縮小構造Sketchの最適化における局所探索法と焼きなまし法の比較

島内 優介[1,a)]　下川 航平[1]　樋口 直哉[1]　篠原 武[1]

概要：多次元データの近似検索のための縮小構造 Sketch について考察する．Sketch は，局所性鋭敏型ハッシュ (locality sensitive hash) の一種であり，基礎分割関数を用いて作成され，実空間での類似性をある程度保持するようにオブジェクトをバイナリ文字列で表したものである．基礎分割関数はデータを均等に分割することで検索精度が良くなることが知られている．本論文では，Sketch の最適化において，局所探索法と焼きなまし法の 2 種類の手法の有効性の違いについて比較および検証を行った．

キーワード：近似検索，縮小構造 Sketch，量子化球面分割，局所探索，焼きなまし法

Comparison of Local Search Method and Annealing Method in Optimization of Reduced Structure Sketch

Yusuke Shimauchi[1,a)]　Kohei Shimokawa[1]　Naoya Higuchi[1]　Takeshi Shinohara[1]

Abstract: Consider the reduced structure Sketch for approximate retrieval of multidimensional data. Sketch is a kind of locality sensitive hash, it is created by using basic division function and expresses objects in binary character string so as to hold similarity in real space to some extent is there. It is known that the basic division function improves search accuracy by equally dividing data. In this paper, in the optimization of Sketch, we compared and verified the effectiveness difference between the two methods, local search method and simulated annealing method.

Keywords: approximate search, reduced structure Sketch, quantized spherical segmentation, local search, annealing method

1. はじめに

計算機の演算能力や記憶容量の向上により，大量のマルチメディアデータを用いたシステムが多く作られている．そのため，膨大なデータの中から必要なデータだけを高速に探し出す情報検索技術が重要である．マルチメディアデータは，多くの場合かなり高次元であり，また劣化や加工も多く，完全一致検索を行うことは難しい．そのため，それらのデータの高速な検索を実現するために，近似検索を用いることが一般的である．近似検索の一般的な検索手法として，R-tree[1], [2] や M-tree[3] などの索引構造を用いるものがある．それらの空間索引は，空間の次元が大きくなると次元の呪いと呼ばれる現象により，検索効率が悪化してしまうことが知られている．そのため，高次元のデータに索引構造を用いる場合，次元縮小を行うことで次元の呪いを緩和する．しかし，空間索引を用いた検索では，データベース内に近似データが存在する質問に対しては非常に高速に検索できる反面，近似データが存在しない質問に対しては検索速度が遅くなる．

過去に次元縮小射影 Simple-Map(S-Map)[4] を用いた R-tree による近似検索が本研究室では提案されているが，本論文では近年考案された検索手法として，高速かつ一定時間で検索を行うことが可能である Sketch[5], [6], [7], [8], [9] を用いる．Sketch は，量子化球面分割 (QBP)[10] 等の基礎

[1] 九州工業大学　情報工学部　知能情報工学科
[a)] o231034y@mail.kyutech.jp

分割関数を用いて作成され，実空間での類似度をある程度保持したまま，オブジェクトをバイナリ文字列で表現したものである．Sketch は実空間上での類似性を完全には保持しないため，Sketch 上での最近傍解が実空間上での最近傍解と等しいとは限らない．そこで Sketch を用いた k 近傍検索では，まず Sketch をフィルタリングとして用いることで $K(>k)$ 個の候補データを取り出す．次に，その K 個の候補データと質問データとの実距離を計算し，k 近傍解を求める．K を大きくすると検索精度は高くなるが，実距離計算コストが増加する．また，Sketch の検索精度（正答率） とは，実空間での k 近傍解が K 個の候補内に存在する確率のことで，基礎分割関数の性能によって決まる．

第 2 章では，近似検索システムと Sketch について紹介する．第 3 章では，Sketch 作成のための基礎分割関数について紹介し，第 4 章で提案を行う．第 5 章で実験を行い，第 6 章でまとめる．

2. 近似検索システム

本章では，本論文で用いる基本的な単語の定義を行う．特に，距離空間，距離関数，近似検索で用いる質問方法，Sketch について紹介する．

2.1 距離空間

近似検索は，質問オブジェクトからの非近似度（距離）により特徴空間内からオブジェクトを取得する過程であると捉えることができる．つまり近似検索は，質問点に対する一種のソーティングや順序付け，ランク付けであり，その基準は距離である．近似検索システムは，質問点との距離が近いオブジェクトを**似ている**と判断し，それらを特徴空間から取り出すシステムのことである．

近似検索システムが対象とするオブジェクトを包括する特徴空間全体を $\mathcal{D} = \mathbb{R}^n$ とする．ここで，$\mathbb{R}$ は実数全体，n は特徴の次元数を表す．索引の対象となる m 個のオブジェクトの集合を $\mathcal{S} = \{O_1, O_2, \ldots, O_i, \ldots, O_m\} \subseteq \mathcal{D}$ とする．O_i は n 次元の特徴 $\{O_i^{(1)}, O_i^{(2)}, \ldots, O_i^{(t)}, \ldots, O_i^{(n)}\}$ を持つ．ここで，$O_i^{(t)} \in \mathbb{R}$ である．任意の 2 点のオブジェクト間の非近似度を示す全域的な**距離関数**を $d : \mathcal{D} \times \mathcal{D} \to \mathbb{R}^+$ とし，$\mathcal{M} = (\mathcal{S}, d)$ を特徴空間と呼ぶこととする．近似検索システムでは，距離関数 d は**距離の公理**である次の条件を満たすものと仮定する．ただし同一性については，場合によって満たさないこともあり，これを疑距離と呼ぶ．

(1) $d(X,Y) \geq 0$ (非負性)
(2) $d(X,Y) = d(Y,X)$ (対称性)
(3) $d(X,Y) \leq d(X,Z) + d(Z,Y)$ (三角不等式)
(4) $d(X,Y) = 0 \Leftrightarrow X = Y$ (同一性)

ここで，$X, Y, Z \in \mathcal{D}$ である．上の条件の中で最も重要なものは，三角不等式である．簡単のために，距離関数は単に**距離**と呼ぶこともある．

近似検索システムは，与えられた質問点 $Q = \{Q^{(1)}, Q^{(2)}, \ldots, Q^{(t)}, \ldots, Q^{(n)}\} \in \mathcal{D}$ に対して近似度が高い（距離が小さい）オブジェクトを結果として与えるものである．

2.2 距離計測

本節では，近似検索で用いる距離計測について説明する．特徴空間内の任意のオブジェクトを x とする．x の特徴は n 組の実数 $(x_{(1)}, x_{(2)}, \ldots, x_{(n)})$ で表される．以下の距離計測関数は距離の公理を満たすものである．

- L_1（マンハッタン）距離 :

$$D(x,y) = \sum_{i=1}^{n} |x_{(i)} - y_{(i)}|$$

- L_2（ユークリッド）距離 :

$$D(x,y) = \sqrt{\sum_{i=1}^{n} (x_{(i)} - y_{(i)})^2}$$

- L_∞距離 :

$$D(x,y) = \max_{i=1}^{n} |x_{(i)} - y_{(i)}|$$

2.3 質問方法

近似検索で用いる質問方法は，質問オブジェクトや要求する非近似度の範囲によって定義される．非近似度の範囲は，一般的に距離によって表される．質問に対する解は，その選択基準を満たすすべてのオブジェクトである．質問により得られた解の集合は順序付けされたリストと捉えることができ，順序付けの基準は質問点からの距離である．

質問方法として，主に範囲質問および近傍質問の 2 種類がある．質問点 Q と距離 $r \in \mathbb{R}^+$ を質問のパラメータとする**範囲質問** $Range(\mathcal{D}, Q, r)$ は，Q から距離 r 以内のオブジェクトを取得する質問である．すなわち，その解の集合は

$$Range(\mathcal{D}, Q, r) = \{O_i \in \mathcal{S} \mid d(O_i, Q) \leq r\}$$

である．特に，r が幅を持たない $(r = 0)$ 範囲質問を点質問 (point query) と呼び，データベースに質問点 Q との距離が 0 となる (特徴が完全に一致する) オブジェクトが存在する場合に限り，それを返す．

範囲質問は最も基本的な質問方法であるが，使用する際に検索範囲を明確に定めなければならない．これは，特徴空間内のオブジェクトの分散や，距離関数についての前提知識がなければ，困難な作業である．もし検索範囲が小さすぎると適切な解を得ることができないので，検索範囲を少しずつ拡大していく等の微調整が必要である．

そこで，質問点 Q から距離が最も小さいオブジェクトを取得する質問 $NN(\mathcal{D}, Q)$ が提案された．これを**最近傍質問**

という．また，最近傍質問を一般化したものとして，質問点 Q から距離が小さい $k(\leq m)$ 個のオブジェクトを取得する質問 $k\text{-}NN(\mathcal{D}, Q, k)$ があり，これを k **近傍質問**という．その解の集合は

$$k\text{-NN}(\mathcal{D}, Q, k) = \{O_i \in \mathcal{S}\}$$

である．もし，特徴空間に含まれるオブジェクトの数 m が k 個よりも少ない場合は，そのすべてのオブジェクトの集合を解として返す．

近傍質問では，初期検索半径を無限大に設定する．検索過程で新たな解候補のオブジェクトが見つかった場合，オブジェクトを暫定解として登録し，検索範囲の収縮を行う．最終的に，検索範囲内にオブジェクトがみつからなくなると，その時点の暫定解を最終的な解として返す手法である．この質問方法は，取得する解の個数のみを指定するため，特徴空間内のオブジェクトや検索に用いる距離関数に関する知識を必要としない．そのため，質問方法として容易に使用することができる．

さらに，これらの質問方法を組み合わせたものとして，**範囲限定** k **近傍質問** $k\text{-}NN\_Range(\mathcal{D}, Q, r, k)$ がある．これは質問点 Q から距離が近い k 個のオブジェクトの中で Q との距離が r 以内のものを取得する．つまり，その解の集合は

$$k\text{-NN\_Range}(\mathcal{D}, Q, r, k) = k\text{-NN}(\mathcal{D}, Q, k) \cap \text{Range}(\mathcal{D}, Q, r)$$

である．特に $k = 1$ のときを**範囲限定最近傍質問**と呼ぶ．範囲限定 k 近傍質問は，一定距離以上離れている解は不必要であるときなどに用いると効果的である．

2.4 Sketch

R-tree などに代表される索引構造を用いた検索は，データベース内によく似たデータが存在する質問 (**近質問**) は非常に高速に検索を行うことができるが，データベース内に似たデータが存在しないような質問 (**遠質問**) は検索効率が悪いという問題がある．そのため，高速かつ一定時間で検索を行うことが可能な空間検索の手法として，Sketch を用いた検索法が考案された．Sketch は**基礎分割関数**を用いてオブジェクト間の距離関係の大部分を保持できるように，オブジェクトをバイナリ文字列で表現したものである．そのため，Sketch 間の距離としてハミング距離を用いることで，実空間上の距離関係を近似することができるという特徴がある．

Sketch を用いた k 近傍質問は，以下の手順で検索を行う．(1) Sketch データベースに対する $K(K \geq k \geq 1)$ 近傍質問 (2) K 個の解から実空間距離に基づき k 近傍の解を返す 最初の段階における Sketch データベースに対する検索では，全探索を用いて K 近傍質問を行う．通常，全探索は索引構造を用いて探索を行った場合と比較して，検索に時間がかかるが，Sketch における全探索では，Sketch が元のデータに比べて小さく圧縮されており，距離を高速に計算することができるため，非常に高速に解を得ることが可能である．次に，得られた解を用いて，実際の距離に基づいた解を生成する．

Sketch を用いた検索では，Sketch がオブジェクト間の距離関係を完全に保持できないため，解の精度が悪くなってしまう．そのため，解の精度を高い水準に維持したまま，検索を高速化するためには，Sketch を作成するための基礎分割関数が重要である．3 章では，任意の距離関数に適用することが可能な基礎分割関数を紹介し，より検索を効率化するための基礎分割関数の提案を行う．

3. 基礎分割関数

本章では，任意の距離空間における Sketch を生成するための基礎分割関数として，量子化球面分割 (BP) を紹介する．基礎分割関数を m 回適用することで長さ m の Sketch が作成できる．QBP では，ピボットと半径を用いて空間を二分割する．

3.1 QBP

QBP は、ランダムに選んだオブジェクトをデータ中央値によって 2 値量子化した点を中心点とし，中心点と中央値座標との距離を半径とする手法である．

オブジェクト集合 $S \in \mathcal{U}$ と中心点 $(p) \in \mathcal{U}$, 中心点と中央値座標との距離である半径 R が与えられたとすると，S は以下のように部分空間 $S_{p_{in}}, S_{p_{out}}$ に分割される．

$$S_{p_{in}} = \{o \in S | d(p, o) \leq R\}$$
$$S_{p_{out}} = \{o \in S | d(p, o) > R\}$$

図 1 は，L_1 空間において集合 $S = \{A, B, C, D, E, F, G, H, I, J\}$ を QBP によって分割した例である．ここで，P はランダムに選ばれたオブジェクト，P' は P を 2 値量子化した中心点，M はデータ中央値座標を示している．

図 1 では, 中心点 P' により，集合 S は，$S_{P'_{in}} = \{A, D, H, I, J\}$ と $S_{P'_{out}} = \{B, C, E, F, G\}$ に分割される．オブジェクト x から基礎分割関数 σ を用いて生成される長さ m の Sketch を $\sigma(x) \in \{1, 0\}^m$ とすると，BP では Sketch の各ビット $\sigma_i(x)$ は中心点 (p_i) を用いて以下のように定義される．

$$\sigma_i(x) = \begin{cases} 0, \text{if } d(p_i, x) \leq R \\ 1, \text{if } d(p_i, x) > R \end{cases} (i = 1, 2, ..., m) \qquad (1)$$

4. 最適化手法

検索精度向上のために，ピボットの選択における最適化が重要である．従来の最適化では，ピボット候補として

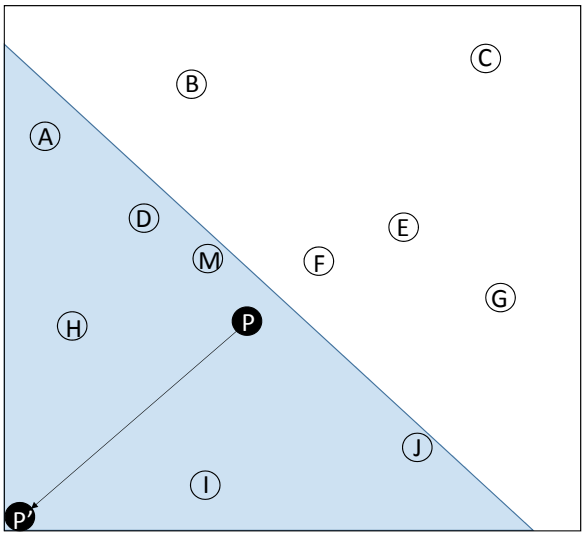

図 1　2 次元 L_1 空間における QBP

データベースからランダムにオブジェクトを取り出し，そのピボット候補を評価関数によって評価する．同様の処理を任意の回数行い，最もスコアの良かったピボット候補をピボットとして決定することで行っていた．ピボットの評価には最小衝突法を用いている．本来，異なるオブジェクトが Sketch 上において同じバイナリ文字列になることを衝突と呼び，この衝突が少ないほど良いピボットと見なす評価手法である．

4.1　局所探索

n 次元のデータに対し，m ビットの Sketch を作成する場合，各ビット毎にピボットがあり，各ピボット毎に n 次元の中心点が存在する．しかし，正規化したデータを用いているため，空間全体に対してデータの存在する範囲は非常に小さく，より適した中心点を見逃している可能性が高い．そこで，局所探索を用いる．局所探索では，中心点の候補として選ばれたオブジェクトを基準に各次元軸の値を少しずつ変化させることで，より良い中心点を探索し検索効率の向上を図っている．中心点候補よりスコアが上回っている点を発見した場合，その点を新たな中心点候補として同様の操作を繰り返す手法である．本論文では，QBP によって作成した Sketch の，2 値量子化した座標値の最小値，最大値を一つだけ反転させる手法 Flip を提案する．

4.2　焼きなまし法（AIR)

焼きなまし法は，大域的最適化問題への汎用の乱択アルゴリズムであり，広大な探索空間内の与えられた関数の大域的最適解に対して，よい近似を与える．従来の焼きなまし法は，温度を用いて高温時にランダムウォーク，徐々に温度が下がるにつれて局所探索を行うものである．

本研究では，AIR[11] を用いる．AIR では，温度の代わりにサンプル数を変化させることで焼きなましを行う．高温時には少ないサンプルを用いることで誤評価を発生させ，従来の焼きなましにおけるランダムウォークを実現しており，低温時には多くのサンプルに対して局所探索を行うことになる．この手法は，高温時には少ないサンプルに対してのみ評価を行うため，従来の焼きなまし法より高速に焼きなますことが可能である．

具体的には，本研究では 64 次元に特徴抽出した各次元 0 から 255 の座標値を持つデータを用い，QBP によって 32 ビットの Sketch を作成することを考える．データベース内からピボットをランダムに 32 個選び，焼きなましを開始する．32 個のピボットから一つを選んで，そのピボットの 64 次元から一つを選び，その量子化された座標値を反転させてスコアが良いものを選ぶ．スコア計算では，一部の値しか変化しないことを利用して，効率よく計算できるように工夫した（計算途中結果の再利用，部分スコアの利用など）．高速化の例として，サンプルを固定するものとピボットを固定するものがあり，REUSE と RETRY を設定した．REUSE は，サンプルを固定する回数であり，RETRY は，ピボットを固定する回数である．ただし，サンプルを取り替えたときには，最適化を行うピボットも取り替えるために，REUSE の約数を RETRY とする．

Algorithm 1 ：REUSE と RETRY

入力： TRIAL : 試行回数
入力： REUSE : 同じサンプルを使用する回数
入力： RETRY : 同じピボットを使用する回数

```
for all TRIAL do
  if TRIAL % REUSE == 0 then
    サンプルの取り替え
  end if
  if TRIAL % RETRY == 0 then
    ピボットの選択
    選択したピボット以外の Sketch による衝突サンプルの検出
  end if
  選択したピボットに対してＦｌｉｐ
  衝突サンプルだけで衝突評価
  if 衝突が悪化しない then
    ピボットを遷移
  end if
end for
```

5.　実験

約 2,900 本の動画から 64 次元に特徴抽出した約 700 万件の画像データに 300 件の質問データを用いて実験を行った．質問データは，近質問，準近質問，遠質問それぞれ 100 件ずつとする．また，m=32 とし，32 ビットの Sketch を作成する．

5.1 局所探索

最適化の試行回数，サンプルサイズ，サンプリングの回数などを変化させ，DB 全体の傾向を確認した．まず，サンプルサイズを DB 全体の約 0.2%にあたる 13600，サンプリング回数を固定し，最適化の試行回数を変化させた．前段階として QBP で各ビット 100 回ずつ試行し，32 ビットの Sketch を作成している．

表 1 局所探索の有効性

試行回数	衝突率 (サンプル)	衝突率 (DB)
1000	1.82×10^{-6}	3.04×10^{-6}
5000	1.23×10^{-6}	2.83×10^{-6}
10000	1.13×10^{-6}	2.81×10^{-6}
20000	1.06×10^{-6}	2.89×10^{-6}
30000	1.04×10^{-6}	2.90×10^{-6}
100000	9.95×10^{-7}	2.88×10^{-6}

表 1 から読み取れるように，サンプルセットの衝突率を下げても DB 全体の衝突率が下がっているわけではないことがわかる．そこでサンプルサイズを DB 全体の約 1.0%，約 2.0%と変化させて実験した．

表 2 サンプルサイズを大きくした効果 (DB の衝突率)

試行回数	約 0.2%)	約 1.0%	約 2.0%
1000	3.04×10^{-6}	2.58×10^{-6}	2.48×10^{-6}
5000	2.83×10^{-6}	2.35×10^{-6}	2.27×10^{-6}
10000	2.81×10^{-6}	2.30×10^{-6}	2.23×10^{-6}
20000	2.89×10^{-6}	2.28×10^{-6}	2.22×10^{-6}
30000	2.90×10^{-6}	2.29×10^{-6}	2.22×10^{-6}
100000	2.88×10^{-6}	2.28×10^{-6}	2.22×10^{-6}

表 2 より，サンプルサイズを大きくすることで DB の衝突率は下がっている．ここで，サンプルサイズを DB の約 2.0 パーセントの 136000，試行回数を 30000 回に固定し，サンプリングの回数を変えて実験した．

表 3 サンプルを取り換えることの有効性

サンプリング回数	衝突率 (DB)	標準偏差 (DB)
1	2.22×10^{-6}	4.98×10^{-8}
6	2.08×10^{-6}	3.10×10^{-8}
50	2.00×10^{-6}	1.82×10^{-8}
100	2.03×10^{-6}	1.07×10^{-8}
300	2.04×10^{-6}	2.54×10^{-8}

表 3 より，サンプリング回数を増やすことで DB の衝突率が安定して下がっているのがわかる．サンプリング回数 300 での標準偏差が大きいのは，一つのサンプルセットに対しての最適化が進んでいないからだと考えられる．

5.2 AIR の有効性

4 章で提案した AIR の有効性を検証する．最適化の試行回数は 10 万回，サンプルサイズは 10 万とする．焼きなまし法を用いた場合と局所探索を用いた場合の DB 全体の衝突率を示す．

表 4 AIR の効果

手法	衝突率	標準偏差	時間
AIR	2.08×10^{-6}	1.74×10^{-8}	28.6 秒
局所探索	2.23×10^{-6}	5.14×10^{-8}	122 秒

表 4 より，AIR の方が局所探索に比べて衝突率が低い Sketch が得られていることがわかる．また，標準偏差も小さく，時間も短い．

5.3 REUSE と RETRY の設定

AIR を用いて，REUSE と RETRY の値を変化させて実験を行った．m=32，K=7,000，k=1 とする．QBP を用いて 32 個のピボットを作成し，その後の Flip の試行回数は 10 万回とした．サンプルサイズは 10 万とし，10 回実験を行った．10 回平均のデータベース全体での衝突率を表 5，処理時間を表 6，正答率を表 7 に示す．

表 5 衝突率（DB）[$\times 10^{-6}$]

RETRY \ REUSE	50	100	200
1	2.06	2.06	2.06
2	2.06	2.07	2.05
25	2.08	2.08	2.08
50	2.10	2.09	2.08
100	-	2.12	2.11
200	-	-	2.13

表 5 より，RETRY が大きいほど衝突率が大きくなった．これは，RETRY が大きいと同じピボットにしか最適化を行えないためだと考えられる．

表 6 処理時間 [秒]

RETRY \ REUSE	50	100	200
1	971	947	931
2	499	484	469
25	49.4	47.9	49.7
50	30.2	29.5	28.6
100	-	20.7	21.2
200	-	-	13.0

表 6 より，毎回ランダムにピボットを選択するより，2 回だけでもピボットを固定することで，大幅に処理時間を

表 7　正答率

RETRY \ REUSE	50	100	200
1	97.1%	97.5%	96.8%
2	96.7%	97.1%	97.0%
25	96.5%	96.9%	97.3%
50	97.1%	96.8%	97.5%
100	-	96.6%	96.8%
200	-	-	96.9%

削減することができた．また，RETRY を大きくするほど，高速になった．

表 7 より，正答率への効果は見られなかった．

6. まとめ

実験結果から，焼きなましの方が局所探索に比べて衝突率が低い Sketch が得られていることが分かる．また，標準偏差も小さく，時間も短い．局所探索を用いて安定して衝突を下げるには，サンプルサイズを大きくする，サンプルセットを時々入れ替えるなどの手法が有効であることが他の実験より分かっているが，AIR ほど高速な探索が実現できない．よって，焼きなましを用いることにより，DB の衝突を安定して下げることができ，高速な最適化が実現できたといえる．しかし，正答率については大きな効果は見られなかった．今後の課題として，正答率を上げるためには，衝突率以外の評価基準が必要であることがあげられる．

参考文献

[1] Guttman, A. : R-trees: A dynamic index structure for spatial searching, Proc. ACM SIGMOD, International Conference on Management of Data, pp. 47-57, (1984).
[2] Navarro, G. : Searching in metric spaces by spatial approximation, The VLDB Journal, pp. 28–46, (2002).
[3] Ciaccia, P., Patella, M., Zezula, P. : M-tree: An ecient access method for similarity search in metric spaces, Proc. 23rd Int. Conf. on Very Large Data Bases, pp. 426-435, (1997).
[4] Shinohara, T., Ishizaka, H. : On dimension reduction mappings for approximate retrieval of multi-dimensional data, Progress in Discovery Science, pp. 224-231, (2002).
[5] Lv, Q. and Josephson, W. and Charikar, M. and Li, K. : Image similarity search with compact data structures, Conference on Information and Knowledge Management archive, Proceedings of the thirteenth ACM international conference on Information and knowledge management”, pp. 208-217, (2004).
[6] Lv, Q. and Josephson, W. and Wang, Z. and Charikar, M. and Li, K. : Ecient ltering with sketches in the ferret toolkit, International Multimedia Conference archive, Proceedings of the 8th ACM international workshop on Multimedia information retrieval, pp. 279-288, (2006).
[7] Jose Muller-Molina, A. and Shinohara, T. : Ecient Similarity Search by Reducing I/O with Compressed Sketches, International Workshop on Similarity Search and Applications archive, Proceedings of the 2009 Second International Workshop on Similarity Search and Applications, pp. 30-38, (2009).
[8] 岩崎 瑶平，篠原 武，A.J.Mulle : Sketch による大容量記憶 データに対する高速な空間検索法に関する研究，第 74 回 SIG-FPAI，(2009)．
[9] Higuchi, N., Imamura, Y., et. al: Nearest neighbor search using sketches as quantized images of dimension reduction, In Proc. ICPRAM’18, pp. 356–363, 2018.
[10] 樋口直哉，焼きなまし法を用いた縮小構造 Sketch の最適化と多次元データ近似検索の高性能化に関する研究，九州工業大学大学院 修士論文，2016．
[11] Imamura, Y., Higuchi N., et. al: Pivot selection for dimension reduction using Annealing by Increasing Resampling, In Proc. LWDA’17, pp. 15–23, 2017.