領域当たりのノード数は増加し，全対経路スカイラインの計算時間が増大する．

一方，参照点埋込み法は，p の影響はなく，$d = 2$ の場合，約 252msec である．G の全ノードに対し参照点から Dijkstra 法を用いて探査するため，参照点数 r，ノード数 $|V|$，属性数 d に比例し計算コストが増加する．

・部分領域に関するデータ: 表 1 に原地図データであるグラフ G に対して，母点選択確率 p で生成された部分領域数 $|\{G_i\}|$ と，G 全体で生成された全境界数，各部分領域 G_i が持つ平均境界ノード数を示す．同表は生成試行回数が 100 回の平均値である．同表より部分領域数が $1/p$，平均ノード数は $|N|/p$ であり，領域当たりの境界数 ♯Bound/G_i が 1 領域あたりのノード数に比較して小さく留まっている．

・境界ノードが持つ境界間経路スカイライン数の平均値：図 8 に，G_i 内で 1 つの境界あたりの境界ノード間の経路スカイラ

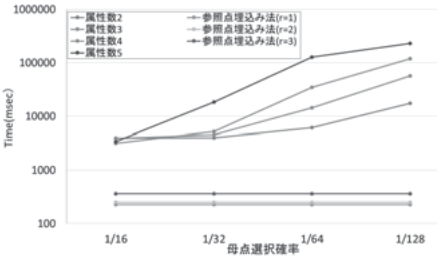

図 7 事前処理時間 (msec) ($|V| = 960, d = 2, 3, 4, 5$)
Fig. 7 Preprocessing Time (msec) (Proposed Method, ARSC) ($|V| = 960, d = 2, 3, 4, 5$)

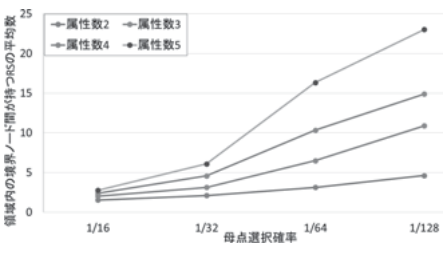

図 8 G_i 内の境界ノードが持つ経路スカイライン数の平均 ($|V| = 960, d = 2, 3, 4, 5$)
Fig. 8 Average Number of Route Skylines between Boundary Nodes in G_i ($|V| = 960, d = 2, 3, 4, 5$)

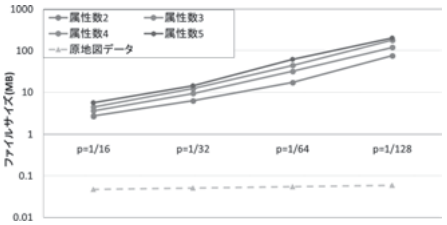

図 9 全対経路索引と原データのファイルサイズ比較 (MB) ($|V| = 960, d = 2, 3, 4, 5, p = 1/16 - 1/128$)
Fig. 9 File Size of All-Pairs Path Indices and the Original Data($|V| = 960, d = 2, 3, 4, 5, p = 1/16 - 1/128$)

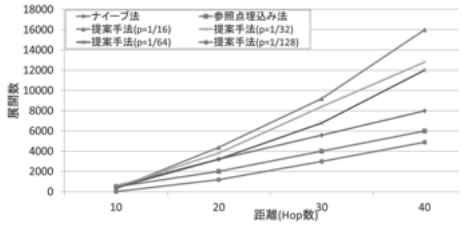

図 10 アクティブ探査リストの展開数と (v_s, v_t) 間ホップ数 (提案手法, ナイーブ手法,ARSC($r = 3$))($|V| = 960, d = 2$)
Fig. 10 ♯Expansions of ActiveSearchList vs (v_s, v_t)-hops (Proposed Method, Naive Method, ARSC($r = 3$) ($|V| = 960, d = 2$)

イン数の平均値を示す．提案手法の特徴として，探査実行処理では境界で経路の延伸操作が行われるため，各境界が持つ経路スカイライン数に比例して展開数が増加する．同図の値は属性数 d の値に応じて増加しているが，分割数を増やしても領域あたりの境界数が増加せず，この値の増加は小さく留まっていることが確認できる．

・全対経路索引と距離索引のファイルサイズ比較：図 9 に全対経路索引のデータを示す．p 値が小さくなると，部分領域 G_i のサイズが大きくなる．これは 2 点を結ぶ全対経路数と RS の各経路長が大きくなることによる．また次元数 d が大きくなると $|RS|$ が大きくなるためファイルサイズも増加する．

一方，参照点埋込み法による距離索引のファイルサイズは，参照点数 r と次元数 d に比例して増加していく．参照点数が 3 で次元数 2 の時，距離索引のファイルサイズは 145KB となり，原地図データの 1.4 倍程度である．

6.3 探査実行処理に関する評価

アクティブ探査リストの展開数，訪問ノード数，探査停止基準により刈込まれた経路数を比較する．

実験では，(v_s, v_t) 間のホップ数に応じたデータを収集するため，(v_s, v_t) をランダムに 70 回指定して測定した．得られた結果から，データの偏りを避けるため，上位・下位部分を 1 割程度ずつ削除した 50 回の平均値を用いている．

・アクティブ探査リストの展開数：図 10 に (v_s, v_t) 間のホップ数に対するアクティブ探査リストの展開数を示す．(v_s, v_t) 間のホップ数が大きくなると展開数が単調に増加している．また，母点選択確率 p の影響を受けて，p 値が小さくなるほど展開数が少ない．

・訪問ノード数：図 11 に探査実行処理で，v_s から探査して，v_t に至る経路スカイラインがすべて得られるまでの間に，一度でも訪問したノード数を示す．提案手法は，他手法と比較して値が小さく，p 値が小さくなるほど訪問ノード数が少なくなる．

・経路刈込み数：図 12 に経路の刈込み数を示す．ここでは 5.2.3 節の探査停止基準に従って，(i) 各ノードでの刈込み，(ii) 目的地点 v_t での刈込み，(iii) アクティブ探査リストの刈込みの合計数を示している．提案手法では，p が小さくなると，他手法と比較して，図 11 での訪問ノード数と比較して，刈込み経路数が大きくなっている．これは 1 つの訪問ノードで多数の刈込みが発生していることを示す．

・探査実行時間：図 13 に v_s, v_t 間ホップ数と探索実行時間の関係を示す．参照点埋込み法ではホップ数が増えるに従い，実行時間が大きくなるのに対して，提案手法ではホップ数によらず効率的に実行できることを示している．

・アクティブ探査リストサイズの動き：図 14 に p 値を変化させた提案手法，ナイーブ手法，ARSC でのアクティブ探査経路リスト数の動きを示す．v_s, v_t は 40 ホップの組合せより抽出した．提案手法では，p が小さくなるほど，概ね最大サイズが小さくなり m 更にアクティブ探査リストが空になって探査を停止するステップ数が小さくなっていることが確認できる．

Note：2 点間の経路スカイラインには，最短経路と異なり複数経路が含まれる．このため v_s, v_t 間の経路として多数の領域を跨ぐ経路に対して経路スカイラインを計算する場合，境界間経路数が増加してアクティブ探査リスト長が大きくなり，探査実行時間に影響を与える．このような場合には，より多くの探査領域を刈込む方法を用いることで探査実行時間を短くする必要がある．

Note：ここでは紙面の関係で割愛しているが，エッジコストの有効桁数が大きくなれば，支配されない経路数が多くなり，経路スカイラインに含まれる経路数が増大する．また，属性値間の相関関係の影響や高次元コストの場合等の定量的な評価も必要である．これらは別稿に譲る．

7. おわりに

本稿では，経路索引を用いた経路スカイライン問合せ手法を提案した．経路スカイライン問題では，探査空間の効率的な刈込みを促進するグラフの展開法が重要である．提案手法ではグラフ分割を用いているが，多次元データに対するアプローチの応用等も考えられる．今後の課題として，経路探査の効率化，経路スカイラインの特徴を用いた応用システム開発等が挙げられる．

【文献】

[1] S. Börzsönyil, D. Kossmann, K. Stocker, *The Skyline Operator*, IEEE ICDE, pp.421–430 (2001)
[2] D. Kossmann, F. Ramsak, S. Rost, *Shooting stars in the sky: An online algorithm for skyline queries*, VLDB, pp.275–286 (2002)

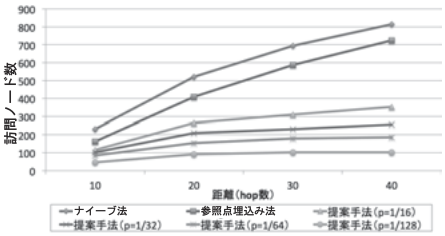

図 11 訪問ノード数と (v_s, v_t) 間ホップ数 (提案手法, ナイーブ手法,ARSC($r = 3$) ($|V| = 960, d = 2$)

Fig. 11 ♯VisitedNodes vs (v_s, v_t)-hops (Proposed Method, Naive Method, ARSC($r = 3$) ($|V| = 960, d = 2$)

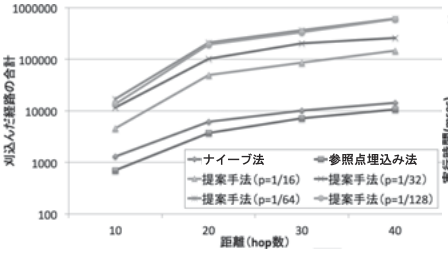

図 12 刈込み経路数と (v_s, v_t) 間 (提案手法, ナイーブ手法,ARSC($r = 3$) ($|V| = 960, d = 2$)

Fig. 12 ♯PrunedPaths vs (v_s, v_t)-hops (Proposed Method, Naive Method, ARSC($r = 3$) ($|V| = 960, d = 2$)

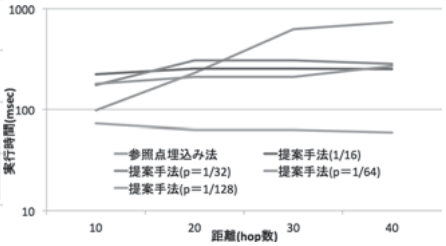

図 13 探査実行時間 (msec) と (v_s, v_t) 間ホップ数 (提案手法,ARSC($r = 3$) ($|V| = 960, d = 2$)

Fig. 13 SearchTime vs (v_s, v_t)-hops (Proposed Method, ARSC($r = 3$) ($|V| = 960, d = 2$)

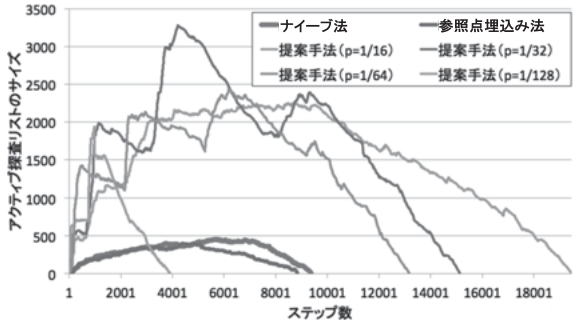

図 14 アクティブ探査経路リストのサイズの動き vs ステップ数 (提案手法, ナイーブ手法, ARSC($r = 3$) ($|V| = 960, d = 2$)

Fig. 14 Size of ActiveSearchList vs Steps (Proposed Method, Naive Method, ARSC($r = 3$) ($|V| = 960, d = 2$)

[3] P. Wu, C. Zhang, Y. Feng, B. Y. Zhao, D. Agrawal, A. E. Abbadi, *Parallelizing Skyline Queries for Scalable Distribution*, EDBT, pp.112–130 (2006)
[4] S. Wang, B. C. Ooi, A. K. H. Tung, L. Xu, *Efficient Skyline Query Processing on Peer-to-Peer Networks*, IEEE ICDE, pp.1126–1135 (2007)
[5] A. Vlachou, C. Doulkeridis, Y. Kotidis, *Angle-based Space Partitioning for Efficient Parallel Skyline Computation*, ACM SIGMOD, pp.227–238 (2008)
[6] H. Kohler, J. Yang, X. Zhou, *Efficient Parallel Skyline Processing using Hyperplane Projections*, ACM SIGMOD, pp.85–96 (2011)
[7] H.-P. Kriegel, M. Renz, M. Schubert, *Route Skyline Queries: A Multi-Preference Path Planning Approach*, IEEE ICDE, pp.261–272 (2010)
[8] F. Graf, H.-P. Kriegel, M. Renz, M. Schubert, *PAROS: Pareto Optimal Route Selection*, ACM SIGMOD, pp.1199–1202 (2010)
[9] F. Graf, M. Renz, H.-P. Kriegel, M. Schubert, *MARiO: Multi-Attribute Routing in Open Street Map*, LNCS, Vol.6849, pp.486–490 (2011)
[10] Y.Yang, J.X.Yu, H.Gao, J.Li, *Finding the Optimal Path over Multi-Cost Graphs*, ACM CIKM, pp.2124–2128 (2012)
[11] Y.Tian, K.C.K.Lee, W-C.Lee, *Finding Skyline Paths in Road Networks*, ACM SIGSPATIAL, pp.444–447 (2010)
[12] K.Deng, X.Zhou, H.T.Shen, *Multi-source Skyline Query Processing in Road Networks*, IEEE ICDE, pp.796–805 (2007)
[13] K.Mouratidis, Yimin Lin, Man Lung Yiu, *Preference Queries in Large Multi-cost Transportation Networks*, IEEE ICDE, pp.533–544 (2010)
[14] M. Sharifzadeh, C. Shahabi, *The Spatial Skyline Queries*, VLDB, pp.751–762 (2006)
[15] H.Samet, *Foundations of Multidimensional and Metric Data Structures*, Morgan Kaufmann (2006).
[16] 蒲原智也，上島紳一，道路網応用のための空間索引木の提案と最短経路探査への応用，IPSJ TOD,2(2),pp.10–28 (2009)
[17] M Erwig.: *The Graph Voronoi Diagram with Applications*, Networks, Vol.36, pp.156–163 (2000)
[18] A. Okabe, B. Boots, K. Sugihara, S.N. Chin.: *Spatial Tessellations:Concepts and Applications of Voronoi Diagrams*, John Wiley (2000)
[19] 国土地理院：数値地図 (空間データ基盤), http://sdf.gsi.go.jp/

蒲原 智也 Tomoya KAMBARA
関西大学先端科学技術推進機構非常勤研究員．2012 関西大学大学院総合情報学研究科博士後期課程修了，博士 (情報学)．データ工学, 特に空間情報管理に関する研究に従事.

小池 実 Minoru KOIKE
関西大学大学院総合情報学研究科博士前期課程在学中．データ工学, 特に経路スカイラインの研究に従事.

壷内 貴弘 Takahiro TSUBOUCHI
関西大学総合情報学部在学中．データ工学の研究に従事.

上島 紳一 Shinichi UESHIMA
関西大学総合情報学部教授．1978 京都大学工学部卒，1983 同大学院工学研究科単位取得退学，工学博士．マルチメディア情報システム，データ工学の研究に従事．情報処理学会，IEEE，ACM 等の会員.