CD-001 部分属性指定に対応した近似最近傍探索アルゴリズム

植村 玲央
大阪大学
uemura.reon@ist.osaka-u.ac.jp

天方 大地
大阪大学
amagata.daichi@ist.osaka-u.ac.jp

アブストラクト

近年，多くのオブジェクトが高次元ベクトルで表現されるようになっているため，高性能なベクトルデータ管理システムが開発されている．その傾向に伴い，システムの主要な機能の 1 つである近似最近傍探索（ANNS）問題が注目を集めている．本論文では，各オブジェクトが高次元ベクトルと複数の属性を有する状況を想定し，属性制約のもとで ANNS を行う問題に取り組む．この問題は，属性制約を満たすオブジェクトの中から，近似最近傍を探索することを目的とする．この設定は探索が必要な応用において標準的であるが，既存研究では全ての属性に制約があることを想定しているものが多く，部分一致に対応できる手法は少ない．本論文では，部分一致に対応できるアプローチを提案し，実世界データを用いた実験により，提案アルゴリズムの優位性を示す．

1 はじめに

機械学習モデルの普及により，ユーザ，アイテム，画像，およびドキュメント等のオブジェクトが高次元ベクトルとして表現されることが一般的となっている．この傾向に伴い，Milvus [26]，Vearch [1]，Vespa [2]，Weaviate [3]，Pinecone [4]，および ADBV [32] 等のベクトルデータ管理システムの重要性が増加している．近似最近傍探索（ANNS）はこれらのシステムにおける主要な機能の一つであり，多くの研究で局所性鋭敏型ハッシュ [24, 25, 33, 38]，量子化 [13–15, 31]，および近接グラフ [6, 12, 19, 22] 等の ANNS 問題に対する高精度な解を効率的に計算するアルゴリズムが開発されている．（高次元空間における厳密な最近傍探索に対する最速のアルゴリズムは線形スキャンであるため，厳密解の計算は遅い [18].）これらのアルゴリズムは，Microsoft [9, 23]，Apple [20]，および Alibaba [32] 等の多くの企業の検索エンジンで採用されていることが知られている．

近年，柔軟な検索を実現するため，属性制約下での ANNS 問題が注目されている [16, 17, 21, 27, 34, 35, 37]．この問題において，各オブジェクトはベクトルと属性が関連付けられており，クエリの属性制約に一致するオブジェクトの中で最もベクトルが類似するもののみが出力される．この属性制約には (i) クエリ時に全ての属性を属性制約として指定し，すべての属性においてクエリが指定する属性値と一致するオブジェクトのみを候補とする完全属性制約，および (ii) クエリ時に一部の属性を制約として指定し，指定された属性の値がクエリで指定されたものと一致するオブジェクトのみを候補とする部分属性制約の 2 種類があり，実世界の多くのシナリオで想定されている．例えば，論文検索エンジンでは，発表年や会議によるフィルタリングが行われる．また，Amazon.com のような e コマースサイトでは，価格，カテゴリ，および評価によるフィルタリングが可能である．この属性によるフィルタリングはこれらのサービスにおける標準的な実装であることから，完全属性制約および部分属性制約下での ANNS 問題が重要であることが分かる．本論文ではこの問題に取り組む．

1.1 既存手法の課題

属性制約に対応できる基本的な手法として，Pre-filter [26, 32]，Post-filter [36]，および inline processing [16] と呼ばれる 3 つのアプローチが存在する．Pre-filter は，まず指定された属性制約に一致する全てのオブジェクトを取得し，その後，これらのオブジェクトの集合を線形スキャンする．このアプローチは正確な解を得られるものの，候補となるオブジェクトの数が小さい場合にのみ効率的である．Post-filter は，オフラインで ANNS インデックスを構築し，クエリが与えられたときにこのインデックスに関連する ANNS アルゴリズムを実行し，検索結果から属性制約を満たさないオブジェクトを除外する．この結果の集合サイズが k に満たない場合は，探索範囲を広げるために $k' > k$ を設定し，ANNS を追加で実行する．これは，属性制約を満たす k 個のオブジェクトが得られるまで繰り返される．このアプローチは ANNS を複数回実行する可能性があり，（非常に）大きな k' を設定しなければならない点で効率的ではない．この非効率性を緩和するため，inline processing では，インデックスを走査し，アクセスされたオブジェクトが属性制約に一致する場合のみ距離を計算する．しかし，inline processing でも多くの不要なオブジェクトにアクセスするため，計算コストが大きくなってしまう．

属性制約下での ANNS 問題に取り組んだ手法として，Filtered-DiskANN [16]，NHQ [27]，および CAPS [17] 等が存在する．Filtered-DiskANN および NHQ はどちらも近接グラフを利用したアルゴリズムである．Filtered-DiskANN はインデックス構築時にオブジェクトのベクトル間の類似性だけでなく，属性集合も考慮してグラフ構造を構築することで，探索効率を向上させている．しかし，オブジェクトが単一属性しか持たない場合のアルゴリズムしか紹介されていない．NHQ は属性値間の距離を定義し，オブジェクト間でのベクトル間距離と属性値間距離を加算した合成距離をもとに近接グラフ構築を行うアルゴリズムである．しかし，本アルゴリズムでの属性間距離は完全属性制約を前提とした距離であるため，部分属性制約には対応していない．CAPS は，Pre-filter と inline processing のハイブリッドアプローチを採用しており，属性値の出現頻度に基づいて決定木を構築する手法である．具体的には，決定木のノードを分割する際，最も頻繁に出現する属性値が分岐条件として決定される．これにより，決定木の各ノードは分岐条件を満たすオブジェクトを含むことになる．しかし，CAPS も完全属性制約を前提としており，指定していない属性が分岐条件として設定されていた場合が全探索に帰着する．このように属性制約下での ANNS 問題に取り組んだ既存研究は，単一属性もしくは完全属性制約を前提としており，複数かつ部分属性制約下での ANNS 問題には対応していない．

1.2 貢献

上記のように，既存研究には部分属性制約への非対応という制限や高い計算コストの課題がある．本論文はこれらの課題を解決し，任意の部分属性制約に対して柔軟に対応できる新たな近接グラフを提案する．この近接グラフを用いて，提案アルゴリズムは Pre-filter，Post-filter，および inline-processing 等のアルゴリズムよりも属性制約を満たさないオブジェクトへのアクセス（属性一致チェックも含む）回数および属性制約を満たすオブジェクトとの距離計算回数を大幅に削減する．

本論文の貢献は以下の通りである．

- 部分属性制約に対応できる高速かつ高精度な ANNS アルゴリズムを提案する．

- 実世界のデータセットを用いた実験を行い，提案アルゴリズムが非常に高速であり，既存アルゴリズムを上回る性能であることを示した．

1.3 構成

本論文の構成は以下の通りである．2章で本論文における問題を定義する．3章で高速かつ高精度なアルゴリズムを提案する．4章で実験結果を報告する．5章で関連研究を紹介し，6章で本論文の結論を述べる．

2 前提

O を n 個のオブジェクトの集合とし，各オブジェクト $o \in O$ は d 次元ベクトル $\in \mathbb{R}^d$ および m 個の属性に関連付けられているものとする．（例えば，$m = 2$ の場合はカテゴリと色といった属性が考えられる．）ここで $d = O(1)$ は十分に大きく，O は静的かつ RAM 上に存在するものと想定する [10, 12, 17, 19, 21, 22, 27, 34, 37]．また，$o.v$ は o のベクトル，$o.A$ は o に付与されている属性値のリストであり，$o.A_i$ は o の i 番目の属性の値（整数[1]）を表すものとする．オブジェクト o と o' の距離は $dist(o, o')$ と表し，$o.v$ と $o'.v$ のユークリッド距離を想定する[2]．ここで，本論文が取り組む問題の厳密版を定義する．

定義 1 (属性制約)．O，q，および属性値のリスト $\mathcal{A} = [q.A_1, ..., q.A_m]$ が与えられたとする．また，i 番目の属性の値 $q.A_i$ が null の場合，この属性には制約がないものとする．このとき，$O(\mathcal{A})$ を以下のように定義する．

$$O(\mathcal{A}) = \{o \mid o \in O, \forall i \in [1, m]\ s.t.\ q.A_i \neq \text{null}, o.A_i = q.A_i\} \tag{1}$$

定義 2 (属性制約下での最近傍探索)．O，q，属性値のリスト $\mathcal{A} = [q.A_1, ..., q.A_m]$，および結果サイズ k が与えられたとき，属性制約下での最近傍探索は $|S^*| = k$ かつ $\forall o \in S^*$ および $\forall o' \in O(\mathcal{A}) \backslash S^*$ に対して $dist(o, q) \leq dist(o', q)$ を満たす S^* を探索するものである（同値の場合は任意の方法で優劣をつける）．

問題定義． 本論文では上記の問題の近似版を考える．近似最近傍探索（ANNS）問題は，$S \subseteq O(\mathcal{A})$ を計算することであり，$|S| = k$ だが S が必ずしも S^* となるわけではない．S の精度を測るためにリコール（$Recall@k$）を以下のように定義する．

$$Recall@k = \frac{|S \cap S^*|}{k} \tag{2}$$

3 提案アルゴリズム

本章では，提案アルゴリズムの詳細について説明する．近似最近傍探索において，近接グラフを用いた手法は，その速度と精度の高さから，実用上最も有効であることが示されている [18]．本研究では，部分属性制約にも柔軟に対応できる近接グラフを組み合わせた新しいデータ構造を提案する．

アイデア． まず，全ての属性に値の指定があるクエリを考える．この場合を場合 1 とする．場合 1 では，O 内のオブジェクトが取り得る全ての属性値の組み合わせに対して近接グラフを構築すれば，クエリで指定された属性制約に該当する近接グラフを使うことにより，単純な貪欲法（クエリに最も近い隣接頂点に辿ることを繰り返す方法）を用いて ANNS を実行できる．さらに，各オブジェクトは一つの近接グラフにしか現れないため，グラフの辺数を無視した場合，空間計算量は $O(n)$ であり，大量データにスケールする．次に，ある単一の属性にのみ制約があるクエリを考え，場合 2 とする．この場合は，各属性の各値を持つオブジェクトの集合に対して近接グラフを構築すれば良い．このとき，各オブジェクトは m 個の近接グラフに現れるため，空間計算量は $O(nm)$ である．最後に，$1 < m' < m$ 個の属性に値の制約がある場合を考え，場合 3 とする．場合 1 および 2 のアイデアを適用する場合，各属性のドメインサイズの平均を γ とすると，$O(\gamma^m)$ 個の近接グラフが必要であり，空間計算量の観点から大量データにスケールしないことは自明である．つまり，定義1を満たす各属性制約に該当する近接グラフをそれぞれ構築する方法は現実的ではない．

ここで，場合 3 の属性制約は場合 1 のサブセットと考えることができ，同様に，場合 2 のスーパーセットと考えられる．例えば，$A_1 = x$，$A_2 = y$，...，$A_m = z$ であるオブジェクトの集合から構築された近接グラフ $G_{x,y,...,z}$ が存在するとし，$A_1 = x$，$A_2 = y$，...，$A_m = null$ の属性制約を指定したクエリが与えられたとする．このとき，$G_{x,y,...,z}$ 内のオブジェクトは全て指定された属性制約を満たすため，この近接グラフを利用することが合理的であることが分かる．しかし，この属性制約を満たすオブジェクトの各属性の値は，$A_1 = x$，$A_2 = y$，...，$A_m = z'$ や $A_1 = x$，$A_2 = y$，...，$A_m = z''$ であるものも存在するため，$G_{x,y,...,z}$ のみで探索するだけで高精度な解を得ることは難しい．そこで，例えば $G_{x,y,...,z}$ と $G_{x,y,...,z'}$ の間に辺が存在すれば，上記の属性制約を満たしつつクエリに近い（または近づく）オブジェクト「のみ」を探索できることが分かる．（つまり，自身がサブセットとなる属性制約に該当する全ての近接グラフに対して貪欲法を実行する必要がない．）空間計算量の増大を最小化しつつこのような辺を導入するため，場合 2 で作成された近接グラフが持つ辺を場合 1 で作成された近接グラフに追加する．一般的に m は小さく [8]，近接グラフでは類似したオブジェクト間に辺が作成される．また，場合 2 の各近接グラフの（平均の）次数を η とすると，各オブジェクトは（平均で）$O(m\eta)$ 個のそのオブジェクトに類似するオブジェクトへの辺を持つ．複数の属性に制約がある場合においても，これらの辺の中で制約を満たすオブジェクトのみを辿ることにより，無駄な距離計算を避けつつクエリに近いオブジェクトに近づくことができる．

3.1節では，提案するデータ構造およびその構築方法について説明する．その後，3.2節において，このデータ構造を活用した ANNS アルゴリズムを提案する．

3.1 データ構造

まず，提案アルゴリズムでは，各オブジェクトの各属性の値は一意の識別子（非負整数）に変換されているものと想定する．以降では，$o.A_i$（$i \in [1, m]$）はその識別子を表すものとする．

概要． 提案アルゴリズムで用いるデータ構造を説明するにあたり，$A_1 = x$，$A_2 = y$，...，$A_m = z$ であるオブジェクトの集合 $O_{x,y,...,z}$ を考える．（ここで，x，y，および z は O 内で取り得る変数である．）このオブジェクト集合の近接グラフを $G_{x,y,...,z} = (O_{x,y,...,z}, E_{x,y,...,z})$ と表現する．このとき，$O_{x,y,...,z}$ はグラフの頂点（＝オブジェクト）の集合を意味し，$E_{x,y,...,z}$ はラベル付き有向辺の集合を表す．（ラベルについては後に説明する．）提案アルゴリズムで用いるデータ構造は，O 内で取り得る $x, y, ..., z$ の組み合わせそれぞれに対して構築される近接グラフ $G_{x,y,...,z}$ の全てをマージしたグラフ $\mathcal{G}$ である．

[1]連続値は近似を許容することで整数領域にスケール可能である．
[2]提案アルゴリズムは他の距離関数も使用可能である．

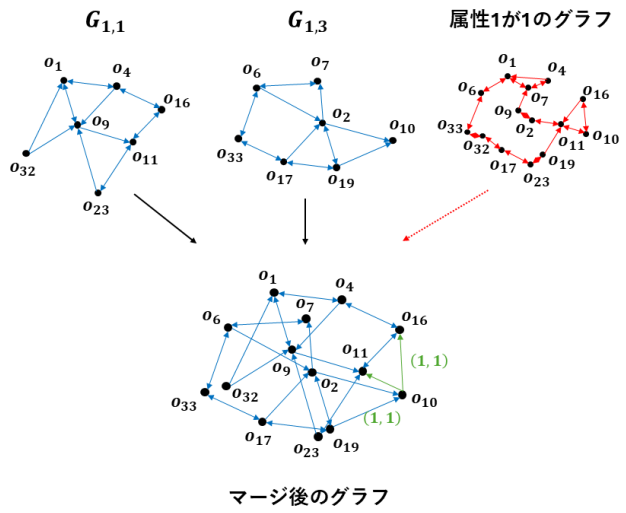


図 1: $m = 2$ **の場合の近接グラフマージの例．黒点がオブジェクトを，青矢印，赤矢印，および緑矢印がエッジを，緑文字がエッジのラベルを表している．**

ラベル付きエッジ． ある近接グラフ $G_{x,y,\ldots,z}$ に 2 つの頂点 o_a および o_b が存在するものとし，$(o_a, o_b) \in E_{x,y,\ldots,z}$ とする．この辺のラベル $l(o_a, o_b)$ は

$$l(o_a, o_b) = \begin{cases} \perp & (\text{if } o_a.A = o_b.A) \\ o_b.A & (\text{otherwise}) \end{cases} \tag{3}$$

である．

例 1. 例えば，$m = 2$，属性 *1* の値が *2* 種類，属性 *2* の値が *3* 種類である場合を考える．この時に考えられる属性値の組み合わせは，$\{1,1\}$，$\{1,2\}$，$\{1,3\}$，$\{2,1\}$，$\{2,2\}$，および $\{2,3\}$ であるが，O 内に $\{1,2\}$ を持つオブジェクトおよび $\{2,2\}$ を持つオブジェクトがない場合，これらの近接グラフについては構築しない．図*1*に，$\{1,1\}$ および $\{1,3\}$ に対して構築した近接グラフ $G_{1,1}$ および $G_{1,3}$ を属性 *1* の値が 1 であるオブジェクト（つまり $\{1,1\}$ および $\{1,3\}$ のどちらかに属するオブジェクト）に対して構築された近接グラフの辺によってマージする操作を示す．属性 *1* の値が 1 のグラフにおいて，オブジェクト o_{10} に着目すると，o_{10} から o_{11} および o_{16} に辺が張られている．o_{11} および o_{16} の属性は $\{1,1\}$ であるため，ラベルを $\{1,1\}$ とした辺 (o_{10}, o_{11}) および辺 (o_{10}, o_{16}) を追加する．これを全てのオブジェクトについて行うことで近接グラフをマージする．

$G_{x,y,\ldots,z}$ の条件． 提案アルゴリズムでは，$G_{x,y,\ldots,z}$ として非階層の任意の近接グラフが利用可能である．本論文では，$G_{x,y,\ldots,z}$ を近似最近傍グラフから構築する．

オフラインアルゴリズム． 提案アルゴリズムの前処理を Algorithm 1に示す．ここで構築されるデータ構造は任意のクエリに対して利用可能であり，前処理は一度だけ行われる．

まず，各 $O_{x,y,\ldots,z} \subseteq O$ に対して $G_{x,y,\ldots,z}$ を作成する．本論文では，$G_{x,y,\ldots,z}$ として近似最近傍グラフを用いる．厳密な最近傍グラフの構築は $O(n^2)$ 時間必要であるが，NNDescent [11] は高精度な近似最近傍グラフを $O(n)$ 時間で構築できる．これらの近接グラフ $G_{x,y,\ldots,z}$ をマージしたグラフ $\mathcal{G}$ を構築する．このとき，全ての $e \in E$ のラベル $l(e)$ は $\perp$ とする．

次に，各オブジェクト $o \in O$ を m 個のオブジェクト集合に入れる．具体的には，$O_{\langle i,x \rangle} = \{o \mid o \in O, o.A_i = x\}$ と定義し，各 $i \in [1, m]$ に対して o を $O_{\langle i, o.A_i \rangle}$ に入れる．その後，

Algorithm 1: Pre-Processing

Input: O, m, K (initial graph degree), K' (hyper-paramter)

1 $\mathcal{G} \leftarrow \varnothing$
2 **for** *each possible* $O_{x,y,\ldots,z} \subseteq O$ **do**
3 $\quad G_{x,y,\ldots,z} \leftarrow \text{NNDescent}(O_{x,y,\ldots,z}, K)$
4 $\quad \mathcal{G} \leftarrow \mathcal{G} \cup G_{x,y,\ldots,z}$ // $\forall e \in E, l(e) = \perp$
5 **foreach** $o \in O$ **do**
6 $\quad$ **foreach** $i \in [1, m]$ **do**
7 $\quad\quad H[\langle i, o.A_i \rangle] \leftarrow H[\langle i, o.A_i \rangle] \cup \{o\}$
8 **foreach** *key* h *of* H **do**
9 $\quad G_h \leftarrow \text{NNDescent}(H[h], K')$
10 $\quad$ **foreach** $(o_a, o_b) \in E_h$ *such that* $(o_a, o_b) \notin E$ **do**
11 $\quad\quad E \leftarrow E \cup \{(o_a, o_b)\}$ with $l(o_a, o_b) = o_b.A$
12 **return** $\mathcal{G}$

各 $O_{\langle i,x \rangle}$ の近似最近傍グラフを構築し，このグラフの各エッジ $e = (o_a, o_b)$ が E に含まれていない場合，$l(e) = o_b.A$ として E に追加する．

時間計算量． 2–4 行目では合計 n 個のオブジェクトに対して NNDescent を実行しており，$K = O(1)$ とすると，この操作は $O(n)$ 時間必要である．5–7 行目では，各オブジェクトを m 個のオブジェクト集合に挿入しているため，$O(nm)$ 時間が生じる．8–11 行目は，合計 nm 個のオブジェクトに対して NNDescent を実行しており，$K' = O(1)$ とすると，$O(nm)$ 時間が生じる．また，この操作で生じる辺の総数は $O(nm)$ であるため，Algorithm 1の時間計算量は $O(nm)$ となる．

空間計算量． 上の時間計算量の議論から，Algorithm 1および $\mathcal{G}$ の空間計算量は $O(nm)$ となる．

3.2 クエリ処理

次に，3.1節で提案したデータ構造を活用する ANNS アルゴリズムを提案する．Algorithm 2にその手順を示す．ここで，ϵ（≥ 1）は探索時間と精度のトレードオフを取るパラメータである．

まず，$\mathcal{G}$ 内のどの頂点（オブジェクト）から探索を始めるかを決定する．このとき，属性制約 $\mathcal{A}$ 内に null がなければ，$\mathcal{A}$ を満たすオブジェクトの集合からランダムなオブジェクトを開始頂点 o_s とする．一方，$\mathcal{A}$ 内に null が含まれる場合，null である属性の値をその属性に存在する値からランダムなものにした $\mathcal{A}'$ を考え，null が存在しない場合と同様の方法で o_s を決める．

次に $\mathcal{G}$ において，o_s から貪欲法による探索を始める．o_s を探索済みオブジェクト集合 V に，$\langle o_s, dist(o_s, q) \rangle$ を優先付きキュー P（暫定解）および P'（探索オブジェクトの候補集合）に挿入する．

今，P' の先頭となるオブジェクトを o とし，P' をポップする．その後，o の各隣接オブジェクトで未訪問のもの o' にアクセスする．このオブジェクトを V に追加し，訪問済みとする．次に，$l(o, o')$ のラベルが $\perp$ の場合，$dist(o', q)$ を計算し，これが暫定解の閾値 τ 未満であれば，P および P' に $\langle o', dist(o', q) \rangle$ を追加し，τ を更新する．また，$|P'| > \epsilon \cdot k$ であれば，P' の末のタプルを削除する．一方，$l(o, o')$ のラベルが $\perp$ でない場合，$o'.A$ が $\mathcal{A}$ を満たす場合にのみ $l(o, o')$ のラベルが $\perp$ の場合と同様の処理を行う．この操作を P' が空になるまで繰り返し，P の先頭の k 個のオブジェクトを出力する．

Algorithm 2: ANNS

Input: $\mathcal{G}$, q, $\mathcal{A} = \{q.A_1, ..., q.A_m\}$, ϵ (≥ 1), and k

/* Determining a start node */

1 $\mathcal{A}' \leftarrow \mathcal{A}$

2 **for** *all* $i \in \mathcal{A}'$ *such that* $q.A_i = null$ **do**

3 Replace $q.A_i$ with a random value in the domain of the i-th attribute

4 $o_s \leftarrow$ a random node in $\{o \mid o \in O, o.A = \mathcal{A}'\}$

/* Traversing the graph from the start node */

5 $V \leftarrow \{o_s\}$ // V is a set of visited nodes

6 $P, P' \leftarrow \langle o_s, dist(o_s, q)\rangle$ // P and P' are sorted by $dist(\cdot, q)$)

7 $\tau \leftarrow \infty$

8 **while** $P' \neq \emptyset$ **do**

9 $o \leftarrow$ the top element in P'

10 Pop P'

11 **foreach** o' *such that* $(o, o') \in E$ *and* $o' \notin V$ **do**

12 $V \leftarrow V \cup \{o'\}$

13 **if** $l(o, o') = \perp$ **then**

14 **if** $dist(o', q) < \tau$ **then**

15 Add $\langle o', d(o', q)\rangle$ into P and P'

16 Update τ // the $\epsilon \cdot k$-th $dist(\cdot, q)$ in P

17 **if** $|P| > \epsilon \cdot k$ **then**

18 Erase the last element in P

19 **else**

20 **if** $o'.A$ *satisfies* $\mathcal{A}$ **then**

21 Run lines 14–18

22 **return** the first k elements in P

4 評価実験

本章では, 実験結果について述べる. 全ての実験は, Ubuntu 24.04.1 LTS, Intel Xeon Gold 6254@3.10GHz CPU, および 768GB の RAM を搭載した計算機で実施した.

4.1 設定

データセット. 実験には，ANNS 問題の研究で一般的に使用される実データセットである Deep [7]，GIST [5]，および SIFT [5] を使用した [16, 18, 21, 27, 30]．Deep は 1,000 万件の 96 次元深層特徴ベクトル，GIST は 100 万件の 960 次元 GIST 特徴ベクトル，および SIFT は 100 万件の 128 次元 SIFT 特徴ベクトルから構成される．

既存研究 [16, 17, 21, 26, 32, 34] と同様に，各 $o.A_i$ を一様分布に従うランダムな値に設定した．ここで $m = 3$ とし，A_1 を 100 種類の非負整数，A_2 を 10 種類の非負整数，および A_3 を 12 種類の非負整数とした．また，各 A_i はすべて一様分布に従うよう設定した．

Deep，GIST，および SIFT すべてのデータセットにおいて，それぞれのソースで与えられるクエリベクトルからランダムに 1,000 個を選んで使用した．また，属性制約として指定する属性を以下のように 3 種類設定した．(1) 指定する属性を 1 つとし，どの属性を指定するかはランダムに選択．(2) 指定する属性を 2 つとし，どの属性を指定するかはランダムに選択．(3) 全ての属性を指定．クエリの属性値もデータセットの設定と同様に，一様分布に従うランダムな値に設定した．

評価アルゴリズム. 評価対象としたアルゴリズムは以下の通りである．

- Pre-filter：1章で紹介したアルゴリズム．
- Post-filter：1章で紹介したアルゴリズム．ANNS インデックスとして近似最近傍グラフ（$K = 30$）を用いた．
- Ours：3章で提案したアルゴリズム．$K = 30$ および $K' = 20$ と設定した．

これらのアルゴリズムは C++ で実装され，g++ 13.3.0 で -O3 最適化オプションを用いてコンパイルした．これらのアルゴリズムのデータ構造は，faiss[3]を用いてマルチスレッド（ハイパースレッディングを用いた 72 スレッド）でオフライン構築した．一方，ANNS はシングルスレッドで実行した．

評価指標. 各アルゴリズムの実行時間と精度のトレードオフを制御するパラメータ ϵ を変化させながら，平均実行時間と平均リコールを測定した．

表 1: 前処理時間 [sec]

データセット	前処理時間 [sec]
Deep	9271.710
GIST	3126.192
SIFT	885.805

表 2: Pre-filter の探索時間 [msec]

データセット	クエリの種類	探索時間 [msec]
Deep	1 属性指定	433.104
	2 属性指定	51.821
	3 属性指定	33.099
GIST	1 属性指定	63.106
	2 属性指定	9.544
	3 属性指定	3.030
SIFT	1 属性指定	31.919
	2 属性指定	4.286
	3 属性指定	2.946

4.2 前処理時間

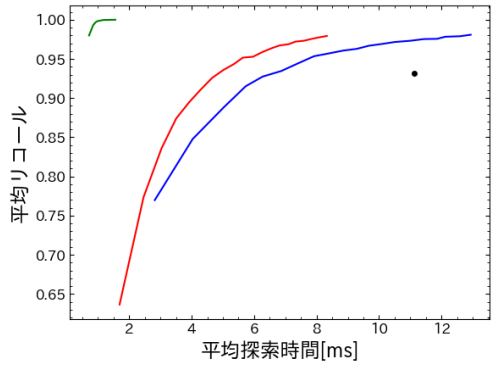
表1に，各データセットにおける提案アルゴリズムの前処理時間を示す．Deep はデータ数が 1,000 万であり，GIST および SIFT の 100 万と比較して非常に大きいため，前処理時間が長くなっている．また，GIST および SIFT のデータ数は同じだが，GIST は 960 次元と SIFT の 128 次元と比較して大きいため，前処理時間も長くなっている．この結果は，3.1節で分析した時間計算量と一致する．

4.3 平均探索時間と平均リコール

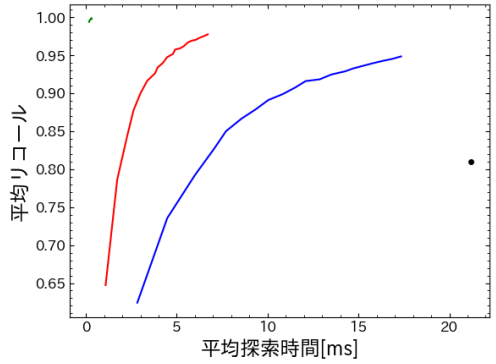
本節では，各アルゴリズムのクエリ処理（オンライン）時間と精度を調べる．表2は Pre-filter の探索時間を示している．Pre-filter は厳密解の探索であるため，リコールは 1 である．図2(a), 図2(b), および図2(c)は，それぞれ Deep, GIST, および SIFT データセット上での Ours の平均探索時間と平均リコールの関係を示している．これらの実験では $k = 10$ とした．

Ours 内での比較. すべてのデータセットにおいて，3 属性指定での時間-精度のトレードオフが 1 属性指定および 2 属性指定での結果を大きく上回っている．3 属性指定は完全属性指定であり，$l(e) = \perp$ となる辺のみ辿ることと，属性制約を満たすオブジェクトの数が少ないことが上の結果の理由である．また，すべてのデータセットにおいて，2

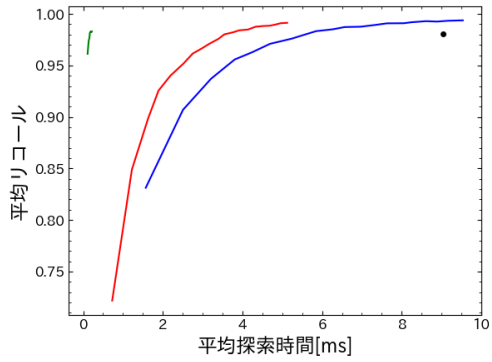
[3]https://github.com/facebookresearch/faiss



(a) Deep



(b) GIST



(c) SIFT

図 2: 各データセットにおける Ours および Post-filter の $k = 10$ での結果．—がOurs における 1 属性指定の結果を，—がOurs における 2 属性指定の結果を，—がOurs における 3 属性指定の結果を，• が Post-filter における 1 属性指定の結果を表している．

属性指定での時間-精度のトレードオフが 1 属性指定のものを大きく上回っている．これは，属性制約を満たす辺の数が理由である．

Ours と Pre-filter の比較. Deep，GIST，および SIFT のデータセットにおいて，3 属性指定の場合は Pre-filter に対して，それぞれ，約 40 倍，約 17 倍，および約 17 倍と大幅な高速化を果たしている（Recall@10 ≈ 1）．これは 3 属性指定において，Ours は $l(e) = \perp$ となる辺のみを辿ることで ANNS を行うことができ，無駄な距離計算が少ないためである．

また，Deep，GIST，および SIFT のデータセットにおいて，1 属性指定の場合は Pre-filter に対してそれぞれ，約 54 倍，約 3.7 倍，および約 9.3 倍の高速化を実現している（Recall@10 ≈ 0.95）．しかし，2 属性指定の場合では約 9.2 倍，約 2.0 倍，および約 1.7 倍となっており，1 属性指定の場合と比較して，見劣りする結果となっている（Recall@10 ≈ 0.95）．これは，2 属性指定において候補となる解の個数は 1 属性指定と比較して少ないが，辿る頂点の数が比較的大きいことが原因と考えられる．.

Deep はデータセット内で最も高速化に成功している．これは，近接グラフの効果の大きさを示しており，Ours が大規模データセットに適していることがわかる．GIST の高速化が SIFT に対して見劣りするのは次元数による影響だと考える．最近傍探索において，探索に占める距離計算の割合は大きい．そのため，データ数が同じであっても，次元数に差があれば探索時間に大きな差が生じる．

Ours と Post-filter の比較. 図2において，1 属性指定の場合における Ours と Post-filter の時間-精度（リコール）のトレードオフを比較すると，Ours の性能の方が優れていることが分かる．Post-filter は属性制約を考慮した探索ができないため，探索範囲が大きい．また，属性制約に含まれる属性数が増えると Post-filter の探索時間が増加することは自明であり，2 属性に制約がある場合の Post-filter の探索時間は 1 属性の場合よりも（同程度のリコールで）遅くなる．（そのため，Post-filter での 2 属性および 3 属性指定の実験を割愛している．）一方，Ours は制約に含まれる属性数が増加するほど探索時間は早くなり，Post-filter に対する優位性がある．

5 関連研究

ANNS. 多くの研究が効率的かつ高精度な ANNS アルゴリズムの開発に取り組んできた．局所性鋭敏型ハッシュは理論的アプローチであるが，実践的には他のアプローチよりも遅い [18]．量子化は空間効率が良いが，高いリコールを提供できない [31]．近接グラフは現在の最先端のアプローチであり [30]，最近傍グラフ型やスモールワールドグラフ型に分類される．Vamana [23] および HNSW [19] が代表的な近接グラフであり，それぞれ前者および後者に属する．これらは属性値付きケースにも拡張されている．

属性制約付き ANNS. 既存研究の多くは近接グラフを使用しており [16, 21, 27, 34, 37]，単一属性を前提としてエッジ追加戦略を工夫することで Post-filter の最適化を図っている．一方，これらの近接グラフを複数属性を持つオブジェクトへ拡張する点は自明ではない．

ディスクベース ANNS. 近年，データベースのデータサイズは急拡大している．それに伴い，ディスクにインデックスを格納し，効率良く探索を行うディスクベース ANNS の研究も行われている [9, 16, 28, 29]．ディスクにデータを格納する設定への提案アルゴリズムの拡張は今後の課題である．

6 結論

本論文では，属性制約下での近似最近傍探索問題に取り組んだ．この問題は実世界で数多くの応用があるが，既存アルゴリズムでは効率的に部分属性制約に対応できない．この問題を解決するため，本論文では新しいアルゴリズムを提案した．また，評価実験において，提案アルゴリズムの性能が既存アルゴリズムの性能を上回ることを示した．

謝辞

本研究の一部は，AIP 加速課題（JPMJCR23U2）の支援を受けたものである．

参考文献

[1] [n. d.]. https://github.com/vearch/vearch.
[2] [n. d.]. https://github.com/vespa-engine.
[3] [n. d.]. https://weaviate.io/.
[4] [n. d.]. https://www.pinecone.io/.
[5] [n. d.]. http://corpus-texmex.irisa.fr/.
[6] Yusuke Arai, Daichi Amagata, Sumio Fujita, and Takahiro Hara. 2021. LGTM: A Fast and Accurate kNN Search Algorithm in High-dimensional Spaces. In *DEXA*. 220–231.
[7] Artem Babenko and Victor Lempitsky. 2016. Efficient Indexing of Billion-scale Datasets of Deep Descriptors. In *CVPR*. 2055–2063.
[8] Yuzheng Cai, Jiayang Shi, Yizhuo Chen, and Weiguo Zheng. 2024. Navigating Labels and Vectors: A Unified Approach to Filtered Approximate Nearest Neighbor Search. *Proceedings of the ACM on Management of Data* 2, 6 (2024), 1–27.
[9] Qi Chen, Bing Zhao, Haidong Wang, Mingqin Li, Chuanjie Liu, Zengzhong Li, Mao Yang, and Jingdong Wang. 2021. SPANN: Highly-efficient Billion-scale Approximate Nearest Neighborhood Search. In *NeurIPS*. 5199–5212.
[10] Wei Dong, Charikar Moses, and Kai Li. 2011. Efficient k-nearest Neighbor Graph Construction for Generic Similarity Measures. In *World Wide Web*. 577–586.
[11] Wei Dong, Charikar Moses, and Kai Li. 2011. Efficient K-nearest Neighbor Graph Construction for Generic Similarity Measures. In *WWW*. 577–586.
[12] Cong Fu, Changxu Wang, and Deng Cai. 2021. High Dimensional Similarity Search with Satellite System Graph: Efficiency, Scalability, and Unindexed Query Compatibility. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 8 (2021), 4139–4150.
[13] Yujian Fu, Cheng Chen, Xiaohui Chen, Weng-Fai Wong, and Bingsheng He. 2024. Optimizing the Number of Clusters for Billion-scale Quantization-based Nearest Neighbor Search. *IEEE Transactions on Knowledge and Data Engineering* (2024).
[14] Jianyang Gao and Cheng Long. 2024. RaBitQ: Quantizing High-Dimensional Vectors with a Theoretical Error Bound for Approximate Nearest Neighbor Search. *Proceedings of the ACM on Management of Data* 2, 3 (2024), 1–27.
[15] Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. 2013. Optimized Product Quantization. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36, 4 (2013), 744–755.
[16] Siddharth Gollapudi, Neel Karia, Varun Sivashankar, Ravishankar Krishnaswamy, Nikit Begwani, Swapnil Raz, Yiyong Lin, Yin Zhang, Neelam Mahapatro, Premkumar Srinivasan, et al. 2023. Filtered-diskann: Graph algorithms for Approximate Nearest Neighbor Search with Filters. In *Web Conference*. 3406–3416.
[17] Gaurav Gupta, Jonah Yi, Benjamin Coleman, Chen Luo, Vihan Lakshman, and Anshumali Shrivastava. 2023. CAPS: A Practical Partition Index for Filtered Similarity Search. *arXiv preprint arXiv:2308.15014* (2023).
[18] Wen Li, Ying Zhang, Yifang Sun, Wei Wang, Mingjie Li, Wenjie Zhang, and Xuemin Lin. 2019. Approximate Nearest Neighbor Search on High Dimensional Data—Experiments, Analyses, and Improvement. *IEEE Transactions on Knowledge and Data Engineering* 32, 8 (2019), 1475–1488.
[19] Yu A Malkov and Dmitry A Yashunin. 2018. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligencee* 42, 4 (2018), 824–836.
[20] Jason Mohoney, Anil Pacaci, Shihabur Rahman Chowdhury, Ali Mousavi, Ihab F Ilyas, Umar Farooq Minhas, Jeffrey Pound, and Theodoros Rekatsinas. 2023. High-throughput Vector Similarity Search in Knowledge Graphs. *Proceedings of the ACM on Management of Data* 1, 2 (2023), 1–25.
[21] Liana Patel, Peter Kraft, Carlos Guestrin, and Matei Zaharia. 2024. ACORN: Performant and Predicate-Agnostic Search Over Vector Embeddings and Structured Data. *Proceedings of the ACM on Management of Data* 2, 3 (2024), 1–27.
[22] Yun Peng, Byron Choi, Tsz Nam Chan, Jianye Yang, and Jianliang Xu. 2023. Efficient Approximate Nearest Neighbor Search in Multi-dimensional Databases. *Proceedings of the ACM on Management of Data* 1, 1 (2023), 1–27.
[23] Suhas Jayaram Subramanya, Devvrit, Rohan Kadekodi, Ravishankar Krishaswamy, and Harsha Vardhan Simhadri. 2019. DiskANN: Fast Accurate Billion-point Nearest Neighbor Search on a Single Node. In *NeurIPS*. 13766–13776.
[24] Yao Tian, Xi Zhao, and Xiaofang Zhou. 2022. DB-LSH: Locality-Sensitive Hashing with Query-based Dynamic Bucketing. In *ICDE*. 2250–2262.
[25] Yao Tian, Xi Zhao, and Xiaofang Zhou. 2024. DB-LSH 2.0: Locality-Sensitive Hashing With Query-Based Dynamic Bucketing. *IEEE Transactions on Knowledge and Data Engineering* 36, 3 (2024), 1000–1015.
[26] Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, Peng Xu, Shengjun Li, Xiangyu Wang, Xiangzhou Guo, Chengming Li, Xiaohai Xu, et al. 2021. Milvus: A Purpose-built Vector Data Management System. In *SIGMOD*. 2614–2627.
[27] Mengzhao Wang, Lingwei Lv, Xiaoliang Xu, Yuxiang Wang, Qiang Yue, and Jiongkang Ni. 2023. An Efficient and Robust Framework for Approximate Nearest Neighbor Search with Attribute Constraint. In *NeurIPS*. 15738–15751.
[28] Mengzhao Wang, Lingwei Lv, Xiaoliang Xu, Yuxiang Wang, Qiang Yue, and Jiongkang Ni. 2023. An Efficient and Robust Framework for Approximate Nearest Neighbor Search with Attribute Constraint. In *NeurIPS*. 15738–15751.
[29] Mengzhao Wang, Weizhi Xu, Xiaomeng Yi, Songlin Wu, Zhangyang Peng, Xiangyu Ke, Yunjun Gao, Xiaoliang Xu, Rentong Guo, and Charles Xie. 2024. Starling: An i/o-efficient disk-resident graph index framework for high-dimensional vector similarity search on data segment. *Proceedings of the ACM on Management of Data* 2, 1 (2024), 1–27.
[30] Mengzhao Wang, Xiaoliang Xu, Qiang Yue, and Yuxiang Wang. 2021. A Comprehensive Survey and Experimental Comparison of Graph-based Approximate Nearest Neighbor Search. *Proceedings of the VLDB Endowment* 14, 11 (2021), 1964–1978.
[31] Runhui Wang and Dong Deng. 2020. DeltaPQ: Lossless Product Quantization Code Compression for High Dimensional Similarity Search. *Proceedings of the VLDB Endowment* 13, 13 (2020), 3603–3616.
[32] Chuangxian Wei, Bin Wu, Sheng Wang, Renjie Lou, Chaoqun Zhan, Feifei Li, and Yuanzhe Cai. 2020. AnalyticDB-V: A Hybrid Analytical Engine Towards Query Fusion for Structured and Unstructured Data. *Proceedings of the VLDB Endowment* 13, 12 (2020), 3152–3165.
[33] Jiuqi Wei, Botao Peng, Xiaodong Lee, and Themis Palpanas. 2024. DET-LSH: A Locality-Sensitive Hashing Scheme with Dynamic Encoding Tree for Approximate Nearest Neighbor Search. *Proceedings of the VLDB Endowment* 17, 9 (2024), 2241–2254.
[34] Wei Wu, Junlin He, Yu Qiao, Guoheng Fu, Li Liu, and Jin Yu. 2022. HQANN: Efficient and Robust Similarity Search for Hybrid Queries with Structured and Unstructured Constraints. In *CIKM*. 4580–4584.
[35] Wen Yang, Tao Li, Gai Fang, and Hong Wei. 2020. Pase: Postgresql Ultra-high-dimensional Approximate Nearest Neighbor Search Extension. In *SIGMOD*. 2241–2253.
[36] Shangdi Yu, Joshua Engels, Yihao Huang, and Julian Shun. 2023. Pecann: Parallel Efficient Clustering with Graph-based Approximate Nearest Neighbor Search. *arXiv preprint arXiv:2312.03940* (2023).
[37] Weijie Zhao, Shulong Tan, and Ping Li. 2022. Constrained Approximate Similarity Search on Proximity Graph. *arXiv preprint arXiv:2210.14958* (2022).
[38] Bolong Zheng, Zhao Xi, Lianggui Weng, Nguyen Quoc Viet Hung, Hang Liu, and Christian S Jensen. 2020. PM-LSH: A Fast and Accurate LSH Framework for High-dimensional Approximate NN Search. *Proceedings of the VLDB Endowment* 13, 5 (2020), 643–655.