Flexible Distance-based Hashing に基づく 大規模多次元データ集合に対する近似最近傍探索手法の改良

糸谷 友里† 森 啓輔†† 若林 真一† 永山 忍† 稲木 雅人†

上土井 陽子†

† 広島市立大学大学院情報科学研究科

†† 広島市立大学情報科学部

〒 731–3194 広島市安佐南区大塚東 3 丁目 4 番 1 号

E-mail: †mc66023@e.hiroshima-cu.ac.jp,{wakaba,s_naga,inagi,yoko}@hiroshima-cu.ac.jp,

††morikei18@gmail.com

あらまし 本論文では，多次元データに対する近似最近傍探索アルゴリズムを提案する．提案するアルゴリズムは Flexible Distance-based Hashing(FDH) と呼ばれる距離ベースのハッシング手法を拡張した探索手法である．本論文で提案する 2 通りの拡張の 1 つは，与えられたクエリに対して，最近傍が含まれている可能性が従来手法より高い候補集合を選択し，その中からクエリに最も近いデータを最終結果として出力する．この手法の主な利点は，アルゴリズムのパラメータ値の変化による計算時間と探索精度の調整が従来手法より柔軟に実現できることである．もう 1 つの拡張は，探索領域を定義するアンカー集合を複数にした手法である．計算機実験により提案した 2 つの拡張探索手法の有効性を示す．

キーワード 最近傍探索，Flexible Distance-based Hashing (FDH)，アンカー集合，学習アルゴリズム

1 はじめに

最近傍探索問題は，機械学習，パターン認識，コンピュータグラフィックス，情報探索など，多くのアプリケーションをもつ基本的な問題である [2], [4], [5], [6]．この問題は与えられたクエリに対し，定義された距離尺度に関して最も近いデータ（最近傍）をデータ集合 D から見つけることが目的である．最近傍探索問題の典型的かつ重要な場合は，距離尺度がユークリッド距離であり，データ空間の次元が $d \geq 20$ のように非常に高い場合である．本稿ではこの種の最近傍探索問題について考察する．

最近傍探索問題の計算時間を短縮するために，様々なアプローチが提案されている．この問題に対する既存のアルゴリズムの大部分は，木構造のデータ構造を使用する探索アルゴリズムとハッシングに基づく近似最近傍探索アルゴリズムの 2 つに分類することができる．最近傍探索アルゴリズムで使用される木構造のデータ構造の典型的な例は KD-tree，Rtree，B+-tree や Cover tree である．木構造のデータ構造を利用する探索アルゴリズムの大部分は正確な探索結果を出力できるが，多次元大規模データ集合に対しては効率的ではない．

一方，近似最近傍探索アルゴリズムは，解の正確さを犠牲にすることで計算時間とメモリ空間における探索効率の向上を目的とする．近似最近傍探索アルゴリズムにはいくつかの種類がある．その中で，ハッシングを利用するアルゴリズムはよく知られている．Locality-Sensitive Hashing（LSH）やスペクトルハッシングなどの様々な種類のハッシング技法が，良好な精度で効率的に近似最近傍を見つけるために提案されている．

本稿では，ハッシングに基づく多次元データの近似最近傍探索アルゴリズムを提案する．提案したアルゴリズムで採用されているハッシング技法は，Flexible Distance-based Hashing（FDH）である [8]．クエリが与えられると，FDH は最近傍が含まれると思われるデータの部分集合を選択する．それらの部分集合に含まれるデータの中で，クエリに最も近いデータが最終探索結果として出力される．

既存の FDH は，候補探索領域として最近傍を有する領域が選択されない場合が常に存在するという欠点を有する．FDH のこの欠点を克服するために，本稿では新たに 2 つのアイデアを FDH に導入する．1 つは最遠 δ 近傍による探索領域の拡張，もう 1 つは探索領域を定義するアンカー集合を複数に拡張する手法である．これら 2 つの拡張手法に対して計算機実験を行い，提案した近似最近傍探索アルゴリズムの有効性と効率を示した．

本論文は以下のように構成される．第 2 章では，最近傍探索問題と Flexible Distance-based Hashing（FDH）について，第 3 章では，FDH の探索領域を拡張するために最遠 δ 近傍を導入し，これに基づく新しい最近傍探索アルゴリズムを提案する．さらに，アンカー集合を複数とする探索手法を提案する．第 4 章では，本稿で提案した 2 種類の最近傍探索アルゴリズムの実験的評価を示す．最後に，第 5 章で結論を述べる．

2 準　　備

2.1 最近傍探索問題

最近傍探索問題は以下のように定義される．n 個の要素を持つデータ集合 $D = \{r_1, ..., p_n\}$ が与えられたとする．D の各要素は d 個の属性 $(p_1, ..., r_d)$ からなり，各属性 r_i は $[L_i, U_i]$ の範囲内の実数である．ここで，L_i と U_i は属性 r_i の下限と上限を表す．定義から，D の各要素は d 次元空間 R^d のデータ点と見なすことができる．

データ集合 D の最近傍探索は，次のように定義される．R^d 内のクエリ q が与えられたとき，クエリ q からユークリッド距離的に最も近い点を D 内から探索する．以下では，R^d 内の 2 つのデータ点 p と q 間のユークリッド距離を $dist(p, q)$ と表す．

2.2 Flexible Distance-based Hashing（FDH）

Flexible Distance-based Hashing（FDH）は，多次元データ空間における最近傍探索のために提案されたハッシング方法である [8]．データ集合 $D = \{p_1, ..., p_n\}$ が与えられたとすると，この方法は最初に D から A 個のデータ点の集合 $A_n = \{a_1, a_2, ..., a_A\}$ を選択する．それぞれのデータ点 a_i と a_j は $i \neq j$ であり，その距離はできるだけ大きくなるようにとる．各 a_i はアンカーと呼ばれる．A_n の各アンカー a_i に対して，r_i で表される”半径” を決定すると，$|D_{near}(r_i)| \simeq |D_{far}(r_i)|$ となる．ここで，$D_{near}(r_i) = \{p \in D | dist(a_i, p) \leqq r_i\}$ かつ $D_{far}(r_i) = \{p \in D | dist(a_i, p) > r_i\}$ である．

アンカー集合を適切に設定することは，FDH を用いた最近傍探索において良い解を得るために重要である．この問題に対するヒューリスティックなアプローチは，反復改善手法である．この手法では，まずアンカー集合がランダムに選択される．次に，$dist(p, q)$ がすべての対のアンカーの中で最小になるように，1 対のアンカー p および q を選択する．次に，別のデータ点 r が D からランダムに選択され，r と他の点との間の最小距離が $dist(p, q)$ より大きい場合，p はアンカーの集合から外され，r は次のアンカー集合の要素になる．そうでなければ r はアンカー集合の候補から外される．この手順を改善が見られなくなるまで繰り返し行う．

d 次元データ空間 R_d のデータ集合 D に対し，アンカー集合 A_n が与えられたとする．R_d 内のデータ点 p が与えられると，$BM(p) = b_1, b_2, ..., b_A$ で表される長さ A のビット列は以下のように定義される．

$$BM(p)[i] = \mathrm{b_i} = \begin{cases} 0 & if dist(a_i, p) \leqq r_i \\ 1 & otherwise \end{cases} \quad (1)$$

このビット列 $BM(p)$ をデータ点 p のビットマップと呼ぶ．ビットマップを使用して，d 次元データ空間 R_d を 2^A 個の部分空間に分割できる．すなわち，$BM(p) = BM(q)$ が成り立つ場合，2 つの異なるデータ点 p と q が同じ部分空間に含まれる．以下では，各部分空間を領域と呼ぶ．領域は，対応するビットマップによって識別できる．例えば，$d = 2$, $A = 3$ の場合，2 次元データ空間は 8 つの領域に分割され，その境界は 3 つの円

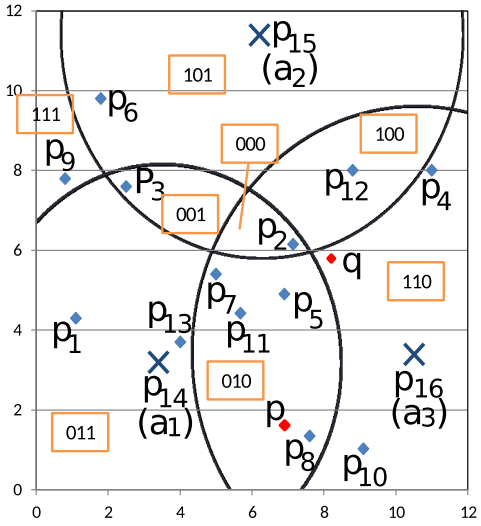

図 1　FDH の一例

で表される．

d 次元データ空間のデータ集合 D に対して，D の各データが対応する領域ごとに管理されている場合は，$BM(q)$ を計算することによって特定のクエリ q の最近傍データを効率的に探索できる．この探索方法は [8] で提案されたもので，Flexible Distance-based Hashing（FDH）と呼ばれる．例えば図 1 では，$p_1, p_2, ..., p_{16}$ の 16 個のデータ点があり，$p_{14} = a_1$, $p_{15} = a_2$，および $p_{16} = a_3$ がアンカーである．データ p が与えられると，そのビットマップは 010 である（$BM(p) = 010$）．ビットマップ 010 のある領域には，p_5, p_7, p_8および p_{11} が含まれる．p とそれらの 4 つのデータ点との間のユークリッド距離を計算し，p に最も近い点として p_8 を見つける．

2 つのデータ点がデータ空間内の近隣に存在する場合，2 つのデータ点が同じ領域に属する（すなわち，それらが同じビットマップを有する）可能性が高いので，FDH は局所的に敏感なハッシングと見なすことができる．FDH の利点は，データ空間の次元数に依存する FDH のパラメータがないため，FDH を用いた最近傍探索はその性能を劣化させることなく任意の次元数に容易に適用できることである．

2.3 FDH を用いた近似最近傍探索

FDH は近似最近傍探索に使用できる．しかしながら，クエリを含む領域が，与えられたクエリに最も近いデータ点を含むという保証がないので，厳密解を見つけることができない可能性がある．この問題点を緩和するために，隣接領域の概念を導入する．2 つのデータ点 p と q が与えられたとき，$Hamming(p, q)$ は p と q のビットマップのハミング距離を表す．例えば，$BM(p) = 011$ および $BM(q) = 110$ であれば，$Hamming(p, q) = 2$ である．$Hamming()$ の表記法を拡張して，$Hamming(p, R)$ が $BM(p)$ と $BM(R)$ のハミング距離を表すようにする．ここで，p はデータ点，R は領域，$BM(R)$ は R のビットマップを表す．

FDH を用いた最近傍探索は以下のように与えられる．正の

整数 H が入力として与えられ，$H \leqq A$ が成り立つと仮定する．ここで，A は FDH のアンカーの数である．そして，与えられたクエリ q，隣接領域の集合 $R_1, R_2, ..., R_k$ は，各 R_i に対して $1 \leqq i \leqq k, BM(q, R_i) \leqq H$ が成り立つように決定される．これらの近傍領域の中から最近傍を検索し，クエリに最も近いデータ点を検索結果として出力する．厳密解が得られる保証はないが，検索の精度は H の値によって制御できる．$H = A$ のとき，この探索アルゴリズムは全探索アルゴリズムと等価であり，得られた結果は厳密解になる．

3 提案アルゴリズム

3.1 FDH を用いた最近傍探索の欠点

2.3 節で説明したように，FDH を用いた近似最近傍探索アルゴリズムは厳密解が見つけられない可能性がある．例えば図 1 でもう一度考えてみる．この図では，3 つのアンカーを含む 16 個のデータ点がある．クエリ q が与えられたとする．q が存在する領域はビットマップ 110 である．以下では，クエリが存在する領域をクエリ領域と呼ぶ．クエリ領域には 3 つのデータ点があるが，いずれも厳密解 p_2 からはほど遠いものである．ハミング距離 H を 1 として指定している場合は，隣接領域のビットマップは，010,100,111 だが，それらの領域には p_2 が含まれていない．H を 2 とすると，p_2 を含む領域が隣接領域に含まれ，厳密解が得られる．ただし，この場合，クエリ領域を含む近接領域の数は 7 であり，1 つを除くすべての領域（領域 001 以外）を探索し最近傍を探す．このように，クエリの最近傍がユークリッド距離的にはクエリの近くに存在しているのにも関わらず，H を大きくしない限り，FDH では見つけられない場合があるのが，FDH を使用した近似最近傍探索の欠点である．この欠点を解決するために，本稿では新たに 2 つのメカニズム，すなわち最遠 δ 近傍の導入による探索領域の拡張と，アンカー集合の複数化を FDH に導入した．これらについては 3.2 節と 3.3 節でそれぞれ説明する．

3.2 FDH の探索領域の拡張

3.2.1 最遠 δ 近傍

図 1 で $H = 1$ の時に，クエリ q に対する正確な解 p_2 が見つけられない理由は，クエリ q が 2 つのアンカー a_1 および a_2 の半径によって定義される球面（円）の非常に近くに存在するからである．この場合，ビットマップ 000 を有する領域は，ハミング距離においてクエリ領域 110 から遠い（クエリから領域までのハミング距離は 2 である）が，ユークリッド距離においては，ビットマップ 000 を有する領域はクエリに近い．この欠点を解決するための方法は，図中の 000 のような領域を，FDH を使用した探索アルゴリズムで探索される隣接領域に追加することである．

クエリ q と 1.0 未満の正の実数で表される δ で定義される最遠 δ 近傍を FDH に新たに導入する．最遠 δ 近傍は，ユークリッド距離的には近いが，δ で制約される領域集合の中でハミング距離においては最も遠い領域である．最遠 δ 近傍 $FN(q, \delta)$ は以下のように定義される．q のビットマップを $BM(q) = b_1, b_2, ..., b_A$ とする．各アンカー a_i，$1 \leqq i \leqq A$ に対して，ビットマップ $b'_1, b'_2, ..., b'_A$ は次のように定義される．

$$if(dist(a_i, q) \leqq r_i) \wedge (dist(a_i, q) \geqq (1-\delta)r_i) then\ b'_i = 1$$
$$if(dist(a_i, q) > r_i) \wedge (dist(a_i, q) \leqq (1+\delta)r_i) then\ b'_i = 0$$
$$otherwise\ b'_i = b_i$$

ここで，r_i はアンカー a_i に対応する半径で，$dist(a_i, q)$ は a_i と q の間のユークリッド距離を表す．次に，q の最遠 δ 近傍，$FN(q, \delta)$ は，ビットマップが上記のように $b'_1, b'_2, ..., b'_A$ である領域である．（つまり，$BM(FN(q, \delta)) = b'_1, b'_2, ..., b'_A$）．例えば，図 1 において，$\delta = 0.1$ とし，クエリ q を与える．$BM(q) = 110$ である．最遠 δ 近傍のビットマップの定義によれば，$b'_1 b'_2 b'_3 = 000$ である．したがって，$FN(q, \delta)$ はビットマップ 000 の領域である．

最遠 δ 近傍を導入することで，FDH に基づく近似最近傍探索アルゴリズムは以下のように変更される．クエリ q，H および δ が与えられると，最初に $NR(q, H)$ で示される近接領域の集合を計算する．これは FDH を用いた元の探索アルゴルズムと同じように定義される．すなわち，$NR(q, H)$ は形式的には次のように定義される．

$$NR(q, H) = \{R | R \in R_{all}, Hamming(q, R) \leqq H\}$$

ここで，R_{all} はデータ空間内のすべての領域の集合である．例えば，図 1 において，q および $H = 1$ の場合，$NR(q, H) = \{110, 010, 100, 111\}$ であり，領域はそれらの対応するビットマップによって表される．説明したように，$H = A$ の場合，$NR(q, H)$ は，データ空間の全領域を表し，アルゴリズムは全探索になる．

次に最遠 δ 近傍に基づいて，以下の式で与えられる $FNR(q, \delta)$ で表される追加の近傍領域を導入する．

$$\begin{aligned} &FNR(q, \delta) \\ &= \{R | R \in R_{all}, Hamming(q, R) + Hamming(R, F) \\ &\leqq Hamming(q, F)\} \end{aligned}$$

ここで，F は q の最遠 δ 近傍である．つまり，$F = FN(q, \delta)$ である．この定義の不等式は，直観的には $FNR(q, \delta)$ の要素である領域 R は q と F の間のどこかに存在するが，F を超えては存在しないということを示している．図 2 は，$NR(q, H)$ および $FNR(q, \delta)$ の概略図を示す．

例えば，図 1 において，q および$\delta = 0.1, F = 000$ および $FNR(q, \delta) = \{000, 010, 100, 110\}$ である．定義の不等式は成り立たないので，他の領域にはこのセットは含まれない．FNR の領域数 (q, δ) は次のように表される．

$$|FNR(q, \delta)| = 2^{Hamming(q,F)}$$

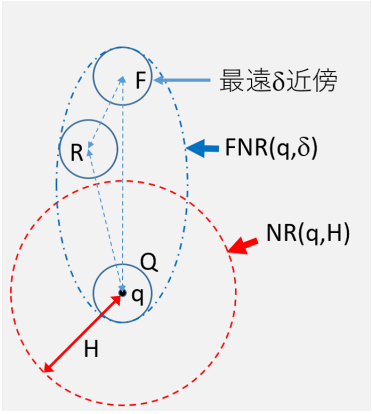

図 2　$NR(q,H)$ および $FNR(q,\delta)$ の概略図

ここで，F は q の最遠 δ 近傍である．

$FNR(q,\delta)$ に基づいて FDH に基づく探索アルゴリズムを次の 2 通りに拡張する．1 つは，最近傍の候補を $NR(q,H) \cup FNR(F,H)$ で探索するアルゴリズムであり，これを δ-FDH という．ただし，$F = FNR(q,\delta)$ である．$FNR(F,H)$ は最遠 δ 近傍からハミング距離 H 以内の領域を表す．δ-FDH はクエリの存在する領域と，クエリの最遠 δ 近傍 F から，それぞれハミング距離 H 以内の領域を探索領域とする探索手法である．

δ-FDH は δ の値が大きすぎる場合，最遠 δ 近傍とクエリ領域の間に未探索の領域が生じる可能性があるという問題点を持つ．この問題点を解消するため，もう 1 つの拡張した探索アルゴリズムを提案する．この探索アルゴリズムでは，最近傍の候補を $NR(q,H) \cup FNR(q,\delta)$ で探索する．この探索アルゴリズムを Extended FDH（E-FDH）という．$NR(q,H)$ と $FNR(q,\delta)$ の両方に属する領域がいくつかある．図 1 の場合，$\{000, 010, 100, 110, 111\}$ の領域が探索される．この例では，厳密解 p_2 を求めることができる．元の探索アルゴリズムでも $H = 2$ の場合同じ結果が得られる．しかし，この場合 7 つの領域が探索されるため，E-FDH よりも長くなる．

3.2.2　δ の適応制御

前節で提案した探索アルゴリズム E-FDH は，FDH を使用した元の探索アルゴリズムより探索精度の点で優れている．しかしながら，改良されたアルゴリズムの問題がまだ 1 つある．つまり，アルゴリズムの有効性はパラメータの値によって異なる．このアルゴリズムには，H とδ の 2 つのパラメータがある．予備実験は，パラメータ値が変化するにつれて結果の精度および計算時間が変化することを示している．実験結果は一般的に H およびδ が大きな値に設定されると結果の精度が大きくなるが，より多くの計算時間を必要とすることを示している．与えられたデータ集合とクエリに対して，事前に最適なパラメータ値を設定することは一般に困難である．

この問題を解決するために，探索中に δ の値を適応制御する方法を導入する．パラメータ値の適応制御の概要は以下の通りである．アルゴリズムが開始すると，δ は 0 に設定される．これは，アルゴリズムの動作が，FDH を使用した元の探索アルゴリズムと同じであることを意味する．アルゴリズムが終了した後，得られた結果は一時的に保存される．それから，δ の値は Δ で示される所定の値に設定され，そしてアルゴリズムは再び実行される．Δ は 1.0 未満の正の実数とする．アルゴリズムが終了した後，得られた結果が改善されるならば，$\delta \leftarrow \delta + \Delta$ として更新し，アルゴリズムは再び実行される．そうでなければ，アルゴリズムは終了する．このように変更した FDH を Adaptive FDH（A-FDH）と呼ぶ．4 章で示すように，実験結果は提案方法が良い解を得るために非常にうまくいくことを示している．

3.2.3　A-FDH の詳細

最遠 δ 近傍及び δ の値の適応制御を含む，近似最近傍探索アルゴリズム A-FDH を以下に示す．

Inputs　クエリ：q，データセット：$D = \{p_1, p_2, ..., p_n\}$，アンカーセット：$A_n = \{a_1, a_2, ..., a_A\}$，半径のセット $R_a = \{r_1, r_2, ..., r_A\}$，最大ハミング距離 H，δ の増加値：Δ.
Step1　$BM(q)$ を計算する．
Step2　$\delta \leftarrow 0$，$SR \leftarrow NR(q,H)$, $min_d \leftarrow \infty$.
Step3　$old_d \leftarrow min_d$.
Step4　各 $q \in SR$ について，
Step4-1　$dist(q,p) < min_d$の場合，$p_{min} \leftarrow p$, $min_d \leftarrow dist(q,p)$
Step4-2　SR にデータ点が残っていない場合は Step 5 に進み，そうでなければ Step4 へ．
Step5　$old_d = min_d$の時, p_{min} を出力したら終了．
Step6　$\delta \leftarrow \delta + \Delta$
Step7　$SR \leftarrow$ {δ の増分で新たに加えられた FNR 内の領域 (q,δ)}，Step3 へ進む．

A-FDH を用いる提案された最近傍探索アルゴリズムの主な利点は，パラメータ値を前もって決定する必要がないことである．第 4 章の実験結果に示すように H とΔ は上に示したアルゴリズムの入力に記載されているが，実際にはそれらの値は前もって固定することができ，満足な結果を得ることができる．

3.3　複数アンカー集合による近傍探索

3.3.1　マルチアンカー FDH

本稿では FDH のもう 1 つの改良として，マルチアンカー FDH（Multi-Anchor FDH，MA-FDH）を提案する．通常の FDH が 1 個のアンカー集合に基づいて多次元空間を複数の領域に分割し，与えられたクエリに近い領域を探索領域として，最近傍探索を行うのに対し，MA-FDH では，複数のアンカー集合を用意し，アンカー集合ごとに多次元空間を領域に分割し，与えられたクエリに対して，アンカー集合ごとのそれぞれの近傍領域において，最近傍探索を行うことで，1 つのアンカー集合を用いる場合よりもよりよい近似解を得ようとするものである．

MA-FDH のアンカー集合の集合を $MA = \{A_1, A_2, \ldots, A_s\}$ とする．ただし，各 A_i は A 個のアンカーから構成されるアンカー集合である．MA-FDH の近傍解探索手順を以下に示す．ただし，H は FDH において探索領域を決定するハミング距離

の上限とする．

MA-FDH は通常の FDH を s 回実行して，得られた近似解の中でクエリに最も近いデータを出力することと等価なので，通常の FDH と MA-FDH が同じハミング距離 H で実行される場合は MA-FDH の近似解が FDH と同じか優れた解が得られることは自明であるが，一方で，計算時間は MA-FDH のほうが増加する．

計算時間の短縮のため，MA-FDH ではデータ集合の各データにフラグを付加する．クエリに対して 1 度距離計算を行ったデータについては，フラグをクエリの識別番号に設定することで，同じクエリに対しては距離計算を 1 度だけ行う．

4 章で示す実験において，MA-FDH のハミング距離 H を適切に設定すれば，同程度の計算時間で，MA-FDH のほうが通常の FDH よりよい解が得られることを示す．

3.3.2 複数アンカー集合の最適設定

前節で提案した MA-FDH においては，複数のアンカー集合をどのように設定するかで得られる近似解の精度が異なる．サンプルデータを用いた複数アンカー集合の最適設定方法を以下に提案する．

MA-FDH で用いる複数アンカー集合のアンカー集合数を s とする．最初に s より十分に大きな S（例えば，$s = 10$ に対して，$S = 100$）個のアンカー集合をランダムに生成する．次に，サンプルデータとして使うクエリデータを Q 個用意する（例えば，$Q = 10000$）．各サンプルデータに対し，S 個のアンカー集合のそれぞれを単独にアンカー集合として実行する通常の FDH を実行し，全探索による厳密解と比較し，正解を得たクエリを各 FDH ごとに記録する．全サンプルデータに対する FDH の実行の終了後，正解を得たクエリデータが最多となるように，グリーディに FDH を s 個選択し，MA-FDH の複数アンカー集合とする．手順の詳細を以下に示す．

まず，最も正解を得たアンカー集合を選択し，MA-FDH のアンカー集合とする．選択されたアンカー集合が正解したクエリを除いた上で，残りのクエリに対して最も正解を得たアンカー集合を選択し，MA-FDH のアンカー集合とする．この操作を s 個のアンカー集合が選択されるまで繰り返す．

4 実験的評価

4.1 評価基準

前章で提案した探索アルゴリズムを評価するための実験をいくつか示す．アルゴリズムの評価基準を以下に示す．クエリ q が与えられたとき，p_{algo} を探索アルゴリズムによって近似最近傍として取得されたデータ点とし，p_{opt} を q の正確な最近傍とする．探索の質を評価するために，絶対誤差 $AE(q, p_{algo}, p_{opt})$ と相対誤差 $RE(q, p_{algo}, p_{opt})$ の 2 つの尺度を定義する．絶対誤差は次のように定義する．

$$AE(q, p_{algo}, p_{opt}) = dist(p_{algo}, q) - dist(p_{opt}, q) \qquad (2)$$

絶対誤差が 0 の場合，相対誤差は 0 である．絶対誤差が 0 でない場合，相対誤差は次のように定義する．

$$RE(q, p_{algo}, p_{opt}) = (AE(q, p_{algo}, p_{opt})/dist(p_{opt}, q)) \times 100(\%) \qquad (3)$$

検索の評価には別の尺度を使用する．最近傍探索が N 回実行され，M 回で正確な最近傍探索が行われると考える $(M \leqq N)$．そして，正答率（Accuracy Rate，AR で表す）は，$AR = (M/N) \times 100(\%)$ と定義される．正確な解は全探索に基づく最近傍探索プログラムで得られるものとする．

探索プログラムの入力として，データ集合を一様乱数で生成する．データ集合は 10,000 個のデータ点で構成され 32，64，128 個の属性（次元）を持つ．属性値は，(-999.99，999.99）の範囲内の実数である．

4.2 実験結果

4.2.1 パラメータ δ 導入の評価

最初に，既存の FDH，FDH に最遠 δ 近傍からハミング距離 H の領域 $NR(F, H)$ を探索領域に加えた δ-FDH，FDH に追加する探索領域を $FNR(q, \delta)$ とする Extended FDH（E-FDH），および δ の適応制御を加えた Adaptive FDH（A-FDH）の 4 角探索手法を比較した実験結果を示す．実験では，アンカーの数 A は 10 に設定され，探索される領域を決定するために探索に使用されるハミング距離 H は 5 までの実験結果を示す．各データ集合の最近傍探索の総数 N は 100,000 である．クエリも一様乱数でランダムに生成される．

属性数 128 の場合の実験結果を図 3,4,6 と表 1 に示す．図 3 と図 4 は δFDH の δ の値を変化させたときの実験結果と FDH の比較を示している．図 6 は E-FDH の δ の値を変化させたときの実験結果と FDH の比較を示している．表 1 は E-FDH と，パラメータ δ を適応的に決定するように変更した場合の A-FDH の実験結果を示している．

図 3 では，▲のプロット点が FDH の結果，◆，●，■はそれぞれ $\delta = 0.01, 0.02, 0.03$ に設定したときの δ-FDH の結果を表している．図 3 から読み取れることとして，パラメータ δ を導入し，最遠 δ 近傍から更に探索領域を増やすことにより，FDH では得ることが出来なかった $H = 2$ と $H = 3$ の間の結果をパラメータ δ の変更により柔軟に得ることが出来るようになったことがわかる．

次に δ の値を 0.1 にしたときの δ-FDH の結果を図 4 に示す．結果を比較しやすくするため FDH のプロット点を平滑線で結んでいる．図 4 より，$\delta = 0.1$ の場合，FDH より相対誤差平均が大きくなり，結果が悪くなっていることがわかる．理由は，図 5 のように，最遠 δ 近傍がクエリから遠くに設定され，探索範囲の間に探索されない領域が発生したためであると推測される．この現象が起こらないように探索範囲を変更した E-FDH の結果を図 6 に示す．

図 6 では，▲のプロット点が FDH の結果，◆，●，■はそれぞれ $\delta = 0.01, 0.05, 0.1$ に設定したときの E-FDH の結果を表している．図 6 から読み取れることとして，図 4 のように

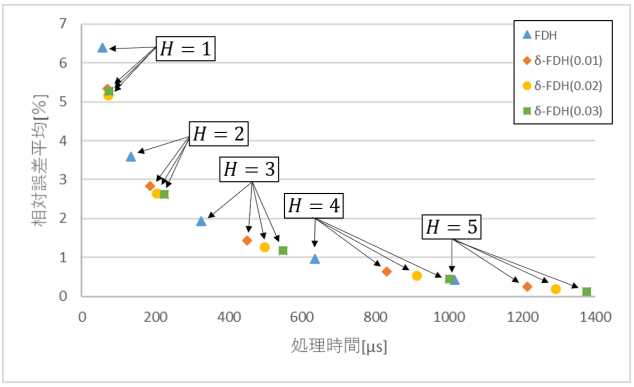

図 3　FDH と δFDH の比較（その 1）

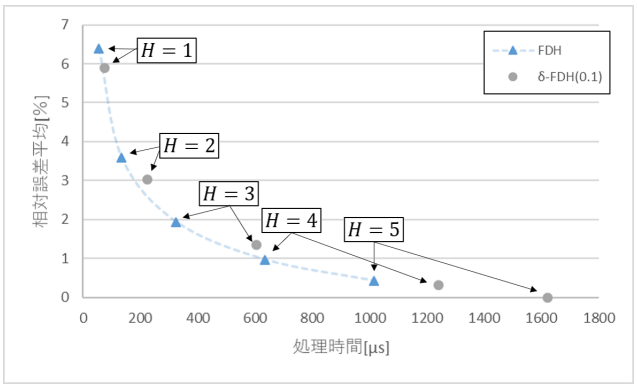

図 4　FDH と δFDH の比較（その 2）

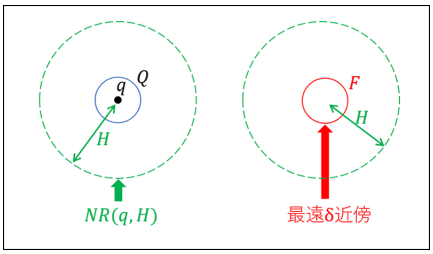

図 5　δ-FDH の探索範囲（ハミング距離空間での模式図）

δ の値を大きくすると，処理時間が一気に増え相対誤差平均は非常に小さくなってはいるものの，FDH より結果が悪くなっていないことがわかる．$\delta = 0.05$ の $H = 1$ の時は FDH より悪くなっているが，これは単にクエリが存在する領域の周りを $H = 1$ の範囲の近傍領域しか探索しておらず，より良い解を得るためにはクエリが存在する領域の周りはもう少し探索する必要があるためである．

表 1 に，E-FDH と A-FDH の結果を示す．$H = 1$ と 2 の結果を見ると，A-FDH の方が E-FDH より処理時間は多くかかっているものの，1 つのクエリに対して適応的に δ の値を決定しているために，相対誤差平均が小さくなっていることがわかる．

4. 2. 2　MA-FDH の評価

次に，マルチアンカー集合 FDH（MA-FDH）の実験結果を示す．次元数 128 のデータ集合を用いたときのアンカー集合数 5,10,20 の場合について，通常の FDH と MA-FDH の正答率，相対誤差平均を比較した結果を図 7，8 に示す．横軸は処理時間で 100,000 個のクエリに対する探索時間の合計，縦軸は

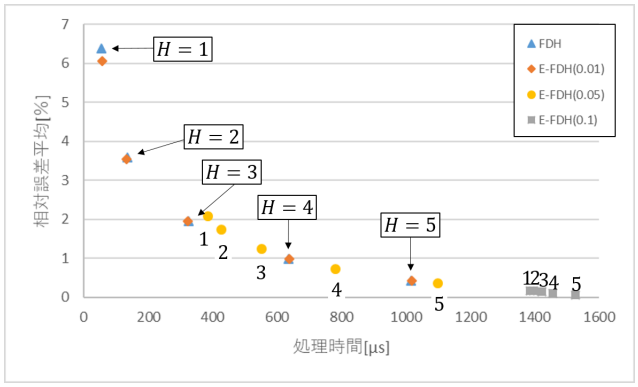

図 6　δ-FDH と EFDH の比較

表 1　E-FDH と A-FDH の結果比較

	δ/Δ	H	相対誤差 平均 [%]	相対誤差 最大 [%]	処理時間 [μs]
E-FDH	0.01 (δ)	1	6.07	27.19	58.1
		2	3.55	22.94	132.9
		3	1.94	21.35	323.4
A-FDH	0.01 (Δ)	1	5.88	27.19	70.6
		2	3.54	22.94	141.2
		3	1.94	21.35	332.6

図 7 が正答率，図 8 が 100,000 個のクエリに対する相対誤差の平均，図中の数字は FDH の探索領域を定めるハミング距離 H を表す．

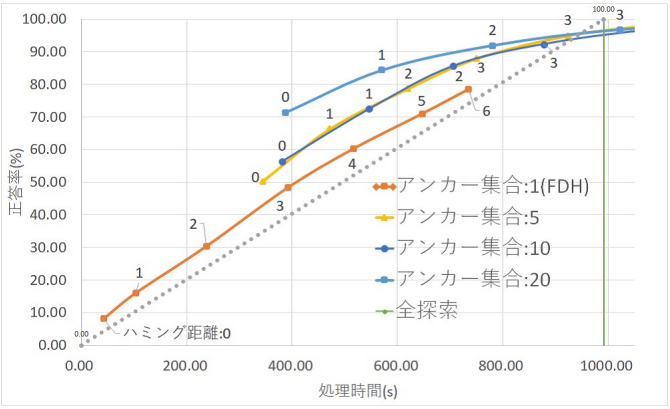

図 7　正答率の比較（属性数 128）

ハミング距離が 2 より大きい場合には同じ処理時間でも，どのアンカー集合数についても MA-FDH のほうが通常の FDH と比べて正答率が高くなることがわかる．また相対誤差平均についても MA-FDH のほうが下回っているため，アンカー集合を複数にすることの有効性が確認できる．また，アンカー集合数が 10 の場合とアンカー集合数が 5 の場合との差があまりないのは，今回，実験の対象としたランダムデータ集合は分布に偏りのあるデータであり，データの密度の高い部分空間の数が 10 であったため，それぞれのアンカー集合の中でお互いに似たアンカーを持つ集合ができてしまったため，アンカー集合数が

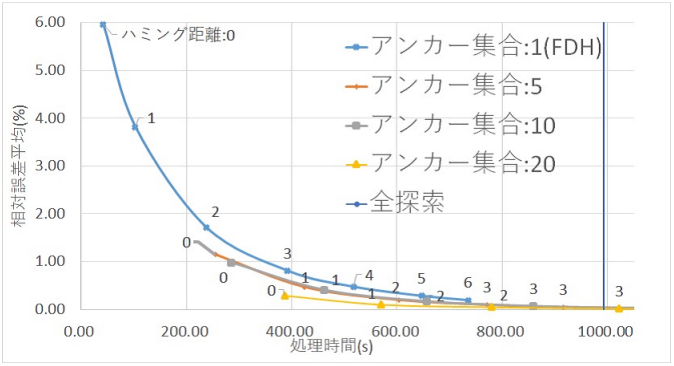

図 8　相対誤差平均の比較（属性数 128）

5 の場合とあまり変わらない結果となってしまったのではないかと考えられる．

属性数 (次元数)64 のデータ集合を用いた場合のアンカー集合数 5，10 の場合について，ランダムに選択した場合と最適に選択した場合のそれぞれでアンカー集合を選んだ MA-FDH の正答率と相対誤差平均を比較した結果を図 9，10 に示す．横軸は処理時間で 100,000 個のクエリに対する探索時間の合計，縦軸は図 9 は正答率，図 10 は 100,000 個のクエリに対する相対誤差の平均，図中の数字は FDH の探索領域を定めるハミング距離 H を表す．

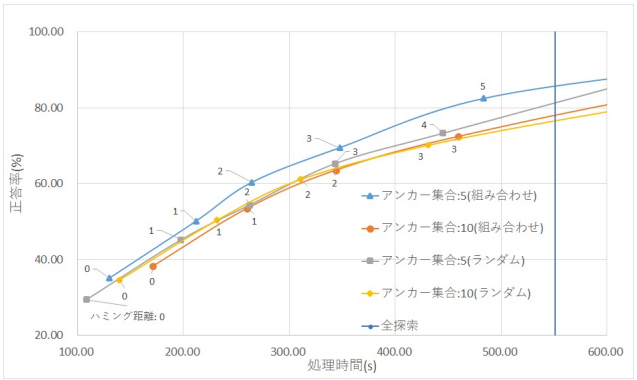

図 9　正答率の比較（属性数 64）

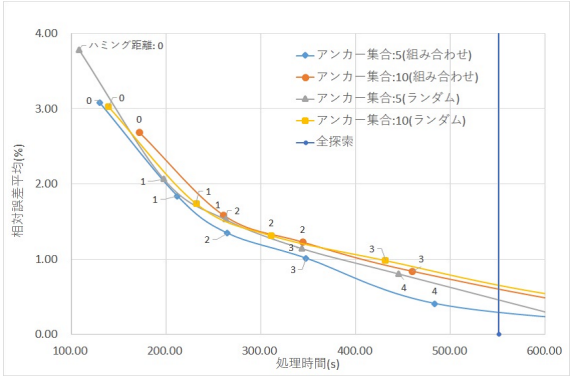

図 10　相対誤差平均の比較（属性数 64）

アンカー集合をランダムに選択したものよりも最適に選択したほうが正答率が高くなり，相対誤差平均も下回ることがわかった．

属性数 (次元数)32, 64, 128 の場合，クエリとの距離計算においてフラグを用いて重複する距離計算を省いた場合と，フラグを用いず，毎回距離計算を行った場合の MA-FDH の処理時間の比較を図 11,12,13 に示す．縦軸は処理時間で 100,000 個のクエリに対する探索時間の合計，横軸は FDH の探索領域を定めるハミング距離 H を表す．

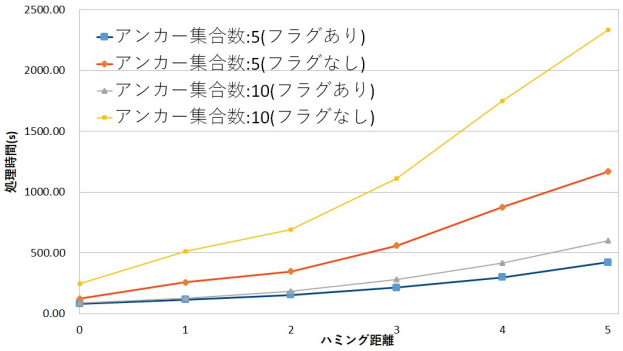

図 11　フラグの有無による処理時間の比較（属性数 32）

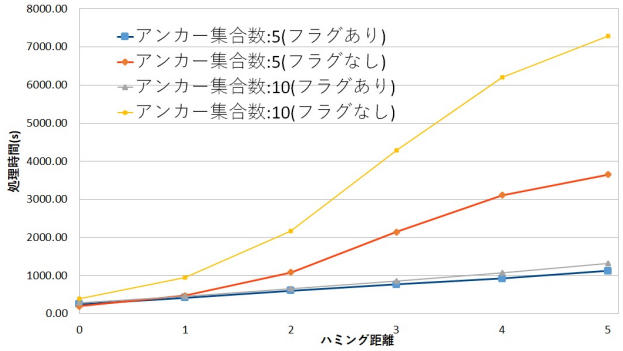

図 12　フラグの有無による処理時間の比較（属性数 64）

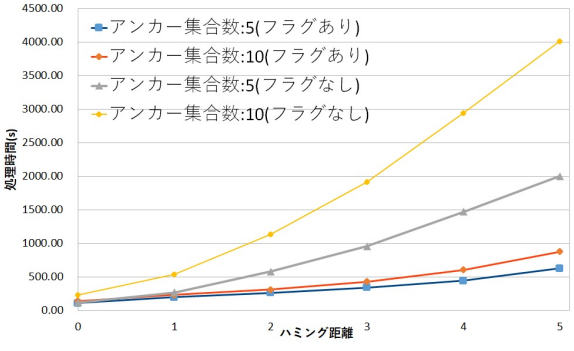

図 13　フラグの有無による処理時間の比較（属性数 128）

フラグを用いた場合と用いなかった場合で同じアンカー集合数でハミング距離が 0 のときは，両者の場合で探索数にほとんど違いがないため計算時間に差がないが，ハミング距離を大きくするにつれて，特にアンカー集合数が 10 の場合については 7 培近く処理時間に差が出ることから，クエリとデータ点間の距離計算が全体の処理時間に大きく影響しており，フラグ導入の有効性が確認された．

5 おわりに

本論文では，大規模多次元データのためのハッシングベースの近似最近傍探索アルゴリズムを提案した．提案探索アルゴリズムは距離ベースのハッシング手法（FDH）に基づいている．本論文では，FDH に基づく最近傍探索手法に対し，２通りの改良手法を提案した．１つ目の改良手法は，最遠 δ 近傍を導入することによって FDH の探索領域を拡張した（δ-FDH，Extended FDH，Adaptive FDH）．さらに，アンカー集合を複数個用いることで，計算時間の増加を抑え，解の精度の向上を図る手法を提案した．実験は，提案した２通りの改良探索アルゴリズムが良い近似解を得るのに有効であることを示した．

FDH を δ-FDH，E-FDH に拡張することで，処理時間と探索精度のより柔軟な制御が可能になった．A-FDH は，アルゴリズムのパラメータの調整をしなくても良い探索結果を得ることができる．また，複数アンカー集合を導入することで，計算時間の増加を招くことなく解探索の探索精度の向上が実現できた．

今後の課題として，より短い計算時間でより良い近似結果を得ることができるように，FDH を用いた探索方法のさらなる改良が望まれる．提案されたアルゴルズムを既存の最近傍アルゴリズムと比較することは興味深く重要である．現実世界のアプリケーションにおいて提案されたアルゴルズムの有効性を示すためには，実験データとしてベンチマークデータを使用する実験が必要となる．δ-FDH，E-FDH，A-FDH に複数アンカー集合を組み合わせることも重要である．さらに，アルゴルズムを k 最近傍アルゴリズムに拡張することも魅力的である．

本研究の一部は科学研究費補助金基盤研究（C）17K001881による．

文　献

[1] Vassills Athitsos, Michalis Potamias, Panagiotis Papapetrou, George Kollios, “Nearest neighbor retrieval using distance-based hashing,” *Proc. IEEE International Conference on Data Engineering*, pp.327–336, 2008.

[2] Gérard Biau, Luc Devroye, *Lectures on the Nearest Neighbor Method*, Springer, 2015.

[3] Junfeng He, Regunathan Radhakrishnan, Shih-Fu Chang, Claus Bauer, “Compact hashing with joint optimization of search accuracy and time,” *Proc. 2011 IEEE Conference on Computer Vision and Pattern Recognition*, pp.753–760, 2011.

[4] Alexander Hinneburg, Charu C. Aggarwal, Daniel A. Keim, “What is the nearest neighbor in high dimensional spaces?,” *Proc. 26th VLDB Conference*, pp.506–515, 2000.

[5] Yoonho Hwang, Bohyung Han, Hee-Kap Ahn, “A fast nearest neighbor search algorithm by nonlinear embedding,” *Proc. 2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp.3053–3060, 2012.

[6] Marius Muja, David G. Lowe, “Scalable nearest neighbor algorithms for high dimensional data,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol.36, No.11, pp.2227–2240, 2014.

[7] Peter N. Yianilos, “Data structures and algorithms for nearest neighbor search in general metric spaces,” *Proc. Fourth annual ACM-SIAM symposium on Discrete algorithms*, pp.311–321, 1993.

[8] Man Lung Yiu, Ira Assent, Christian S. Jensen, and Panos Kalnis, “Outsourced similarity search on metric data assets,” *IEEE Trans. Knowledge and Data Engineering*, Vol.24, No.2, pp.338–352, 2012.