アドホックネットワークにおける多次元データ Top-k 検索手法

天方 大地[1,a)]　佐々木 勇和[1,b)]　原 隆浩[1,c)]　西尾 章治郎[1,d)]

概要：近年，ユーザが指定する検索条件に基づいてデータのスコアを決定し，上位 k 個のスコアをもつデータを検索する Top-k 検索への関心が高まっている．また，無線通信端末のみで構成される一時的なネットワークであるアドホックネットワークの普及が進んでおり，本稿では，アドホックネットワーク環境における Top-k 検索処理を考える．検索対象となるデータは，一般的に多次元の属性値をもっており，ユーザの検索条件や検索範囲によって，上位 k 個となるデータが異なる．一方，無線通信を行うアドホックネットワークでは，不要なトラヒックを削減することが重要であるため，効率的な検索方法が必要となる．本稿では，ユーザごとに異なる条件が指定された場合においても，低トラヒック，および低遅延で上位 k 個のデータを取得する手法，ClusTo を提案する．ClusTo では，クラスタを用いてクエリのルーティングを行い，ユーザが指定する検索範囲に効率的に検索クエリを送信する．さらに，検索結果の取得に不要な端末へのクエリ送信を抑制しつつ，正解集合に含まれるデータを効率的に取得する．シミュレーション実験の結果から，ClusTo は，低トラヒック，および低遅延で上位 k 個のデータを取得できることを確認した．

1. はじめに

Top-k 検索は，ユーザが指定する検索条件とスコアリング関数に基づき，最も関連する上位 k 個のデータを検索する．複数の次元の属性値をもつデータから，ユーザの趣向に最も近いデータを検索できる Top-k 検索は，多くのアプリケーションで利用され，様々な分野において研究が行われている．例えば，ウェブや P2P ネットワーク [7] では，ユーザの興味に合うドキュメントの検索や，センサネットワーク [4] では，環境のモニタリングなどがある．無線通信端末のみで構成される一時的なネットワークであるアドホックネットワーク [1], [3], [6] においても，Top-k 検索は有効なデータ検索手段である．例えば，災害等が発生した場合，インターネット等の既存の通信インフラが利用できない可能性がある．そのような場合における救急活動では，被災現場に派遣された救急隊員のもつ無線通信端末でアドホックネットワークを構築することが有効である．救急隊員は自身の担当する現場での被災情報（例えば被災者の年齢や負傷状況など）を端末に入力し，Top-k 検索によって自身の周囲に存在する上位 k 個の被災情報を取得することにより，円滑に医療対応を行うことが可能である．

[1] 大阪大学大学院情報科学研究科マルチメディア工学専攻
Department of Multimedia Engineering, Graduate School of Information Science and Technology, Osaka University
[a)] amagata.daichi@ist.osaka-u.ac.jp
[b)] sasaki.yuya@ist.osaka-u.ac.jp
[c)] hara@ist.osaka-u.ac.jp
[d)] nishio@ist.osaka-u.ac.jp

アドホックネットワークでは無線通信によりネットワークを構成するため，トラヒックが増加すると，帯域の圧迫によるパケットロスや，端末の消費電力の増大といった問題が生じる．そのため，Top-k 検索を行う際，上位 k 個に含まれないデータの返信やメッセージ送信端末数の削減による，不要なトラヒックの抑制が重要である．理想的には，検索結果の取得に必要な端末のみで Top-k 検索を行うことが望ましい．一方，検索結果に含まれるデータは，ユーザが指定する検索条件，検索範囲，および要求データ数 k によって異なる．ここで，単純な検索方法としては，次の 2 つが考えられる．1 つ目は，検索クエリを検索範囲内の全ての端末にフラッディングにより送信し，全端末から上位 k 個のデータを取得することである．しかし，この方法では，検索結果に含まれるデータを持たない多くの端末に検索クエリが転送される問題がある．もう一方は，検索条件ごとに効率的なメッセージの送信経路を作成することである．しかし，Top-k 検索を行うユーザが，それぞれ異なる検索条件を指定する場合，それらの異なる条件の個々に対して，経路構築，および管理が必要となり，そのオーバヘッドが非常に大きくなる．また，いずれかの検索結果に含まれる全てのデータを全端末に送信することにより，ローカルに上位 k 個のデータを取得することも考えられるが，送信すべきデータ数が非常に多くなること，およびデータの更新処理に大きなトラヒックが生じることが問題となる．このように，多次元データを対象とする Top-k 検索の効率化は容易ではないが，低トラヒック，および低遅延の達成

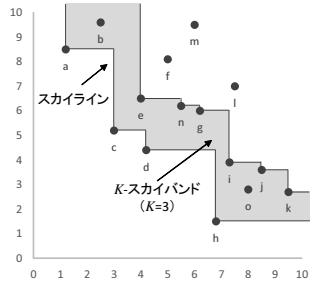


図 1 スカイラインと K-スカイバンドの例

は重要である．

そこで，本稿では，アドホックネットワークにおいて，ユーザごとに異なる検索条件が指定された場合においても，低トラヒック，および低遅延で上位 k 個のデータを取得する手法である ClusTo[*1]を提案する．ClusTo は，スカイライン [2] を用いたクラスタリングにより，多くの検索結果に含まれることが予想されるデータを持つ端末をクラスタヘッドとする．この際，隣接端末の持つデータの K-スカイバンド [5] を把握することにより，検索結果の取得に可能な限り必要な端末のみで Top-k 検索を行う．クエリ転送時には，自身のデータと隣接端末の K-スカイバンドからフィルタを作成し，不要なデータの返信を削減するために用いる．また，クラスタベースのクエリルーティングを行い，ユーザの指定する検索範囲に低トラヒックでクエリを送信する．その後，フィルタにより設定した閾値から，クエリを送信すべき端末を決定し，検索結果に含まれる可能性のあるデータのみを取得する．K-スカイバンドを用いることにより，いかなる検索条件においても，低トラヒックで Top-k 検索を行うことが可能である．シミュレーション実験の結果から，ClusTo は低トラヒック，および低遅延で Top-k 検索を処理できることを確認した．

以下では，2 章で想定環境と定義について説明し，3 章において関連研究について述べる．4 章で提案手法 ClusTo について説明し，5 章でシミュレーション実験の結果を示す．最後に 6 章で本稿のまとめと今後の課題について述べる．

2. 想定環境と定義

本章では，本稿における想定環境について説明し，Top-k 検索などの定義について述べる．

2.1 想定環境

2.1.1 データモデル

本稿で想定するデータ o_i は，一意の識別子 i，および m 個のメタデータをもつものとする．それぞれのメタデータ $v_i[j](1 \leq j \leq m)$ は，何らかのスコアリング関数により値が決定され，ユークリッド空間に従うものとする．さらに，$v_i[j]$ は，$[0, value_{max}]$ の範囲で正規化されているものとする．本稿では，メタデータの個数 m をデータの次元数と呼ぶ．また，データはメタデータのみからなるとは限らず，データサイズは，基本的にメタデータのサイズの合計よりも大きいものとする．また，ネットワークにデータの複製は存在しないものとする．

[*1] Cluster & K-skyband-based multi-dimensional Top-k query processing

2.1.2 ネットワークモデル

ネットワーク内には，n 台の端末が存在し，それぞれ一意の識別子 $\{N_1, N_2, \cdots, N_n\}$ が割り当てられているものとする．各端末は同一の無線規格を用いて通信を行い，全て通信半径は同じもの（r[m]）とする．さらに，GPS などの機器により自身の位置を正確に把握できるものとする．また，端末は固定であるとする．実際には，端末をもつユーザは移動することも考えられるが，本稿では，移動が起きた場合においても，ネットワークトポロジに影響を与えないほどの局所的な移動のみとし，固定とみなす．

2.2 定義

本稿における Top-k 検索では，スコアリング関数 f を用いて，メタデータからデータ o_i のスコアを決定する．具体的には，o_i のスコア（$score(o_i)$）は，$score(o_i) = f(o_i) = \sum_{j=1}^{m} w_j \times v_i[j], (\sum_{j=1}^{m} w_j = 1, w_j \geq 0)$ として計算される．ユーザは，各 w_j を入力とし，各次元に対するユーザの趣向を示す．また，f は単調線形関数であり，全ての次元において，$v_i[j] \leq v_{i'}[j]$ ならば，$f(o_i) \leq f(o_{i'})$ であることを満たす．さらに，ユーザは自身の位置から d[m] 以内に存在するデータを検索対象とする．本稿では，スコアリング関数 f，および検索範囲 d を検索条件とよぶ．

定義 1 (Top-k 検索)．Top-k 検索の検索結果に含まれるデータは，ユーザから d[m] 以内に存在する端末の持つデータのうち，スコアの最も小さい k 個のデータとする．

ここで，指定できる k の値には上限（最大値）K があるものとする．

定義 2 (ドミナント)．あるデータ $o_i, o_{i'}$ に対して，全ての次元 j において，$v_i[j] \leq v_{i'}[j]$ を満たし，かつ，少なくとも 1 つの次元 j' において，$v_i[j'] < v_{i'}[j']$ を満たすとき，o_i は $o_{i'}$ をドミナントしているという．

例えば，図 1 は，2 次元データの集合を示し，a は b をドミナントしている．

定義 3 (スカイライン)．スカイラインは，あるデータ集合において，他のどのデータにもドミナントされていないデータの集合とする．

例えば，図 1 では，a，c，d，および h がスカイラインに含まれる．

定理 1. スコアリング関数が単調線形関数であるとき，最上位のデータは必ずスカイラインに含まれる．

定義 4 (K-スカイバンド)．K-スカイバンドは，あるデータ集合において，高々 K-1 個のデータにドミナントされているデータの集合とする．

定理 2. スコアリング関数が単調線形関数であるとき，上位 K 個のデータは，必ず K-スカイバンドに含まれる．

証明. ある上位 K 個に含まれるデータ o_i が K-スカイバンドに含まれないと仮定する．このとき，少なくとも K 個のデータが o_i をドミナントしている．単調線形関数の性質から，少なくとも K 個のデータのスコアは，o_i のスコアより小さいことは自明である．そのため，o_i は上位 K

個のデータに含まれることはない．したがって，背理法により，定理 2 は成り立つ．また，スカイラインは，K-スカイバンドにける K=1 の場合の集合であるため，定理 1 は成り立つ．□

3. 関連研究

本章では，様々な分散環境での Top-k 検索における代表的な既存手法を紹介する．

文献 [4] では，無線センサネットワークにおいて，ドミナントグラフ [8] を用いた Top-k 検索手法を提案している．この手法では，ドミナントグラフを用いて，上位 k 個に含まれ得るデータを計算し，シンクにデータを返信する．シンクは受信したデータからフィルタを作成し，ネットワーク内のセンサ端末にフィルタを送信する．各センサ端末は，データの更新が起きた際，フィルタから返信すべきデータを求める．この手法では，Top-k 検索を行う端末がシンクのみである点で，アドホックネットワークと異なる．また，ユーザが指定することのない検索条件をも考慮する点で問題がある．文献 [7] では，P2P ネットワークにおいて，代表ピアを用いた Top-k 検索手法を提案している．代表ピアは，論理リンクの存在するピアの持つ K-スカイバンドを複製として保持する．さらに，隣接する代表ピアの持つデータのスカイラインを管理し，ある Top-k 検索での閾値とスカイラインから，クエリを送信すべき代表ピアを決定する．この手法は，データの複製を想定している点，およびデータの検索範囲がネットワーク全体である点で本稿の想定と異なる．また，パケットロスを想定していない点でアドホックネットワークへの適応は困難である．

文献 [1], [6] では，モバイルアドホックネットワークにおける Top-k 検索手法を提案している．文献 [6] では，正確な閾値を設定するため，2 フェーズによる Top-k 検索を行う．1 フェーズ目でネットワーク内のデータの k 番目のスコアを取得し，2 フェーズ目で閾値以下のスコアをもつデータを返信する．しかし，クエリをフラッディングにより転送しているため，検索結果の取得に必要がない端末以外への無駄な転送が多い．文献 [1] では，経路表を作成し，検索結果の取得に必要な隣接端末にクエリを送信する．しかし，検索条件ごとに経路表を作成する必要があるため，検索条件の種類が多い場合，オーバヘッドが非常に大きくなる問題がある．一方，ClusTo は，いかなる検索条件が指定された場合においても，検索範囲内に存在する全ての端末にクエリを送信することなく，低遅延で Top-k 検索を行うことができる．さらに，文献 [1] と異なり，クラスタリングは 1 度のみの実行であり，検索条件が指定されるごとにオーバヘッドが発生することはない．

4. 提案手法

本章では，提案手法 ClusTo における，クラスタリング方法，および Top-k 検索のメッセージ処理方法について説明する．

4.1 クラスタの構築

本節では，いかなる検索条件においても少ない端末数で Top-k 検索を行うことを目的とした 1 ホップクラスタリングの方法について説明する．

検索結果に含まれるデータを持つ端末へクエリを送信するために，単純にフラッディングを用いることが考えられる [6]．しかし，フラッディングによる転送では，検索結果の取得に必要がない多くの端末にクエリが送信されてしまい，無駄なトラヒックが発生してしまう．ClusTo は，クラスタを用いてクエリをルーティングすることにより，この問題を解決する．クラスタを用いたルーティングでは，他のクラスタに属する端末と隣接する端末（ゲートウェイノード）を介し，クラスタヘッド間でクエリの送信が行われる．このとき，クラスタヘッドの持つデータが，クラスタ内の端末の持つデータをドミナントしている場合，これらの端末にクエリを送信せずに Top-k 検索を行える．そのため，より少ない端末数での Top-k 検索の実現を目的として，多くのデータをドミナントすると予想されるデータを持つ端末をクラスタヘッドとする．具体的には，各端末は自身の持つデータにおけるスカイラインを計算し，以下の式からクラスタヘッドとなるためのタイマを設定する．

$$Timer = \frac{\alpha \cdot SkyAvgDist + (1-\alpha) \cdot SkyMinDist}{\beta}. \quad (1)$$

$$SkyAvgDist = \frac{\sum_{i=1}^{|skyline|}(\sqrt{\sum_{j=1}^{m} v_i[j]^2})}{|skyline|}. \quad (2)$$

$$SkyMinDist = \min_{1 \le i \le |skyline|}(\sqrt{\sum_{j=1}^{m} v_i[j]^2}). \quad (3)$$

式 (1) における $\alpha \in [0,1]$，および $\beta(>0)$ はシステムパラメータであり，β はタイマの値が不要に大きくなることを防ぐために用いる．式 (2)，および (3) における $|skyline|$ は各端末がもつデータにおけるスカイラインのデータ数を表す．$SkyAvgDist$，および $SkyMinDist$ は，自身の持つスカイラインとなるデータ（メタデータ）と原点（0, 0, ...,0）との平均距離，および最短距離をそれぞれ示す．定理 1 から，スカイラインは何らかの検索条件に対して最上位となるデータの集合であるため，これらの値が小さいほど，多くの検索に含まれやすいデータをもっていることを示す．

タイマは値が小さいものから発火され，タイマが発火した端末からクラスタヘッドとなる．クラスタヘッドは，クラスタ構築メッセージをブロードキャストする．このメッセージには，クラスタヘッドの識別子，送信端末識別子，送信端末の位置情報，送信端末の K-スカイバンドに含まれるデータのメタデータ，および識別子が含まれる．受信した端末は，それがクラスタヘッドから送信されたもの（クラスタヘッドの識別子と送信端末識別子が同じ）である場合，タイマをキャンセルする．また，クラスタ構築メッ

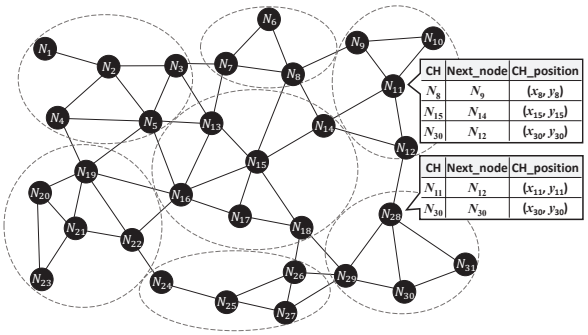


図 2 クラスタと経路表の例

セージを受信した全ての端末は，以下の処理を行う．(i) 経路表を作成する．経路表のそれぞれのエントリにはクラスタヘッド（CH）の識別子，その位置情報（CH_position），およびそのクラスタヘッドにクエリを送信するための隣接端末（next_node）の識別子[*2]が格納される（図 2）．さらに，(ii) 隣接端末の位置情報，および K-スカイバンドを記録する．データそのものではなく，K-スカイバンドに含まれるデータのメタデータを送信するのは，データのスコアを計算するためには，これらの情報のみ把握すればよく，データそのものを送信するとオーバヘッドが大きくなるためである．定理 2 から，上位 $k(\leq K)$ 個のデータは，検索範囲に存在する端末の K-スカイバンド内に存在するため，隣接端末の K-スカイバンドを把握することにより，いかなる検索条件が指定された場合においても，効果的なフィルタを作成できる（4.2 節）．さらに，これまでにクラスタ構築メッセージを送信していない場合，(iii) クラスタ構築メッセージをブロードキャストする．このとき，クラスタヘッドの識別子は，受信したメッセージのクラスタヘッドの識別子とする．

一定時間経過後，クラスタヘッド以外の端末は，自身が把握する最短距離のクラスタヘッドに，メンバ通知メッセージを送信する．このメッセージには，送信端末識別子，および自身が把握しているクラスタヘッドの識別子と，その位置情報が含まれている．これにより，各クラスタヘッドは，隣接するクラスタのクラスタヘッド，next_node，および自身のクラスタに所属するメンバ端末を把握できる．このとき，next_node は送信先となるクラスタヘッドに最も近い端末とする．これは，送信先クラスタヘッドに最も近い端末は，そのクラスタヘッドと隣接している可能性が高いため，送信先クラスタのゲートウェイノードに検索クエリを転送する必要がなく，検索クエリのホップ数を削減できるためである．

以上の動作により，効率的な Top-k 検索のためのクラスタを構築し，検索クエリを送信する端末数を抑制する．また，各端末は，自身の隣接端末の位置情報，および K-スカイバンドを把握できる．さらに，データの更新が起きた場合においても，各端末は隣接端末の情報のみを保持しているため，更新された K-スカイバンドを隣接端末にブロードキャストするだけでよい．

[*2] クラスタヘッドと隣接している場合，next_node=CH となる

4.2 フィルタの作成

ClusTo では，経路表を用いてクエリルーティングを行い，ユーザが指定する検索範囲まで検索クエリを転送する．その後，各クラスタにおいて検索クエリが送信されていない端末に検索クエリを送信し，検索結果に含まれるデータを取得する．ここで，無駄なトラヒックを削減するため，検索結果の取得に貢献しない端末へは検索クエリを送信すべきではない．そのために，各端末が，検索範囲内の上位 k 番目のデータのスコア（閾値）を見積もることにより，検索結果に含まれるデータを持たない隣接端末への不要な検索クエリの転送を抑制することが有効である．これを実現する直感的な手法として，各端末が検索クエリにフィルタとして，自身が把握する上位 k 個のデータのスコアとその識別子を添付し，それを受信した端末が，この情報と自身が持つデータのスコアから，閾値を見積もることが考えられる．しかし，k が大きい場合，フィルタの送信にかかるオーバヘッドが大きくなる問題がある．

そこで，ClusTo では，フィルタによるオーバヘッドを固定かつ小さく抑えるため，k 個分のデータではなく，各端末が把握する，上位 1 番目，$\lfloor k/2 \rfloor + 1$ 番目，および k 番目となる 3 つのデータのスコアとその識別子の組合せ（それぞれ lb，me，および ub）をフィルタとする．さらに，閾値を正確な値に近づけるため，各端末は，受信した検索クエリに含まれるフィルタ，および自身と自身の隣接端末が持つデータのスコア情報からフィルタを更新する．

フィルタ作成のアルゴリズムを，アルゴリズム 1 に示す．各端末は，基本的に，受信したフィルタと自身と自身の隣接端末が持つデータにおける上位 k 個のデータのスコア情報から，フィルタを作成する[*3]．ここで，検索クエリに添付するフィルタは，3 つのスコアと識別子の組合せであるため，$k > 3$ の場合，クエリ発行端末からクエリの送信元端末までの経路に含まれる端末，およびそれらの隣接端末（以後，上流端末）の持つデータにおいて，上位 k 個となったデータのスコアと識別子の全ては把握できない．このとき，何らかの方法により，把握できなかったデータのスコアを推定する必要があるが，閾値が正確な値よりも小さくなってしまうと，検索範囲内の上位 k 個に含まれるデータを持つ端末にクエリを送信しない可能性がある．一方，検索結果に含まれるデータを持つ端末へ確実にクエリを転送するために，把握できなかったデータのスコアを大きめ（セーフティ）に推定しすぎると，不要な検索クエリの転送，および検索結果に含まれないデータの返信が増加してしまう．そのため，検索結果に含まれるデータを持つ端末へのクエリ転送を保証しつつ，閾値が不要に大きくならないようにすることが重要である．

そこで，ClusTo では，上流端末の持つデータにおける上位 k 個のデータのうち，フィルタとして添付されなかったデータ[*4]のスコア（および識別子）を，各端末が把握で

[*3] 4.1 節の処理により，隣接端末の持つデータのスコアを計算できる．

[*4] クエリ送信元端末において，lb から me，および me から ub の

Algorithm 1 フィルタの作成

Input: lb, me, ub
Output: lb', me', ub'
1: /* Node, N_p, makes a filter from a query sent by N_q */
2: *local top-k data*← ∅
3: $lm\_data \leftarrow mu\_data \leftarrow \emptyset$
4: $lm\_data_{com} \leftarrow mu\_data_{com} \leftarrow \emptyset$
5: **for** $\forall N_{q'} \in |N_p.neighboring\ node|$ **do**
6: **if** $N_{q'}$, is within query range **then**
7: **if** $Dist(N_q, N_{q'}) \leq r$ **then**
8: $lm\_data_{com} \leftarrow lm\_data_{com} \cup \{\forall\{score(o_j), j\} \mid o_j \in N_{q'}$'s K-skyband, $score(o_j) \in [lb.score, me.score]\}$
9: $mu\_data_{com} \leftarrow mu\_data_{com} \cup \{\forall\{score(o_j), j\} \mid o_j \in N_{q'}$'s K-skyband, $score(o_j) \in (me.score, ub.score]\}$
10: **end if**
11: **end if**
12: **end for**
13: Sort lm_data_{com} and mu_data_{com} by ascending order
14: $lm\_data \leftarrow lb \cup lm\_data_{com}$
15: **if** $|lm\_data| < \lfloor k/2 \rfloor + 1$ **then**
16: **for** i=0 to $\lfloor k/2 \rfloor + 1 - |lm\_data|$ **do**
17: $lm\_data \leftarrow lm\_data \cup \{me.score, -(i+1)\}$
18: **end for**
19: **end if**
20: $mu\_data \leftarrow mu\_data_{com}$
21: **if** $|mu\_data| < k - \lfloor k/2 \rfloor - 1$ **then**
22: **for** i=0 to $k - \lfloor k/2 \rfloor - 1 - |mu\_data|$ **do**
23: $mu\_data \leftarrow mu\_data \cup \{ub.score, -(i+k)\}$
24: **end for**
25: **end if**
26: Calculate *local top-k data* from its own data and neighbors' K-skyband within query range
27: *local top-k data* ← *local top-k data* ∪ *lm_data* ∪ *me* ∪ *mu_data* ∪ *ub*
28: Sort *local top-k data* by ascending order
29: $lb' \leftarrow$ *local top-k data*[0]
30: $me' \leftarrow$ *local top-k data*[$\lfloor k/2 \rfloor$]
31: $j \leftarrow \lfloor k/2 \rfloor$
32: **while** $me'.id < 0$ **do**
33: $j \leftarrow j + 1$
34: $me' \leftarrow$ *local top-k data*[j]
35: **end while**
36: $ub' \leftarrow$ *local top-k data*[$k-1$]
37: $j \leftarrow k - 1$
38: **while** $ub'.id < 0$ **do**
39: $j \leftarrow j + 1$
40: $ub' \leftarrow$ *local top-k data*[j]
41: **end while**

きるデータのスコア情報を基に推定する．具体的には，各端末は，まず，クエリ送信元端末と自身との共通の通信範囲内に存在する端末の持つデータのスコアを計算する．このスコアと受信したフィルタから，クエリ送信元端末が把握している上位 k 個のデータを推定する（5〜13 行）．また，共通の通信範囲内の端末が持つデータのうち，スコアがフィルタの範囲内となるものの個数が，lb から me，および me から ub に存在するデータの個数に満たなかった場合，不足分を，[lb, me] 間は me，[me, ub] 間は ub と見なす（15〜19 行，および 21〜25 行）．このように，把握していないデータのスコアを [lb, me] 間，および [me, ub] 間における最大スコアとして補うことにより，クエリ送信元端末が把握している上位 k 個のデータのスコアを小さく見積もることはない．その結果，検索結果に含まれるデータを持つ端末へのクエリ転送を保証できる．その後，受信したフィルタ，補完した { スコア，識別子 }，および自身と隣接端末が持つデータから上位 k 個となるものを計算し，自身のフィルタを作成する（26〜41 行）．

間には，それぞれ $(\lfloor k/2 \rfloor - 1)$ 個，および $(k - \lfloor k/2 \rfloor - 2)$ 個のデータが存在する．

4.3 Top-k 検索

本節では，ClusTo における Top-k 検索のメッセージ処理方法について説明する．

ClusTo では，2 種類の検索クエリ，クラスタ間クエリ（*CtoC-query*），およびデータ収集クエリ（*data_acquisition-query*）を用いる．クラスタ間クエリにより，ユーザが指定した検索範囲を保証し，その後，データ収集クエリにより，上位 k 個に含まれる可能性があるデータを取得する．

まず，クラスタ間クエリ qc_i は，qc_i=<org, org_position, sender_id, k, d, w, F, AT, CL> で表される．i は検索クエリの識別子，org は検索クエリ発行端末の識別子，org_position は，検索クエリ発行端末の位置情報，および sender_id は検索クエリ送信端末の識別子である．検索クエリ発行端末 N_{org} は，要求データ数 k，検索距離 d，および w=$\{w_1, w_2, \cdots, w_m\}$ を指定する．また，データのスコアを計算し，フィルタ F を作成する．その後，自身の経路表における全てのクラスタヘッドの識別子（CH_j）とその next_node を，AT=*AddressTuple*=$\{\{CH_0, next\_node_0\}, \{CH_1, next\_node_1\}, \cdots\}$ に格納する．このとき，next_node が検索範囲内に含まれない場合は，AT には格納しない．また，CL=*ClusterList* は，検索クエリが送信されたクラスタヘッドの識別子を格納するリストであり，N_{org} が送信する検索クエリでは ∅ である（N_{org} がクラスタヘッドならば，CL={org}）．検索クエリを作成後，N_{org} は，AT に含まれる next_node にクエリを送信する．これを受信した端末 N_p におけるメッセージ処理をアルゴリズム 2，およびアルゴリズム 3 に示す．また，検索クエリを送信後，クエリ応答を受信した場合の処理をアルゴリズム 4 に示す．N_p は，すでに検索クエリが送信されているクラスタヘッドを除いたクラスタヘッドにクエリを送信する（アルゴリズム 2，6〜15 行）．このとき，クラスタヘッドに送信するための next_node が検索範囲外であるならば，そのクラスタヘッドにはクエリを送信しない（アルゴリズム 2，10〜15 行）．また，AT が ∅ となった場合は自身を葉端末とし，閾値以下のスコアをもつデータを持っている端末にデータ収集クエリを送信する（アルゴリズム 3）．

データ収集クエリ qd_i は，qd_i=<org, org_position, sender_id, k, d, w, F, A> で表される．A=*Address* は，データ収集クエリの送信先端末の識別子を格納するリストである．自身の AT に含まれる全ての next_node からクエリ応答を受信したクラスタヘッドも同様にデータ収集クエリを送信する．検索クエリを受信した各端末は，フィルタから閾値を設定する．さらに，一度，葉端末まで検索クエリを転送後，クエリ応答により閾値を更新する（アルゴリズム 4，2〜5 行）．その後，データ収集クエリを送信することにより，可能な限り閾値を小さくし，検索結果の取得に不要な端末への検索クエリ送信を抑制できる．ここで，検索範囲の境界付近では，クラスタヘッドは検索範囲外であるが，そのクラスタに所属する端末は検索範囲内に存在

Algorithm 2 検索クエリ q_q の処理

```
 1: if N_p receives q_q for the first time then
 2:    parent ← q_q.sedner_id
 3:    Set F by Algorithm 1
 4:    threshold ← F[2].score //ub.score
 5:    if qc_q then
 6:       CL ← q_q.CL
 7:       for i=0 to |q_q.AT|-1 do
 8:          CL ← CL ∪ q_q.AT.CH[i]
 9:       end for
10:       AT ← ∅
11:       for ∀t ← RoutingTable.entry do
12:          if t.CH is not included in CL and
             t.next_node is within d from N_org then
13:             AT ← AT ∪ {t.CH, t.next_node}
14:          end if
15:       end for
16:       if AT≠ ∅ then
17:          qc_q ← <org, org_position, p, k, d, w, F, AT, CL>
18:          Send qc_q to nodes included in AT as next_node
19:       else
20:          LeafFlag ← 1
21:          Perform Algorithm 3
22:       end if
23:    end if
24:    if qd_q then
25:       Reply data whose score is not more than threshold to
          parent
26:    end if
27: end if
```

Algorithm 3 葉端末，およびクラスタヘッドの処理

```
 1: A ← ∅
 2: if LeafFlag = 1 then
 3:    Put in all neighbors' identifiers, N_r which fulfill all the
       following conditions to A
       · N_r is within d from N_org
       · N_r has data whose score is not more than threshold
       · N_r's nearest cluster head is not within d from N_org
 4: end if
 5: if N_p is a cluster head then
 6:    Put in all cluster members' identifiers, N_r which fulfill all
       the following conditions to A
       · N_r is within d from N_org
       · N_r has data whose score is not more than threshold
       · N_r is not parent
 7: end if
 8: if A ≠ ∅ then
 9:    qd_q ← <org, org_position, p, k, d, w, F, A>
10:    Send qd_q to nodes included in A
11: else
12:    Reply data whose score is not less than threshold to parent
13: end if
```

Algorithm 4 クエリ応答の処理

```
 1: /* N_p receives r_i */
 2: if threshold > r_i.threshold then
 3:    threshold ← r_i.threshold
 4:    D ← D ∪ r_i.D
 5: end if
 6: if N_p receives reply from all next_nodes inlcluded in AT for
    the fist time then
 7:    if N_p is a cluster head then
 8:       Perform Algorithm 3
 9:    else
10:       D′ ← {∀data∈ D|data.score ≤ threshold}
11:       r_i ← <p, D′, threshold>
12:       Send r_i to parent
13:    end if
14: else if N_p receives reply from all next_nodes inlcluded in A
    then
15:    D′ ← {∀ data∈ D|data.score ≤ threshold}
16:    r_i ← <p, D′, threshold>
17:    Send r_i to parent
18: end if
```

する場合がある．その端末は，クラスタヘッドによるデータ収集クエリの対象とならないため，上位 k 個に含まれるデータを持っている場合でも，クラスタヘッドからデータ収集クエリが送信されない．そのため，葉端末が，上記の条件を満たす端末にデータ収集クエリを送信する必要がある．このとき，葉端末におけるデータ収集クエリを送信する条件の 3 つ目（アルゴリズム 3，3 行）により，検索範囲内の端末へのクエリ送信を保証する．

クエリ応答 r_i は，r_i=<sender_id, D, threshold> で表される．sender_id はクエリ応答の送信端末識別子である．D には，閾値（threshold）以下のスコアをもつデータが含まれる．データ収集クエリを送信する必要のない葉端末（アルゴリズム 3，11〜13 行），およびデータ収集クエリを受信した端末は，閾値以下のスコアをもつデータを親端末に返信する．

検索クエリ，およびクエリ応答を受信した端末は，受信確認のために ACK を送信元端末に返信する．検索クエリ，またはクエリ応答の送信端末は，一定時間内に ACK を受信できなかった場合，再送を行う．

以上の動作により，ClusTo は，いかなる検索条件が指定された場合においても，検索クエリを送信する端末数を抑制しつつ，検索結果に含まれるデータを取得できる．また，4.1 節において，検索結果に含まれることが多いと考えられるデータを持つ端末をクラスタヘッドとしている．ClusTo では，クラスタヘッドにクエリを送信していくため，クラスタヘッドがクエリを受信した際，フィルタを更新し，閾値を小さく設定できるという利点がある．これにより，データ収集クエリの送信先端末数を抑制でき，不要なトラヒックを削減できる．

5. シミュレーション評価

本章では，提案手法の性能評価のために行ったシミュレーション実験の結果を示す．本実験では，ネットワークシミュレータ Qualnet6.1[*5]を用いた．

5.1 シミュレーション環境

800[m]×800[m] の 2 次元平面状の領域に n 台の端末が存在する．各端末は，IEEE802.11b を使用し，伝送速度 11[Mbps]，通信伝搬距離が 100[m] 程度となる送信電力でデータを送信する．各端末は，それぞれ 128[Byte] のサイズの m 次元データを 50 個持つものとした．各次元のメタデータの値は [0, 250] の範囲の整数とした．本実験では，メタデータの値の分布として，一様，相関，非相関を用いた．一様は，[0, 250] の範囲内で一様に値を発生させた．相関は，1 つのデータの各次元の値が似た値となるように作成した．非相関は，1 つのデータのある次元の値が小さい場合は，他のある次元の値は大きくなる（逆も同様）ように作成した．また，シミュレーション中に 30%のデータが更新されるものとした．

Top-k 検索手法として，ClusTo，2 フェーズ検索手法 [6]，および単純手法を用いた．単純手法は，クエリをフラッディングにより転送し，クエリには自身が把握する上位 k

[*5] Scalable Network Technologoes: Creaters of QualNet Network Simulator Software,
<URL: http://www.scalable-networks.com/>.

個のデータのスコアと識別子の組合せを添付する．クエリを受信した端末において，上位 k 個となるデータが更新される場合，この組合せも更新される．クエリ応答では，k 番目のスコアを閾値に設定し，閾値以下のスコアをもつデータを返信する．

本実験では，K を 15 とし，クエリ応答の再送回数を 10 回として，再送回数内でのデータの取得精度を調べた．スコアリング関数において，ユーザが入力する各 $w_j(1 \leq j \leq m)$ はランダムに決定するものとした．また，検索範囲 d を $d = 50 \cdot j$[m] とし，j は，[2, 8] の範囲の整数であり，平均 5，分散 20 の正規分布に従って決定される．各端末は [60, 600][sec] の範囲で決定したランダムな間隔で Top-k 検索を実行するものとした．以上のシミュレーション環境において，各端末の位置をランダムに決定し，900[sec] 経過させた時の以下の評価値を調べた．

- 取得精度 : 順位付き検索結果の性能を測る MAP(Mean Average Precision) の値を取得精度とする．MAP は，各クエリの平均精度 AP(Average Precision) を平均化したものである．AP および MAP は以下の式で求める．

$$AP_i = \frac{1}{k} \sum_{j=1}^{k} \frac{x}{j} \cdot e \tag{4}$$

$$MAP = \frac{1}{querynum} \sum_{i=1}^{querynum} AP_i \tag{5}$$

  AP_i は i 番目のクエリの平均精度である．x は，取得した解の上位 j 個のうち正解集合の j 位以内である解の個数である．$querynum$ はクエリの発行数を示す．また，e は以下のように定義される．

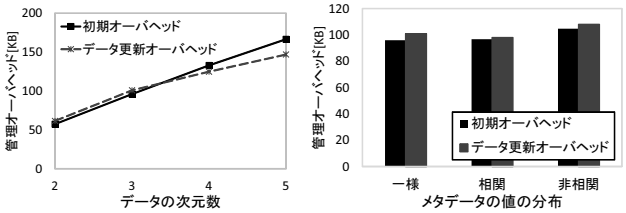
$$e = \begin{cases} 1 & (j \text{ 番目の解が正解集合に含まれる}) \\ 0 & (j \text{ 番目の解が正解集合に含まれない}) \end{cases} \tag{6}$$

  したがって，MAP は，より上位のデータを取得できているほど高い値となる．
- 遅延 : 全検索クエリに対する，検索クエリ発行端末が検索クエリを発行してから検索結果を取得するまでの平均時間．
- トラヒック : Top-k 検索中に送信されたメッセージの総バイト数を試行回数で割った値．
- アクセス端末数 : Top-k 検索中にクエリが送信された平均端末数．
- 管理オーバヘッド : クラスタリングの際に送信されたメッセージの総バイト数（初期オーバヘッド），およびデータ更新が起きた際に送信するメッセージの総バイト数（データ更新オーバヘッド）．

5.2 評価結果

5.2.1 初期およびデータ更新オーバヘッド

まず，提案手法の管理オーバヘッドを，データの次元数 m，およびメタデータの値の分布を変化させて調べた．その結果を図 3 に示す．このとき，端末数 n は 300 とし，図 3(a) ではメタデータの値の分布は一様とした．また，図 3(b) では，m を 3 とした．



(a) データの次元数 m の影響　(b) メタデータの値の分布の影響

図 3　管理オーバヘッド

図 3(a) より，ClusTo は，m が大きくなるほどトラヒックが増加することがわかる．これは，m が大きくなるほど，他のデータにドミナントされるデータの数が少なくなり，K-スカイバンドに含まれるデータが多くなる．そのため，クラスタ構築メッセージ，およびデータ更新を通知するメッセージのサイズが大きくなる．また，図 3(b) より，メタデータの値の分布が非相関である場合にもトラヒックが増加する．これは，ある次元のメタデータの値が小さいとき，他の次元のメタデータの値が大きいことが多いため，ドミナントされる場合が少なくなり，m が大きい場合と同様の影響が生じる．

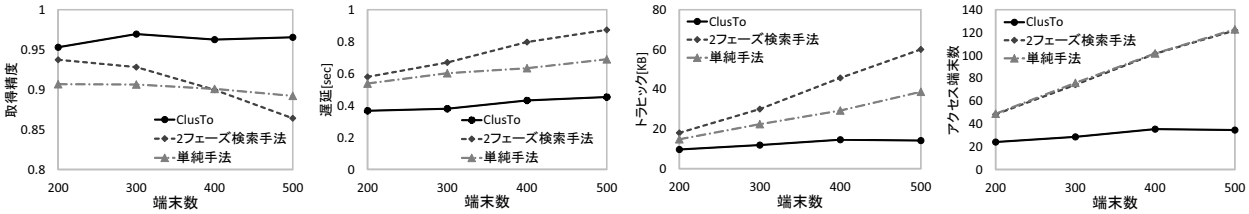
5.2.2 Top-k 検索における性能

次に，各手法の性能を調べるため，端末数 n，および要求データ数 k を変化させた．これらの結果を，それぞれ図 4，および図 5 に示す．このとき，$m = 3$，メタデータの値の分布は一様とした．n を変化させる時の要求データ数 k は，[1, 15] の範囲で決定し，分布は，平均 5，分散 5 の正規分布に従うものとした．k を変化させる時の端末数 n は 300 とした．

端末数 n の影響．図 4(a) より，ClusTo は他の手法よりも高い精度を維持できていることがわかる．これは，検索クエリ，およびクエリ応答を送信する端末数を抑制できているため，トラヒックを削減し，パケット衝突によるデータの損失を抑制できているためである．2 フェーズ検索手法では端末数が多い場合，取得精度が低下する．これは，一度の Top-k 検索においてフラッディングを 2 度行うため，端末数が増えるとパケット衝突が頻繁に発生するからである．

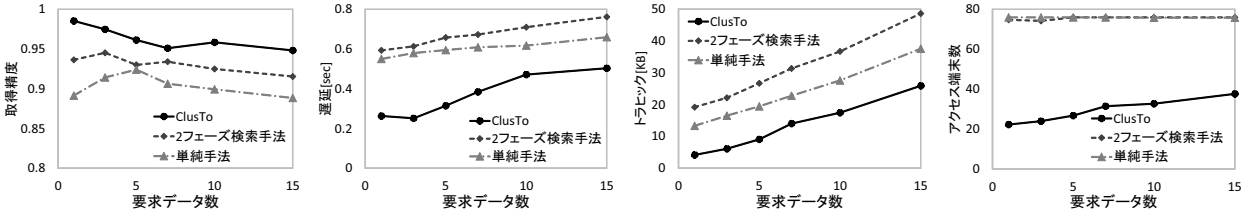
図 4(b) より，ClusTo は，遅延が最も小さい．これは，検索範囲の全ての端末にクエリを送信することなく，Top-k 検索を実行しているためである．また，データ収集クエリを，閾値以下のスコアとなるデータをもつ端末のみに送信しているため，トラヒックを削減し，パケットロスによるメッセージの再送回数を低減できることから，低遅延を達成できている．2 フェーズ検索手法は，2 度のフラッディングによる検索クエリの転送を行なっているため，単純手法よりも遅延が大きい．

図 4(c) より，ClusTo は低トラヒックでの Top-k 検索を処理できることがわかる．2 フェーズ検索手法と単純手法



(a) 取得精度　(b) 遅延　(c) トラヒック　(d) アクセス端末数

図 4　端末数 n の影響



(a) 取得精度　(b) 遅延　(c) トラヒック　(d) アクセス端末数

図 5　要求データ数 k の影響

は端末数が増えると，フラッディングによるトラヒックが増加している．しかし，ClusTo は，クラスタベースのルーティングにより，検索範囲を保証しているため，端末数が増加した場合においても，クエリを送信する端末数は基本的に変わらない．その結果が図 4(d) からわかる．ClusTo はフラッディングを行う手法に比べて，クエリの送信先端末数を大きく削減できている．ここで，n=300 のとき，ClusTo と 2 フェーズ検索手法のトラヒックの差は，29.9−13.4=16.5[KB] である．ClusTo では，初期およびデータ更新でのオーバヘッドが 95.9+100.8=196.7[KB] 発生する．したがって，この環境では，196.7/16.5=11.92$\cdots \fallingdotseq$ 12 回の Top-k 検索を行うと，ClusTo の方が全体のトラヒックが小さくなることが見込まれる．また，これは端末数の 5%未満であり，ネットワーク内の多くの端末の各々が Top-k 検索を行う場合，ClusTo は有効であると言える．

要求データ数 k の影響．図 5(a) より，k が大きくなると，全ての手法において取得精度が低下している．これは，返信するデータ数が多くなるため，トラヒックが増加（図 5(c)）し，パケットロスの機会が増加するためである．しかし，ClusTo は，図 4 の結果の理由から，最も高い精度を維持している．また，k が大きくなるとデータ収集クエリの送信先（アクセス端末数）が増加する（図 5(d)）ため，遅延，およびトラヒックが増加する（図 5(b)，5(c)）．

6. おわりに

本稿では，アドホックネットワークにおいて，いかなる検索条件が指定された場合においても効率的に Top-k 検索を行うことを目的とし，ClusTo を提案した．ClusTo では，スカイラインを用いたクラスタリングを行い，クエリルーティングとフィルタ更新の効率化を実現している．さらに，フィルタ更新では，隣接端末のもつ K-スカイバンドを用いることにより，フィルタの精度を高めることができる．また，クラスタベースでのクエリルーティングの後，閾値以下のデータをもつ端末に検索クエリを送信することにより，検索結果の取得に不要な端末への検索クエリ送信を抑制しつつ，上位 k 個のデータを高確率で取得している．シミュレーション実験の結果から，ClusTo は，検索クエリの送信先端末を可能な限り抑制して Top-k 検索を行なっているため，低トラヒック，および低遅延を達成できることを確認した．

本稿では，固定（ネットワークトポロジが変化しない）アドホックネットワークを想定した．しかし，実際の環境では，端末が移動し，端末間のリンク切断が起きる場合も考えられる．ClusTo は，隣接端末の情報に基づいて，フィルタの更新やクエリ送信を行なっているため，トポロジの変化が問題となる．そのため，今後はモバイル環境への拡張について取り組む予定である．

謝辞　本研究の一部は，文部科学省研究費補助金・基盤研究 S（21220002），および基盤研究 B（24300037）の研究助成によるものである．ここに記して謝意を表す．

参考文献

[1] Amagata, D., Sasaki, Y., Hara, T., and Nishio, S.: A robust routing method for top-k queries in mobile ad hoc networks, *Proc. Int. Conf. on MDM*, pp.251–256 (2013).

[2] Børzsønyi, S., Kossmann, D., and Stocker, K.: The skyline operator, *Proc. Int. Conf. on Data Engineering*, pp.421–430 (2001).

[3] Hagihara, R., Shinohara, M., Hara, T., and Nishio, S.: A message processing method for top-k query for traffic reduction in ad hoc networks, *Proc. Int. Conf. on MDM*, pp.11–20 (2009).

[4] Jiang, H., Cheng, J., Wang, D., Wang, C., and Tan, G.: Continuous multi-dimensional top-k query processing in sensor networks, *Proc. Int. Conf. on INFOCOM*, pp.793–801 (2011).

[5] Papadias, D., Tao, Y, Fu, G., and Seeger, B.: Progressive skyline computation in database systems, *ACM Trans. Database Systems*, Vol.30, No.1, pp.41–82 (2005).

[6] Sasaki, Y., Hara, T., and Nishio, S.: Two-phase top-k query processing in mobile ad hoc networks, *Proc. Int. Conf. on Network-Based Information Systems*, pp.42–49 (2011).

[7] Vlachou, A., Doulkeridis, C., Nørvåg, K., and Vazirgiannis, M.: On efficient top-k query processing in highly distributed envirinments, *Proc. Int. Conf. on SIGMOD*, pp.753–764 (2008).

[8] Zou, L., Chen, L.: "Pareto-based dominant graph: an efficient indexing structure to answer top-k queries", *IEEE Trans. on Knowledge and Data Enginieering*, Vol.23, No.5, pp.727–741 (2011).