高次元データへのアクセス回数を削減する近傍グラフ更新手法

吉田　哲也† HakimHACID††

† 北海道大学大学院情報科学研究科　〒 060–0814 札幌市北区北 14 条西 9 丁目
†† ERIC Laboratory, Lyon 2 University　5, avenue Pierre Mendès-France, 69676 Bron cedex, France
E-mail: †yoshida@meme.hokudai.ac.jp, ††hhacid@eric.univ-lyon2.fr

あらまし　高次元空間におけるデータの近傍探索は，データベースやデータマイニングなどの様々な分野で活用される重要な技術の一つである．近傍グラフは，直観的に理解しやすいメカニズムで各データに対する近傍（隣接データ）を定義したグラフ構造であり，近傍探索を実現する上で各データに対する近傍への索引として利用できる．しかし，近傍グラフの構築に加え，構築したグラフの更新に要する計算量が大きいという問題がある．この問題に対して近傍グラフの更新を局所的に実現する手法も提案されているが，この手法では全てのデータに 2 回のアクセスする必要があり，データサイズが膨大になった場合に課題が残る．本稿では，この手法を改善して，高次元データへのアクセス回数を削減する近傍グラフ更新手法を検討する．
キーワード　近傍グラフ，更新手法，高次元データ

A Neighborhood Graph Updating Method with Reduced Database Access for High Dimensional Data

Tetsuya YOSHIDA† and Hakim HACID††

† Grad. School of Info. Science and Technology, Hokkaido University　N-14 W-9, Sapporo, 060–0814 Japan
†† ERIC Laboratory, Lyon 2 University　5, avenue Pierre Mendès-France, 69676 Bron cedex, France
E-mail: †yoshida@meme.hokudai.ac.jp, ††hhacid@eric.univ-lyon2.fr

Abstract　The point location (neighborhood search) in a multidimensional space is a significant problem in several fields such as databases (DBs) and data mining (DM). Neighborhood graphs are interesting and widely utilized representation to realize this due to their intuitive mechanisms for neighborhood determination. One drawback of this approach is that, besides the construction of graphs, update complexity is very high. A method for local updating to tackle this problem was proposed previously. However, it might not be so scalable since it requires two DB scanning, which are time consuming when the size of DB gets huge. Toward reducing the expensive DB scanning, we propose a neighborhood graph updating method with theoretical properties.
Key words　neighborhood graph, updating method, high dimensional data

1. はじめに

情報技術の発展に伴い，ますます複雑で高性能な情報処理の実現が求められるようになっている．マルチメディア，ウェブなど，様々な表現形式やメディアを通じて情報が伝えられるが，異なるデータ表現形式の処理に加えて，データの大規模性や高次元性が情報処理をいっそう困難なものとしている．

大規模で高次元なデータを処理するために重要な処理の一つとして高次元データ空間における効率的な近傍探索が挙げられる．高次元データ処理の自動化において効率的な近傍探索の実現は必須であり，これまでも多くのアルゴリズムや技術が考案されてきた．たとえば，自己組織化マップ [12]，k-近傍アルゴリズム [2] などはデータマイニングの分野でも頻繁に用いられており，また，多くのデータベース索引付け技術にも用いられている [4].

近傍グラフは人間の直観的な近傍決定メカニズムをある種反映したやりかたで各データに対する近傍（隣接データ）を定義した構造であり，一旦近傍グラフを構築すれば，高次元データにおける近傍探索を実現する際の索引として利用できる表現形式である [3], [10], [13]．しかし，近傍グラフの構築に要する計算量に加え，近傍グラフの更新に要する計算量が大きいという課題がある．この課題に対し，文献 [6], [7] ではデータの追加に対する近傍グラフの局所的な更新手法が提案された．この手法では，高次元空間において追加するデータを中心とする超球を

構築し，更新対象とすべきデータを超球内に局所化することを通じて効率的な更新を実現している．しかし，超球の構築にはデータに 2 回アクセスして全データをスキャンする必要があるため，データサイズが膨大になった場合に課題が残る．

本稿では，文献 [6], [7] での手法を発展させ，データへのアクセス回数を低減した近傍グラフ更新手法を検討する．全てのデータ対間の距離に対する最大値を d_{max} として保存し，d_{max} に基づいて文献 [6], [7] での手法に対する上限の役割を果たす超球を構築して更新対処とすべきデータを局所化して効率的な更新を実現する．更に，d_{max} の利用に伴う問題への対処を検討し，更新手法として提案する．

2. 節では，近傍探索および近傍グラフについて簡単に紹介し，近傍グラフの更新手法についても概説する． 3. 節では本稿で提案するデータへのアクセス回数を低減した近傍グラフ更新手法の詳細について述べる． 3. 節での手法の正当性に対する検証実験の結果を 4. 節で報告し， 5. 節で本稿のまとめについて述べる．

2. データベースにおける近傍探索

データベースにおける近傍探索の役割は，データベースに保存するデータをあらかじめ前処理しておき，クエリとなるデータに対する近傍データを比較的短時間で見つけることにある．1 次元データに対する近傍探索は，データをソートしておき 2 分探索を用いることにより，n をデータ数として $O(n \log n)$ で効果的に実現できる．2 次元データに対する近傍探索も，多次元空間におけるトポロジーモデルの基礎となるボロノイ図を用いることで比較的容易に実現できる [10].

しかし，次元数 p が増加した場合には，高次元空間における近傍探索は複雑になり実現することが困難になる．これまでにも高次元空間における近傍探索に対して様々な手法が提案されてきた．たとえば，1 つの軸に対するデータの射影を利用する手法 [2], [5], [9] や，データ間の部分距離を用いる手法 [1] などが提案されてきた．

本稿では，上記の問題に対する別のアプローチとして，データベース中のデータの関係を近傍グラフを用いて表現し，近傍グラフを用いて近傍探索を実現するアプローチに着目する．以下では，近傍グラフを用いる手法について簡単に紹介する．

2.1 近傍グラフ

近傍グラフ（Neighborhood graphs, proximity graphs）とは近傍の概念に基づく幾何構造であり，各データに対する近傍データを同定するために用いられる．

2.1.1 表　　記

p 次元空間 $\mathcal{R}^p$ におけるデータの集合を$\boldsymbol{V}$と表記する．$|\boldsymbol{V}|$ で集合$\boldsymbol{V}$の要素数を表す．集合$\boldsymbol{V}$の各要素 v を頂点に対応させることを通じて，頂点集合$\boldsymbol{V}$と辺集合 $\boldsymbol{E} : \boldsymbol{V} \times \boldsymbol{V}$ からなるグラフを $G(\boldsymbol{V}, \boldsymbol{E})$ と表記する．

頂点集合$\boldsymbol{V}$の各頂点 v に対し，関数 $N(v)$ は頂点 v に接続する隣接頂点の集合を返す．すなわち，$N(v)$ に含まれる全ての頂点 v' に対し，任意の頂点対 (v, v') は辺集合$\boldsymbol{E}$に含まれる．距離関数 $d : \boldsymbol{V} \times \boldsymbol{V} \to \mathcal{R}^+$ は何らかの距離尺度に基づいて定義され，通常，関数 $d(\cdot, \cdot)$ は対称であることが仮定される [14]. 以下では，グラフ $G(\boldsymbol{V}, \boldsymbol{E})$ は無向な単純グラフであると仮定する．

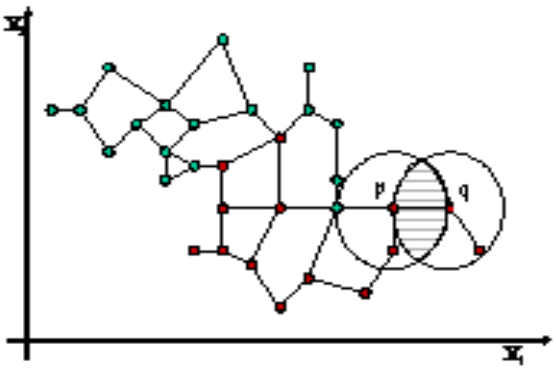

図 1　2 次元空間における相対近傍グラフの例

2.1.2 近傍グラフの例

これまでに様々な近傍グラフの表現が提案されている．近傍グラフの表現として，たとえば，ドローネ三角形 [10]，相対近傍グラフ (RNG) [13]，ガブリエルグラフ [3]，最小全域木 [10] などがある．以下では相対近傍グラフを例として説明する．

相対近傍グラフ $G_{RNG}(\boldsymbol{V}, \boldsymbol{E})$ においては，頂点集合$\boldsymbol{V}$に含まれる任意の 2 頂点 (v_i, v_j) は，以下で定義される相対近傍性を満たす時に限ってその頂点間に辺を持つ．

[定義 1]（相対近傍性） 頂点 v_i を中心として半径 $d(v_i, v_j)$ である超球を $\mathcal{H}(v_i, v_j)$ とし，頂点 v_j を中心として半径 $d(v_j, v_i)$ である超球を $\mathcal{H}(v_j, v_i)$ とする．2 つの超球 $\mathcal{H}(v_i, v_j)$ と $\mathcal{H}(v_j, v_i)$ の交わりから成る領域 $\mathcal{A}(v_i, v_j)$ に任意の頂点 $v \in \boldsymbol{V}$ が含まれない．すなわち，

$$\mathcal{A}(v_1, v_2) = \mathcal{H}(v_1, v_2) \cap \mathcal{H}(v_2, v_1) \tag{1}$$

$$(v_i, v_j) \in \boldsymbol{E} \;\; iff \;\; \mathcal{A}(v_i, v_j) \cap \boldsymbol{V} = \phi \tag{2}$$

与えられた頂点集合$\boldsymbol{V}$に対して定義 1 の相対近傍性を満たす頂点間にのみ辺を張ることにより，相対近傍グラフ $G_{RNG}(\boldsymbol{V}, \boldsymbol{E})$ が構築される [13]．2 次元空間における相対近傍グラフの例を図 1 に示す．

他の近傍グラフに対しても，定義 1 に対応する近傍性を適宜定義しなおすことにより，与えられた頂点集合$\boldsymbol{V}$に対して近傍グラフが構築される．なお，以下の節で述べる近傍グラフの更新手法は，相対近傍グラフに限らず一般の近傍グラフに対しても適用可能であることに注意されたい．

2.2 近傍グラフ構築アルゴリズム

これまで様々な近傍グラフ構築アルゴリズムが提案されてきた．ここでは図 1 に示した相対近傍グラフに対するアルゴリズムについて説明する．ドローネ三角形などの他の表現に対するアルゴリズムに対しては文献 [16] を参照されたい．

近傍グラフ構築アルゴリズムの多くに共通して用いられる技法として，グラフを逐次的に洗練させていくアプローチがある．このアプローチでは，対称とする近傍グラフ表現を定義する近

傍性を満たさない辺を逐次的に削除することを通じて近傍グラフを構築する．辺の削除は，通常，近傍グラフの定義や幾何的性質を通じて行われる．

このアプローチでは，

1) データ v の近傍にあるデータ集合の列挙

2) 近傍グラフを定義する領域（近傍性）に 1) で求めたデータが含まれるか否かの確認

を繰り返すことでグラフを構築する．相対近傍グラフの場合では，頂点 v の近傍集合 N(v) に含まれる各頂点 v' に対し，領域 $\mathcal{A}(v,v')$ に含まれるデータの有無を確認することに対応する．上記の操作は，$n=|\boldsymbol{V}|$, $\boldsymbol{V}\subseteqq\mathcal{R}^p$ として，標準的な手法を用いると $O(n^3)$ の計算量を要する．

Toussaint は，ドローネ三角形から出発して相対近傍グラフを $O(n^2)$ で構築するアルゴリズムを提案した [14]．Katajainen は同じ計算量を要する別のアルゴリズムを提案し [8]，Smith は $O(n^{23/12})$ で構築するアルゴリズムを提案した [11]．これらのアルゴリズムは $O(n^3)$ を要する標準的な手法と比較して低い計算量で近傍グラフを構築でき，高速に動作するという利点がある．しかし，一旦構築した近傍グラフの更新は考慮していないため，新しいデータを追加するためには新たなグラフを再構築しなおす必要がある．

2.3 データ追加に対する近傍グラフ更新手法 [6], [7]

文献 [6], [7] では，追加されるデータと，追加されるデータにより影響を受ける辺を考慮した近傍グラフ構築手法が提案された．この手法では，下記のステップによりデータ q の追加に伴う近傍グラフ $G(\boldsymbol{V},\boldsymbol{E})$ の更新を行う．

1) $\mathcal{R}^p$ において，q を中心として半径 r の超球 SR を構築することで，q の近傍となりうる頂点（データ）を超球 SR 内に限定する．

2) 超球 SR 内に限定してグラフ $G(\boldsymbol{V},\boldsymbol{E})$ の更新を計算し，辺集合$\boldsymbol{E}$を修正する．

上記では，データ q の最近傍データに隣接するデータが全て超球 SR に含まれるように半径 r を決定する．なお，データ q の追加に伴うグラフの更新により，q に対する近傍（隣接）データを得ることができるため，q をクエリと見なすことができる．以下では，半径の長さおよびその半径で規程される超球 SR を添え字で区別して表記する．たとえば，文献 [6], [7] の手法における半径および超球はそれぞれ r_{nf}，SR_{nf} と表記する．

文献 [6], [7] での手法では，半径と超球を下記により決定していた．追加するデータ q の最近傍頂点を頂点 v_{q1} とし，その間の距離を $d_1=d(q,v_{q1})$ とする．また，頂点 v_{q1} に接続する頂点のうち最遠頂点を頂点 v_{q2} とし，その間の距離を $d_2=d(v_{q1},v_{q2})$ とする．頂点 v_{q1}, v_{q2} に基づき，半径 r_{nf} を下記で定義する．

$$r_{nf}\equiv(d_1+d_2)(1+\epsilon) \tag{3}$$

ここで，$\epsilon\in[0,1]$ はパラメータであり，扱うべきデータの性質に応じて決定される．

上記で，データ数を $n(=|\boldsymbol{V}|)$，超球 SR_{nf} に含まれるデータ数を n' ($n'\ll n$) とすると，更新に要する計算量は $O(2n+n'^3)$ である．この中の各項は

1. 全データ中から q に対する最近傍データ v_{q1} の決定 ($O(n)$) と，超球 SR_{nf} に含まれるデータの決定 ($O(n)$)
2. 超球 SR_{nf} に限定した頂点に対する辺の追加および削除 ($O(n'^3)$)

に対応する．上記で，仮定 $n'\ll n$ より 2. に対する処理は標準的な手法を用いていた．

以下では，上記の手法を LocalInsert($G(\boldsymbol{V},\boldsymbol{E})$,$q$,$\epsilon$) と表記することとする．LocalInsert($G(\boldsymbol{V},\boldsymbol{E})$,$q$,$\epsilon$) はパラメータ ϵ のもとでデータ q をグラフ $G(\boldsymbol{V},\boldsymbol{E})$ を追加したグラフを返す．文献 [6], [7] での評価実験を通じ，LocalInsert($G(\boldsymbol{V},\boldsymbol{E})$,$q$,$\epsilon$) により与えられたデータ集合$\boldsymbol{V}$に対してデータ q を追加することにより正しい近傍グラフ(注1) が得られることが確認されている．

3. データへのアクセス回数を削減する近傍グラフ更新手法

計算幾何学の分野において，近傍グラフの構築や管理における計算量の低減に向けて様々な研究がなされてきた [15]．本稿では，2.3 節で紹介した近傍グラフ更新手法に焦点を当て，このアプローチに沿いながらデータへのアクセス回数を削減する更新手法を検討する．

3.1 データへのアクセス回数を削減したデータの追加

2.3 節で述べた LocalInsert($G(\boldsymbol{V},\boldsymbol{E})$,$q$,$\epsilon$) においては，データ量が増えた場合にはデータのスキャンに要する時間がボトルネックとなる可能性がある．特に，データ q の追加に伴う近傍グラフの更新を超球 SR_{nf} に限定して実行する際，q に対する最近傍データ v_{q1} の同定と，構築した超球 SR_{nf} 内に含まれるデータの同定を行うために，それぞれ全てのデータをスキャンする必要があった．

データへのアクセス回数を減らすために，本稿では超球の半径を LocalInsert($G(\boldsymbol{V},\boldsymbol{E})$,$q$,$\epsilon$) とは異なるアプローチで決定することを検討する．具体的には，近傍グラフ $G(\boldsymbol{V},\boldsymbol{E})$ を構築する際，グラフ G に加えて，辺集合$\boldsymbol{E}$における最大距離を d_{max} として保存し，この値を半径の決定に利用する．

$$d_{max}\equiv\arg\max_{v_i,v_j\in\boldsymbol{V}}d(v_i,v_j) \tag{4}$$

式 (4) の定義より，以下の性質が成り立つ．

[性質 1] 頂点集合$\boldsymbol{V}$に含まれる任意の 2 頂点 v_i,v_j ($v_i\neq v_j$) に対し，頂点間に辺 (v_i,v_j) があれば，$d(v_i,v_j)\leqq d_{max}$ が成り立つ．

d_{max} に基づき，半径 $r_{d_{max}}$ を以下で定義する．

$$r_{d_{max}}\equiv 2d_{max}(1+\epsilon) \tag{5}$$

性質 1 より，辺集合$\boldsymbol{E}$に含まれる全ての辺に対し，$d_1\leqq d_{max}$, $d_2\leqq d_{max}$ が成り立つ．このため，以下の性質がなりたつ．

(注1)：本稿では，与えられた頂点集合$\boldsymbol{V}$に対し，近傍グラフの性質を満たすのに必要十分な辺集合のみを含むグラフを正しい近傍グラフと呼ぶ．頂点集合$\boldsymbol{V}$に対して構築した近傍グラフ G の正しさは，$O(n^3)$ を要する標準的な手法で構築したグラフ G' との同型性の判定により確認できる．

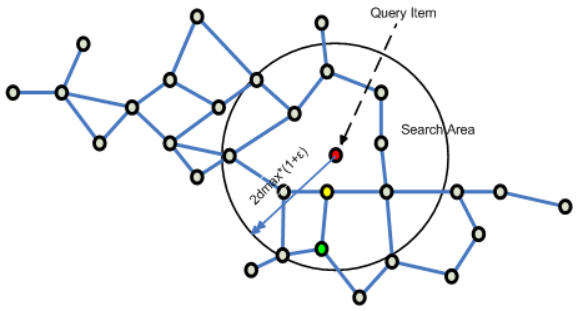

図 2　q を中心とする超球 $SR_{d_{max}}$ の構築

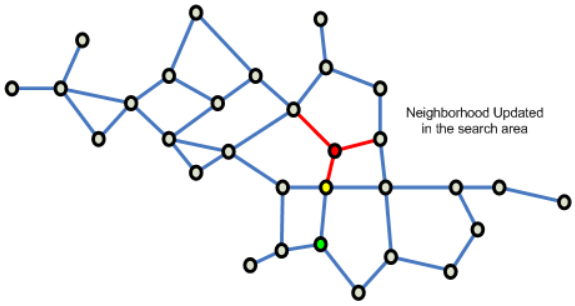

図 3　超球 $SR_{d_{max}}$ に含まれるデータに対してのみグラフの更新

［性質 2］　$r_{nf} = (d_1 + d_2)(1+\epsilon) \leqq 2d_{max}(1+\epsilon) = r_{d_{max}}$　for $\forall q \in \boldsymbol{V}$.

d_{max} を用いることにより，集合 $\boldsymbol{V}$ に含まれる任意の q に対して r_{nf} の上限に対応する $r_{d_{max}}$ を計算し，q を中心とした半径 $r_{d_{max}}$ の超球 $SR_{d_{max}}$ を構築する．

超球 $SR_{d_{max}}$ の定義より，以下の性質が成り立つ．

［性質 3］　$SR_{nf} \subseteqq SR_{d_{max}}$　for　$\forall q \in \boldsymbol{V}$.

上記のアプローチでは，集合$\boldsymbol{V}$に含まれる任意の q に対して最近傍点 v_{q1}（および，v_{q1} に対する最遠近傍点 v_{q2}）を求める必要がないため，1 回のみのスキャン（$O(n)$）で超球を構築できる．このため，d_{max} に基づく手法の計算量は，$n = |\boldsymbol{V}|$, 超球 $SR_{d_{max}}$ 内のデータ数を $n_{d_{max}}$ として，$O(n + n_{d_{max}}^3)$ である．LocalInsert$(G(\boldsymbol{V},\boldsymbol{E}),q,\epsilon)$ と同様に，超球 $SR_{d_{max}}$ に含まれるデータの同定には $O(n)$ 必要である．

性質 3 より $q \in \boldsymbol{V}$ に対して $SR_{nf} \subseteqq SR_{d_{max}} \subseteqq \boldsymbol{V}$ が成り立つため，以下の性質が成り立つ．

［性質 4］　$n' \leqq n_{d_{max}}$　for　$\forall q \in \boldsymbol{V}$.

上記の性質に基づき，$q \in \boldsymbol{V}$ に対しては超球 $SR_{d_{max}}$ を利用する更新手法に対して以下の性質が成り立つ．

［性質 5］　2.3 節で述べた LocalInsert$(G(\boldsymbol{V},\boldsymbol{E}),q,\epsilon)$ が $q \in \boldsymbol{V}$ に対して近傍グラフ G' の更新により正しい近傍グラフ G を構築するならば，超球 $SR_{d_{max}}$ に基づく手法も正しい近傍グラフ G を構築する．

証明　性質 3 より，$\forall q \in \boldsymbol{V}$ に対する更新では超球 SR_{nf} に含まれるデータは全て超球 $SR_{d_{max}}$ に含まれる．このため，もし q を中心とする超球 SR_{nf} に含まれる部分グラフのみを更新することで正しい近傍グラフ G が得られるならば，同様に q を中心とする超球 $SR_{d_{max}}$ に含まれる部分グラフのみを更新することで正しい近傍グラフ G が得られる．□

図 2，3 に d_{max} に基づく更新手法の挙動を示す．

3.1.1　d_{max} に伴う問題と解決法

d_{max} に基づく更新手法はデータへのアクセス回数を減らすことが可能であるが，d_{max} を用いるだけでは正しい近傍グラフに更新できない場合がある．図 4，5 に一例を示す．上述の手法でも，$r_{nf} \leqq r_{d_{max}}$ が成り立つならば $q \notin \boldsymbol{V}$ に対しても近傍グラフを正しく更新できる．しかし，$r_{nf} > r_{d_{max}}$ であるような q に対しては，$SR_{d_{max}} \cap \boldsymbol{V} = \phi$ となり更新時に考慮すべきデータを超球 $SR_{d_{max}}$ 内に限定できないため，更新が正しく行われないことになってしまう．

近傍グラフ

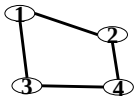

図 4　近傍グラフ G と，G に対する d_{max} の計算

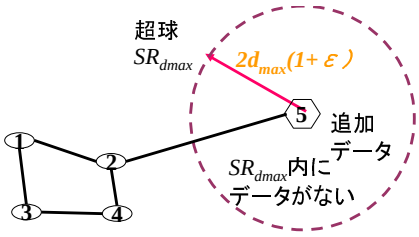

図 5　新規データ q の G への追加．集合$\boldsymbol{V}$に含まれる任意のデータ v に対して $d(q,v) > 2d_{max}(1+\epsilon)$ の場合，$SR_{d_{max}} \cap \boldsymbol{V} = \phi$ となり G が正しく更新されない．

上記の問題は，集合 $\boldsymbol{V}$ に含まれるデータ，および，$q \notin \boldsymbol{V} \wedge r_{nf} \leqq r_{d_{max}}$ が成り立つようなデータに対しては性質 2–5 が成り立つが，一般には r_{nf} は $r_{d_{max}}$ より大きい場合があるため $q \notin \boldsymbol{V}$ に対しては成り立たないことによる．この問題に対処するため，近傍グラフ $G(\boldsymbol{V},\boldsymbol{E})$ に対して d_{max} と LocalInsert$(G(\boldsymbol{V},\boldsymbol{E}),q,\epsilon)$ における SR_{nf} を組み合わせることにより，$SR_{d_{max}} \cap \boldsymbol{V} = \phi$ の場合に対しても正しい更新を実現する．

この手法では，まず d_{max} に基づく更新を試み，$SR_{d_{max}} \cap \boldsymbol{V} = \phi$ の場合にのみ SR_{nf} に基づく更新を行う．なお，併用することによる計算量の増加を防ぐため，超球 $SR_{d_{max}}$ に含まれる頂点（データ）を全データ中から探し出す際，同時に q に対する最近傍点 v_{q1} も見つけて v_{q1} への参照を保存しておく．v_{q1} も同時にみつけておくことにより，$SR_{d_{max}} \cap \boldsymbol{V} = \phi$ の場合であっても v_{q1} を見つけるためのスキャンを省いて SR_{nf} に基づく更新を行うことが可能となる．

超球 $SR_{d_{max}}$ を利用する際の別の問題としては，d_1 と d_2 に対して d_{max} が非常に大きな見積もりとなった場合，$r_{nf} \ll r_{d_{max}}$ となってしまうことにより $|SR_{nf}| \ll |SR_{d_{max}}|$ となる場合がある．この場合，超球 $SR_{d_{max}}$ は更新対象となるデータの局所化に貢献せず，効率的な更新ができなくなってしまう．

この問題に対処するため，$SR_{d_{max}}$ を更に利用し，SR_{nf} に類似した局所化を行い更新を行う．LocalInsert$(G(\boldsymbol{V},\boldsymbol{E}),q,\epsilon)$ に

Algorithm 1 LocalInsert$_{d_{max}}(G(\boldsymbol{V},\boldsymbol{E}),q,d_{max},\epsilon)$

Require: $G(\boldsymbol{V},\boldsymbol{E})$; //近傍グラフ
Require: q; //近傍グラフ G に追加するデータ
Require: d_{max}; //集合$\boldsymbol{E}$に含まれる辺の距離の最大値
Require: ϵ; //$\epsilon \in [0,1]$

1: $r_{d_{max}} = 2d_{max}(1+\epsilon)$;
2: $\boldsymbol{V}_{d_{max}} = \{v \in \boldsymbol{V} | d(q,v) \leqq r_{d_{max}}\}$;
3: $v_{q1} = G$ における q への最近傍点;
4: $d_1 = d(q, v_{q1})$;
5: $v_{q2} = G$ において v_{q1} に接続する頂点の中の最遠点;
6: $r_{nf} = (d_1 + d_2)(1+\epsilon)$;
7: **if** $(\boldsymbol{V}_{d_{max}} \neq \phi)$ **then**
8: $\quad \boldsymbol{V}^{SR} = \{v \in \boldsymbol{V}_{d_{max}} | \ r_{nf} > d(q,v)\}$;
9: **else**
10: $\quad \boldsymbol{V}^{SR} = \{v \in \boldsymbol{V} | \ r_{nf} > d(q,v)\}$;
11: **end if**
12: $\boldsymbol{V} = \boldsymbol{V} \cup \{q\}$;
13: G の部分グラフ G_{sub} に対して集合 $\boldsymbol{E}$ の更新 ($G_{sub} \subseteqq G$ であり，G_{sub} は $\boldsymbol{V}^{SR}$ 内に含まれる頂点と辺のみを含む)
14: **return** （更新後の）$G(\boldsymbol{V},\boldsymbol{E})$;

おける SR_{nf} との相違は，全データ$\boldsymbol{V}$に対してグラフ G を更新するのではなく，$SR_{d_{max}}$ に含まれるデータに対してのみ更新を行う点にある．$SR_{d_{max}} \subseteqq \boldsymbol{V}$ が成り立つため，更新のためにチェックすべきデータ数（頂点数および辺数）が減少し，単純に $SR_{d_{max}}$ を利用する場合に比べて能が向上すると期待される．

上述した手法を以下では LocalInsert$_{d_{max}}(G(\boldsymbol{V},\boldsymbol{E}),q,d_{max},\epsilon)$ と呼び，アルゴリズム 1 に示す．提案する手法は近傍グラフの更新に対して以下の性質を満たす．

［性質 6］ LocalInsert$_{d_{max}}(G(\boldsymbol{V},\boldsymbol{E}),q,d_{max},\epsilon)$ は，近傍グラフの更新に対して LocalInsert$(G(\boldsymbol{V},\boldsymbol{E}),q,\epsilon)$ と同程度かより少ないデータアクセス数を要する．

証明 定義より $SR_{d_{max}} \subseteqq \boldsymbol{V}$ が成り立つため，$|SR_{d_{max}}| \leqq |\boldsymbol{V}|$ が成り立つ．$SR_{d_{max}} \cap \boldsymbol{V} = \phi$ の場合，LocalInsert$_{d_{max}}(G(\boldsymbol{V},\boldsymbol{E}),q,d_{max},\epsilon)$ は LocalInsert$(G(\boldsymbol{V},\boldsymbol{E}),q,\epsilon)$ と等価である．他方，$SR_{d_{max}} \cap \boldsymbol{V} \neq \phi$ の場合，$SR_{d_{max}} \subseteqq \boldsymbol{V}$ が成り立つため，アルゴリズム 1 の 8 行目の $\boldsymbol{V}^{SR}$ は，10 行目の $\boldsymbol{V}^{SR}$（LocalInsert$(G(\boldsymbol{V},\boldsymbol{E}),q,\epsilon)$ で構築する超球に対応）より小さい．このため，LocalInsert$_{d_{max}}(G(\boldsymbol{V},\boldsymbol{E}),q,d_{max},\epsilon)$ で要するデータアクセス数はより少なくなる．□

$SR_{d_{max}} \cap \boldsymbol{V} = \phi$ の場合， 2.3 節で述べた LocalInsert$(G(\boldsymbol{V},\boldsymbol{E}),q,\epsilon)$ と等価になる(注2) しかし，これは追加すべきデータ q が集合$\boldsymbol{V}$に含まれる全データよりも遙かに遠くにあり，いわば集合$\boldsymbol{V}$に含まれる全データにとって q が外れ値のような場合にのみ生じる．通常データベースは関連するデータを管理するために用いられるため，データベースに蓄えられるデータは何らかの意味で類似していると考えられる．このため，多くのデータに対しては d_{max} に基づいて更新され，LocalInsert$_{d_{max}}(G(\boldsymbol{V},\boldsymbol{E}),q,d_{max},\epsilon)$ による性能向上が期待される．

(注2)：最初に述べたように，本稿ではデータへのアクセス回数を減少することに焦点を当てるため，q に対する v_{q1} の同定と同時に行われる超球 $\boldsymbol{V}^{SR}$ の構築に要する計算量は無視している．ただし，計算機を用いた実測値を比較する場合には，この計算時間も無視できない．

4. 予備的評価

本節では 3. 節で提案した LocalInsert$_{d_{max}}$ $(G(\boldsymbol{V},\boldsymbol{E}),q,d_{max},\epsilon)$ に対する以下の 2 つの観点からの予備的評価について報告する．

1) 近傍グラフ更新の正しさ
2) 実機での実行時間

実験に際し，異なるサイズ ($n = |\boldsymbol{V}| = 5\times10^3, 10\times10^3, 20\times10^3$) を持つデータを生成した．なお，生成したデータにおける次元数 p は 60 とした．データの生成は，$\mathcal{R}^p$ における一様分布からのランダムサンプリングにより行った．また，データ数 $n = 5\times10^3$，次元数 p= 20 である実データに対する実験も行った．実験では，予備実験の結果に基づき ϵ を 0.1 とし，追加するデータが近傍グラフに対する凸包に含まれる場合を対象とした．

以下で報告する評価は予備的なものにとどまり，提案手法の原理的な性質の確認にとどまっている．実機上での性能の検証については今後実験を重ね，発表時に報告する予定である．

4.1 評価結果

1) に対しては，LocalInsert$_{d_{max}}(G(\boldsymbol{V},\boldsymbol{E}),q,d_{max},\epsilon)$ の正しさを以下の手順により確認した．各近傍グラフ $G(\boldsymbol{V},\boldsymbol{E})$ に対し，頂点集合$\boldsymbol{V}$から頂点 v をランダムに選択し，v を含まない近傍グラフ $G'(\boldsymbol{V}\backslash\{v\},\boldsymbol{E'})$) を作成する．次に，頂点 v を LocalInsert$_{d_{max}}(G'(\boldsymbol{V}\backslash\{v\},\boldsymbol{E'}),v,d_{max},\epsilon)$ を用いて G' に追加し，もとのグラフ G と同型なグラフが生成されるかを確認した．各グラフに対して上記の手順を繰り返したところ，全ての場合において LocalInsert$_{d_{max}}(G(\boldsymbol{V},\boldsymbol{E}),q,d_{max},\epsilon)$ もとのグラフ G と同型なグラフが生成されることを確認した．これにより，提案した LocalInsert$_{d_{max}}(G(\boldsymbol{V},\boldsymbol{E}),q,d_{max},\epsilon)$ による近傍グラフ更新の正しさを確認した．

2) に対しては，データ追加に要する実行時間を計測した 10 回平均の結果を表 4.2 に示す．実行時間の計測は 2.8 GHz の CPU，512MB のメモリ，Windows XP を OS とする計算機上で行った．上述のように LocalInsert$_{d_{max}}(G(\boldsymbol{V},\boldsymbol{E}),q,d_{max},\epsilon)$ は正しい近傍グラフを構築するが，残念ながら現状では LocalInsert$(G(\boldsymbol{V},\boldsymbol{E}),q,\epsilon)$ よりもより多くの時間を要しており，実機上での性能向上を確認できなかった．

4.2 考　　察

4.1 節での結果に対し，現状で推測している要因を以下に述べる．

まず第一に，性質 6 で LocalInsert$_{d_{max}}(G(\boldsymbol{V},\boldsymbol{E}),q,d_{max},\epsilon)$ の原理的な性質について述べたが，対象とするデータセットによっては $r_{d_{max}}$ は r_{nf} と比較して非常に大きくなり，性質 6 を保証するためには $|SR_{d_{max}}|$ が非常に大きくなる可能性がある．このため，アルゴリズム 1 の 2 行目で構築する $\boldsymbol{V}_{d_{max}}$ は近傍グラフ更新に対するデータを十分に局所化できていない可能性がある．

第 2 に，現在の実装では $\boldsymbol{V}_{d_{max}}$ に含まれる全データ v に対

表 1　LocalInsert と LocalInsert$_{d_{max}}$ の平均時間を示す．$n = 5 \times 10^3$ に対する行は実データに対する結果．

データセット		実行時間 (ミリ秒)	
n ($\times 10^3$)	dimension	LocalInsert	LocalInsert$_{d_{max}}$
5	60	55.2	109.2
10	60	139.1	282.7
20	60	256.2	784.6
5*	21*	24.57	48.71

して距離計算を繰り返すため，距離計算を 2 行目と 8 行目（あるいは 10 行目）の 2 回繰り返してしまっている．データアクセスに要する原理的な解析結果に加え，実機での実行時間は距離計算にも影響を受けるため，次元数 p が大きい場合には距離計算に要する実行時間が無視できなくなる．実装レベルでこの問題を回避するため，2 行目で行った計算結果を保存し，8 行目と 10 行目で再利用することで距離の再計算を避けることを検討している．全データに対する距離値を保存するために更にメモリを使用することになるが，距離計算に伴う実行時間を抑えることが可能になると期待される．

第 3 に，現状で生成したデータ規模が小さいため，全てオンメモリでの検索が可能であり，データアクセス数を低減させた効果が実行時間に反映されていない可能性がある．近傍グラフを用いる利点は，近傍グラフの構造を近傍データ抽出に対する索引として利用することにある．この点を検証するためには，メモリに入りきらない大規模なデータを生成し，ハードディスクなどの 2 次記憶へのアクセスが必要な場合に対して実験を行う必要がある．

今後は，上記で検討した項目の検証実験および実装を進め，その結果を報告する予定である．

5. おわりに

本稿ではデータへのアクセス回数を削減する近傍グラフ更新手法について提案した．文献 [6], [7] の手法は近傍グラフの更新を実現するが，更新には全てのデータに 2 回アクセスして更新のためのデータを同定する必要があり，データサイズが膨大になった場合に課題が残る．データへのアクセス回数の削減に向けて，本稿では全てのデータ対間の距離に対する最大値を d_{max} として保存し，d_{max} に基づいて文献 [6], [7] での手法に対する上限となるような超球を利用する更新手法を検討した．更に，d_{max} の利用に伴う問題に対処する方法を検討し，検討した手法を LocalInsert$_{d_{max}}(G(\boldsymbol{V},\boldsymbol{E}),q,d_{max},\epsilon)$ として提案した．提案した手法を計算機上に実装し，近傍グラフ更新に対する正しさを検証した．しかし，残念ながら現状では実機での実行速度に対してはまだまだ課題が残ることも明らかとなった．今後は実機上での実行速度の向上に取り組む予定である．

謝　辞

本研究を支援してくださいました田中譲教授に謝意を表します．本研究の一部は，日本学術振興会先端研究拠点事業–拠点形成型– (No.18001) および文部科学省科研費若手研究 (B) (No.18700131) の補助による．

文　献

[1] C.D. Bei and R. M. Gray. An improvement of the minimum distortion encoding algorithm for vector quantization. *IEEE Trans. on Commu.*, 33:1132–1133, 1985.

[2] J.H. Friedman, F. Baskett, and L.J. Shustek. An algorithm for finding nearest neighbors. *IEEE Trans. Computers*, 24(10):1000–1006, 1975.

[3] K. R. Gabriel and R. R. Sokal. A new statistical approach to geographic variation analysis. *Systematic Zoology*, 18:259–278, 1969.

[4] V. Gaede and O. Günther. Multidimensional access methods. *ACM Computing Survey*, 30(2):170–231, 1998.

[5] L. Guan and M. Kamel. Equal-average hyperplane partitioning method for vector quantization of image data. *Pattern Recognition Letters*, 13(10):693–699, 1992.

[6] H. Hacid and D.A. Zighed. An effective method for locally neighborhood graphs updating. In *16th International Conference on Database and Expert Systems Applications (DEXA '05)*, pages 930–939, 2005.

[7] H. Hacid and D.A. Zighed. Content-based image retrieval in large image databases. In *IEEE International Conference on Granular Computing (GrC 2006)*, pages 498–501, 2006.

[8] J. Katajainen. The region approach for computing relative neighborhood graphs in the lp metric. *Computing*, 40:147–161, 1988.

[9] C.-H. Lee and L. H Chen. Fast closest codeword search algorithm for vector quantisation. *IEE Proc.-Vis. Image Signal Process*, 141:143–148, 1994.

[10] F. Preparata and M. I. Shamos. *Computational Geometry-Introduction*. Springer-Verlag, New-York, 1985.

[11] W. D. Smith. Studies in computational geometry motivated by mesh generation. *PhD thesis, Princeton University*, 1989.

[12] P. Somervuo and T. Kohonen. Self-organizing maps and learning vector quantization for feature sequences. *Neural Processing Letters*, 10(2):151–159, 1999.

[13] G. T. Toussaint. The relative neighborhood graphs in a finite planar set. *Pattern recognition*, 12:261–268, 1980.

[14] G. T. Toussaint. Some insolved problems on proximity graphs. In D.W. Dearholt and F. Harrary, editors, *Proceeding of the first workshop on proximity graphs*, 1991.

[15] Godfried T. Toussaint. Geometric proximity graphs for improving nearest neighbor methods in instance-based learning and data mining. *Int. J. of Computational Geometry Application*, 15(2):101–150, 2005.

[16] E. Welzl, P. Su, and R.L. Drysdale III. A comparison of sequential delaunay triangulation algorithms. *Computational Geometry*, 7:361–385, 1997.