一般発表 | Track 2: ビッグデータ基盤技術・データセキュリティ・プライバシ

2026年3月2日(月) 15:30 ~ 17:40 | E会場

[9E] グラフデータベース

座長:Peng Shaowen(奈良先端科学技 大学院大学)　コメンテータ:山本 大介(名古屋工業大学)

15:30 ~ 15:55

[9E-01] グラフスキーマ評価手法の提案とユーザスタディによる整合性検証

*湯川 楓祐[1]、塩川 浩昭[1] (1. 筑波大学)

15:55 ~ 16:20

[9E-02] 機械学習によるサブグラフマッチングアルゴリズム選択

*倉谷 元琉[1]、Konstantinos Skitsas[2]、Karras Panagiotis[3]、天方 大地[1]、佐々木 勇和[1] (1. 大阪大学大学院、2. オーフス大学、3. コペンハーゲン大学)

16:20 ~ 16:45

[9E-03] グラフモデルを基盤とする異種データベース統合システムの開発と評価

*若林 拓[1]、常 穹[1]、宮崎 純[1] (1. 東京科学大学)

16:45 ~ 17:05

[9E-04] 意味埋め込みを用いた連邦型知識グラフ問合せにおける情報源選択

*林田 康太[1]、天笠 俊之[2] (1. 筑波大学 知識・データ工学研究室、2. 筑波大学 計算科学研究センター)

グラフスキーマ評価手法の提案とユーザスタディによる整合性検証

湯川　楓祐†　　塩川　浩昭††

† 筑波大学情報学群情報科学類　〒 305–8573 茨城県つくば市天王台 1-1-1
†† 筑波大学計算科学研究センター　〒 305–8573 茨城県つくば市天王台 1-1-1
E-mail: †f.yukawa@kde.cs.tsukuba.ac.jp, ††shiokawa@cs.tsukuba.ac.jp

あらまし　プロパティグラフ（PG）は属性を持つノードとエッジによりデータを表現するグラフデータモデルであり，ソーシャルネットワークをはじめとする多様な領域で利用されている．PG はスキーマレス運用によって柔軟なデータ表現を可能とする一方で，データ構造の一貫性が保証されず，クエリ性能の低下やデータ統合の複雑化を招く可能性がある．この問題に対処するため，PG に対して構造的制約を与えるスキーマを自動抽出する研究が進められているが，抽出されたスキーマの品質を定量的に評価する方法は十分に確立されていない．そこで本研究では PG のスキーマ品質を網羅性と簡潔性の観点から定量化する評価手法を提案する．提案手法の有効性を検証するため，まず複数の実データセットに対してスキーマに意図的な誤りを挿入する実験を行い，提案手法が適切に評価値を変化させることを確認した．さらにユーザスタディを通じて人間の判断と提案手法が示す評価傾向の整合性を分析し，提案手法が人間の判断と一貫した評価傾向を示すことを明らかにした．

キーワード　グラフデータベース，プロパティグラフ，NoSQL，スキーマ，ユーザスタディ

1　はじめに

プロパティグラフ（PG）は属性付きのノードとエッジからなるデータモデルであり，グラフデータベースにおいて広く利用されている．PG ではスキーマを事前に厳密に定義せずに運用できるためデータの構造の拡張に柔軟に対応できる一方，スキーマが明示されないまま拡張が進むとデータの一貫性が低下し，クエリの最適化やデータ統合が困難となる．本稿では，実際にデータが格納された PG を**インスタンス**と呼ぶ．図 1 に，ソーシャルネットワークを模した PG インスタンスの例を示す．ノードやエッジにはラベルとプロパティが付与され，ユーザや投稿，それらの関係が表現されている．

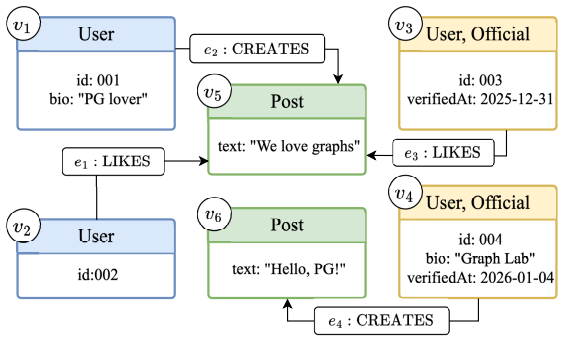


図 1: $\mathcal{G}$：SNS を模した PG の例

このような PG インスタンスに対し後付けでスキーマを導入することで，許容されるラベルやプロパティなどの制約を明示できるため，データ一貫性の検証やデータ統合の支援が期待できる．しかし，インスタンスが大規模かつ複雑な場合には人手によるスキーマ設計は困難であり，近年では PG インスタンスからスキーマを自動抽出する手法が多数提案されている [1–4]．

しかし同一の PG インスタンスに対しても，人手設計や自動抽出などにより複数の異なるスキーマ候補が得られ得る．図 2 は人手で設計されたスキーマの例であり，全体構造は直感的に理解しやすい一方，希少なプロパティ `bio` が欠落していたり，`Post` ノードの不要な区別が導入されている．一方，図 3 に示す自動抽出スキーマは属性の欠落を補完しているが，`User` や `id` が重複して定義されており，必要な定義が最小化されていない．このように PG のスキーマ品質は多面的であり，複数のスキーマ候補を客観的かつ定量的に比較することは容易ではない．

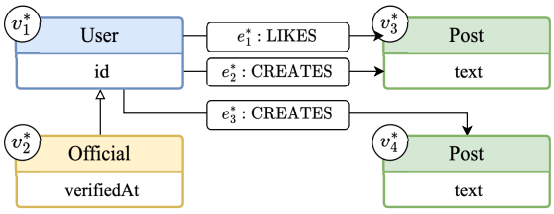


図 2: $\mathcal{S}^*$：人手で設計されたスキーマの例

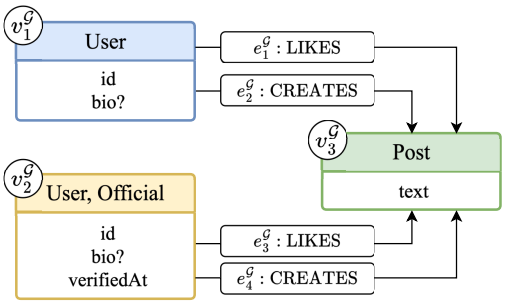


図 3: $\mathcal{S}^{\mathcal{G}}$：Xue の手法 [1] により自動抽出されたスキーマ

これまでリレーショナルデータや半構造化データを対象として，さまざまな観点からスキーマ品質を評価する研究が行われてきた [5,6]．しかし複数ラベルやプロパティの組み合わせに基づく多様な構造を許容する PG 特有の柔軟なデータモデルを考慮した定量的評価手法は十分に確立されていない．

そこで本研究では (i) インスタンスに観測される属性や制約をどの程度捉えているかを測る「網羅性」，(ii) 不要な重複や過剰な細分化を抑えた最小限の記述であるかを測る「簡潔性」，およびこれらを統合した指標である「C2 スコア」に基づく PG のスキーマ評価手法を提案する．本研究の主な貢献は以下の通りである．

1. **定量性**：網羅性と簡潔性に基づく定量的なスキーマ評価指標を定義し，それらの調和平均として統合指標 C2 スコアを導入した．
2. **適用性**：任意プロパティや継承関係などを含む複雑な PG スキーマに対しても，一貫した品質評価が可能であることを示した．
3. **スケーラビリティ**：大規模 PG インスタンスから生成されたスキーマに対しても，実用的な時間で評価可能であることを確認した．
4. **妥当性**：ユーザスタディおよび複数の実データ・人工データを用いた実験により，提案指標が人間の直感的評価と整合する傾向を持つことを示した．

2 関 連 研 究

半構造化データに対するスキーマ定義，抽出，および評価はデータ構造の理解や運用効率向上を目的として長年研究されてきた．本節ではまず既存のスキーマ定義と抽出技術を概観し，最後にスキーマ評価に関する研究動向と課題を整理する．

2.1 半構造化データのスキーマ定義

XML では DTD や XSD，JSON では JSON Schema，RDF では RDF Schema や OWL といったスキーマ言語が整備され，データ整合性の確保や相互運用性の向上を支えてきた [7–11]．

一方，PG におけるスキーマ定義は比較的新しい研究分野であり，初期には基本的なデータ構造や制約の形式化が行われ [12]，その後にはデータ検証やスキーマの拡張を扱う理論的枠組みや，一階述語論理に基づく形式的定義が提案された [13,14]．近年では GQL を見据えた PG-Schema [15] をはじめ，関数従属性や正規化手法に関する研究も進展している [16–18]．PG のスキーマの明確な定義と標準化は，クエリ高速化，データ品質の向上および異種システム間の相互運用性の確保に向けて重要な課題である [19,20]．

しかし実運用の場面ではスキーマが事前に設計されていない場合も多く，実データからの後付け設計は容易ではない．このギャップを解消するため，既存データからスキーマ情報を自動的に推定するスキーマ抽出技術が不可欠となる．

2.2 スキーマ抽出

スキーマ抽出は，明示的なスキーマを持たないデータからその構造的特徴を自動的に導出する技術である．XML や JSON におけるスキーマ抽出は，パス構造や属性分布を利用した手法からヒューリスティックおよび並列処理可能な手法へと発展してきた [5,7,21–23]．また RDF では多対多関係を持つグラフ構造を扱うためクラスタリングに基づくスキーマ抽出が主流となり [24,25]，この発想は一部の PG 向けスキーマ抽出手法にも引き継がれている．

PG においては，これまで類似度ベースクラスタリング手法 [1]，ルールベース手法 [2]，および階層型クラスタリング手法 [3,4] が提案されてきた．しかし，型階層の表現能力，プロパティ共起の扱い，スケーラビリティや冗長性といった点でそれぞれの手法ごとに限界があり，標準的アプローチは確立されていない．したがって，(1) スキーマ抽出手法を比較・分析するための基盤として (2) 抽出したスキーマが実運用に耐えうる品質であるかを事前に検証するための仕組みとしてスキーマの品質を客観的に判断できる評価の枠組みが必要である．

表 1: スキーマ評価手法の比較

評価項目	正解との比較	人間の主観評価	提案手法
正解スキーマの要否	× 必要	✓ 不要	✓ **不要**
定量性 / 再現性	✓ 高い	× 低い	✓ **高い**
実データへの適用	× 困難	✓ 可能	✓ **可能**

2.3 スキーマ評価

既存の半構造化データ分野では，正解スキーマとの一致度に基づく精度・再現率，スキーマの網羅性や冗長性，および実行性能などを用いた定量的な評価が行われてきた [5,22,23,26]．また，スキーマの可読性や妥当性に関する人間による主観的評価も重視されている [5,27]．

一方，PG 領域では主に正解スキーマとの一致度に基づく評価がスキーマ抽出手法の性能比較に用いられてきたが [1–3]，正解スキーマが存在しない実運用環境においてスキーマや抽出手法の品質・性能を評価する枠組みは十分に検討されていない．また，定量的な指標と人間の主観的評価との対応関係を体系的に検証した研究も報告されていない．

そこで本研究では，正解スキーマに依存せず，網羅性および簡潔性の観点から任意の PG スキーマの品質を定量的に評価する手法を提案する．さらに，ユーザスタディを通じて提案指標と人間の主観的評価との整合性を検証する．

表 1 に，本研究の評価手法と既存の評価手法との違いをまとめる．正解スキーマとの比較に基づく評価は 客観性や定量性に優れる一方で，正解スキーマを前提とするため実運用環境への適用が困難である．また，人間による主観評価は実データに対して柔軟に適用可能であるものの，評価の再現性や定量性に課題が残る．これに対し，本研究で提案する評価手法は正解スキーマを必要とせず，網羅性および簡潔性という定量指標に基づいて客観的かつ再現可能な評価を実現する点に特徴があり，実運用環境における PG スキーマ評価への適用が可能である．

表 2: 本稿で用いる主な記号

記号	説明
$\mathcal{G}$	PG インスタンス
$V(\cdot), E(\cdot)$	ノード集合，エッジ集合
$\lambda(o)$	オブジェクト o のラベル集合
$\kappa(o)$	オブジェクト o のプロパティキー集合
$\mu(o,k)$	オブジェクト o のプロパティ k の必須／任意
$\rho(o)$	オブジェクト o の親オブジェクト型の集合
$\mathbf{src}(e), \mathbf{dst}(e)$	エッジ e の始点ノード，終点ノード
$\mathbf{req}(o), \mathbf{opt}(o)$	オブジェクト o の必須／任意プロパティ集合
$\mathcal{S}^*$	評価対象となる元のスキーマ
$\mathcal{S}^+$	継承関係を展開したスキーマ（$= \mathbf{Flat}(\mathcal{S}^*)$）
$\mathcal{S}^{\mathcal{G}}$	$\mathcal{G}$ から抽出したスキーマ（$= \mathbf{Abs}(\mathcal{G})$）
$\mathbf{sim}_V(\cdot,\cdot), \mathbf{sim}_E(\cdot,\cdot)$	ノード型（エッジ型）間の類似度
$\mathbf{Cvg}_V, \mathbf{Cvg}_E$	ノード／エッジ網羅性
$\mathbf{Con}_V, \mathbf{Con}_E$	ノード／エッジ簡潔性
$\mathbf{C2}_V, \mathbf{C2}_E$	ノード／エッジ C2 スコア

3 事 前 準 備

3.1 用語の定義

本稿では，PG におけるノードおよびエッジを総称して**オブジェクト**と呼ぶ．また，オブジェクトに付与されるラベルおよびプロパティキーを総称して**属性**と呼ぶ．さらに，実際に観測されるノードとエッジの集合として与えられるグラフを**インスタンス**と呼び，インスタンスに現れ得るオブジェクトの構造的特徴を抽象化したものを**スキーマ**と呼ぶ．以降，特に断りのない限り，インスタンスとスキーマを区別して議論する．

3.2 プロパティグラフ

本研究では ISO/IEC 39075:2024 [28] に準拠し，GQL と互換性のある PG を対象とする．以下では，本稿の議論に必要な最小限の構成要素のみを定義する．

定義 3.2.1 (プロパティグラフ)**.** ラベル集合を $\mathcal{L}$，プロパティキー集合を $\mathcal{K}$ とする．プロパティグラフを $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \lambda, \kappa, \sigma)$ と定義する．ここで $\mathcal{V}$ はノードの有限集合，$\mathcal{E}$ は有向エッジの有限集合であり，$(\mathcal{V} \cap \mathcal{E}) = \varnothing$ とする．$\lambda : (\mathcal{V} \cup \mathcal{E}) \to 2^{\mathcal{L}}$ および $\kappa : (\mathcal{V} \cup \mathcal{E}) \to 2^{\mathcal{K}}$ はそれぞれ各オブジェクトに付与されたラベル集合およびプロパティキー集合を返す関数である．また $\sigma : \mathcal{E} \to (\mathcal{V} \times \mathcal{V})$ は各エッジの始点および終点ノードの順序付きタプルを返す関数である．

本稿ではエッジ $e \in \mathcal{E}$ に対して $\sigma(e) = (\mathbf{src}(e), \mathbf{dst}(e))$ と表記し，$\mathbf{src}(e)$ と $\mathbf{dst}(e)$ をそれぞれ**始点ノード**，**終点ノード**と呼ぶ．また，ノード集合を $V(\mathcal{G}) = \mathcal{V}$，エッジ集合を $E(\mathcal{G}) = \mathcal{E}$ と表記する．ここで，本稿で用する主な記号を表 2 に示す．

3.3 PG のスキーマ

PG のスキーマはインスタンスに現れ得るノードおよびエッジの構造的制約を抽象的に記述したものである．本研究では，実用的なプロパティグラフスキーマに必要な要素を体系的に整理した PG-Schema [15] に準拠したスキーマモデルを採用する．

本稿では，スキーマを構成する基本単位として**ノード型**および**エッジ型**を導入する．ノード型は特定のラベル集合とプロパティキー集合を持つノードの型を表し，エッジ型はこれらに加えて始点および終点となるノード型が定められたエッジの型を表す．以降，ノード型とエッジ型を総称して**オブジェクト型**，あるいは単に**型**と呼ぶ．

定義 3.3.1 (PG のスキーマ)**.** ラベル集合を $\mathcal{L}$，プロパティキー集合を $\mathcal{K}$ とする．PG のスキーマを $\mathcal{S} = (\mathcal{V}_T, \mathcal{E}_T, \lambda, \kappa, \sigma, \mu, \rho)$ と定義する．ここで $\mathcal{V}_T$ はノード型の有限集合，$\mathcal{E}_T$ はエッジ型の有限集合であり，$(\mathcal{V}_T \cap \mathcal{E}_T) = \varnothing$ とする．λ および κ は定義 3.2.1 と同様に，各オブジェクト型のラベル集合およびプロパティキー集合を返す関数である．また，$\mu : (\mathcal{V}_T \cup \mathcal{E}_T) \times \mathcal{K} \to \{\mathbf{True}, \mathbf{False}\}$ は各プロパティが必須か否かを返す関数であり，$\rho : (\mathcal{V}_T \cup \mathcal{E}_T) \to 2^{(\mathcal{V}_T \cup \mathcal{E}_T)}$ は定義 3.4.1 の継承関係に基づく親オブジェクト型を返す関数である．

本稿では，オブジェクト型 o の必須プロパティ集合および任意プロパティ集合を $\mathbf{req}(o) = \{k \in \mathcal{K} \mid \mu(o,k) = \mathbf{True}\}, \mathbf{opt}(o) = \{k \in \mathcal{K} \mid \mu(o,k) = \mathbf{False}\}$ と表記する．また任意プロパティを図示する際は，図 3 に示すようにプロパティ名の末尾に「?」を付与して表記する．

なお PG-Schema では定義 3.3.1 に加えてプロパティの型やエッジ多重度などの制約も定義されているが，本稿では議論を簡潔にするためにこれらの制約を定義から省略する．これらの制約は，4.3 節で述べる手順に軽微な拡張を施すことで本稿の評価手法に組み込むことが可能である．

3.4 オブジェクト型の継承

PG のスキーマでは複数の型が共通のラベルやプロパティを持つことが多く，これらを個別に定義するとスキーマが冗長になり保守性が低下する．この問題を避けるため，PG-Schema [15] では種々の制約を再利用する仕組みとして型の継承が導入されている．本稿でもこの考え方に基づき，オブジェクト型の継承関係を明示的に定義する．

定義 3.4.1 (オブジェクト型の継承関係)**.** オブジェクト型の集合 $\mathcal{V}_T \cup \mathcal{E}_T$ 上の二項関係 $\prec \subseteq (\mathcal{V}_T \times \mathcal{V}_T) \cup (\mathcal{E}_T \times \mathcal{E}_T)$ を **継承関係**と呼ぶ．$o_1 \prec o_2$ のとき，o_1 を **子オブジェクト型**，o_2 を **親オブジェクト型**と呼ぶ．

また継承関係 $\prec$ は非巡回であり，多重継承を許容する．加えて継承関係は推移律を満たす．すなわち，$o_1 \prec o_2$ かつ $o_2 \prec o_3$ が成り立つとき，o_1 は o_3 を継承するものと解釈する．

子オブジェクト型 o_1 が親オブジェクト型 o_2 を継承する場合，o_1 は o_2 によって課されるすべての構造的制約を満たすことを意味する．ただし例外的に，o_1 と o_2 の間で矛盾する制約が定義されている場合，o_2 の制約は o_1 に継承されない．例えば，親ノード型 o_2 があるプロパティ k を必須と定義し，子ノード型 o_1 が同じプロパティ k を任意と定義している場合，o_1 は k を任意プロパティとして扱う．

本稿では，継承関係を図示する際には図 2 および図 3 に示すように，先端が $\triangleright$ の矢印で表す．

4 提案手法

4.1 基本アイデア

本研究では，PG スキーマの品質をインスタンスに観測されるデータ構造をどの程度表現できているかという**網羅性**と，不要な冗長性を避け，人間にとって理解・保守しやすい形で記述されているかという**簡潔性**の 2 つの観点で捉える．この考え方に基づき，網羅性は「インスタンスに現れる構造がスキーマによってどの程度説明されているか」を評価する指標として定義し，簡潔性は「インスタンス構造の説明に本質的に寄与しない冗長な要素がどの程度抑制されているか」を評価する指標として定義する．最終的にこれら二つの指標を調和平均として統合することで，一方に偏ったスキーマを過大評価しない単一の品質指標である C2 スコアを導入する．

4.2 提案手法の全体像

本研究の目的は，インスタンス $\mathcal{G}$ に対するスキーマ $\mathcal{S}^*$ の品質を定量的に評価することである．そのためには，インスタンスに現れるデータ構造とスキーマが表現する構造とを同一の比較単位で対応付けた上で，網羅性と簡潔性をそれぞれ評価する必要がある．提案手法は，この比較単位の不一致を解消するための前処理と網羅性・簡潔性を算出する評価ステップから構成される．提案手法の全体像を図 4 に示す．

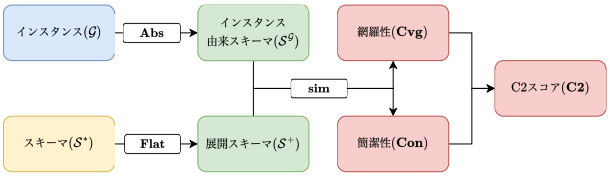


図 4: 提案手法の全体像

インスタンス $\mathcal{G}$ は個々のオブジェクトの集合として与えられる一方，スキーマはオブジェクト型の集合として与えられるため，両者の表現粒度は本質的に異なり，直接比較することができない．そこで本研究ではインスタンスに観測される要素を構造的特徴に基づいて抽象化し，**インスタンス由来スキーマ** $\mathcal{S}^{\mathcal{G}} = \mathbf{Abs}(\mathcal{G})$ を構成する．この操作により，インスタンス側をオブジェクト型の集合として扱うことが可能となる．

しかし，比較対象となるスキーマ側が継承関係を暗黙に含んだままである場合，依然として比較粒度は一致しない．PG のスキーマでは，属性や制約が親オブジェクト型に定義され子オブジェクト型には暗黙的に継承されることが多く，表層的な型定義のみを比較するとスキーマが本来表現可能な構造を過小評価してしまうおそれがある．そこで本研究では，継承によって暗黙的に与えられる属性や制約を各オブジェクト型に明示的に反映することで，インスタンス由来スキーマとの比較粒度を一致させる操作を導入する．この操作を**継承関係の展開**と呼び，この操作を施したスキーマを $\mathcal{S}^+ = \mathbf{Flat}(\mathcal{S}^*)$ として導入する．

以上の前処理により，インスタンス側を $\mathcal{S}^{\mathcal{G}}$，スキーマ側を $\mathcal{S}^+$ として両者を同一の比較単位で扱うことが可能となる．この上で網羅性および簡潔性の評価，および両者を統合した C2 スコアの算出を行う．以降では，これら各ステップについて順に形式的定義と評価方法を述べる．

4.3 インスタンス由来スキーマの生成

本研究ではインスタンス $\mathcal{G}$ に現れるノードおよびエッジをその構造的特徴に基づいて分類し，インスタンス由来スキーマ $\mathcal{S}^{\mathcal{G}}$ を構成する．この抽象化操作を $\mathcal{S}^{\mathcal{G}} = \mathbf{Abs}(\mathcal{G})$ と定義する．

$\mathbf{Abs}(\cdot)$ はノードについてはラベルが完全に一致するもの同士を，エッジについては（エッジラベル，始点ノードラベル，終点ノードラベル）のタプルが一致するもの同士をそれぞれ部分集合に分割し，各部分集合から対応するノード型およびエッジ型を生成する操作である．各オブジェクト型に付随するプロパティについては，部分集合内のすべてのオブジェクトに共通して出現するものを必須，そうでないものを任意として区別する．

なおこの操作は決定的であり，同一のインスタンス $\mathcal{G}$ に対して $\mathbf{Abs}(\mathcal{G})$ は常に一意に定まる．なお，$\mathbf{Abs}(\cdot)$ の具体的なアルゴリズムは付録 1.1 に示す．

4.4 継承関係の展開

本研究では，継承によって暗黙的に与えられる属性や制約を各オブジェクト型に明示的に反映するための操作として継承関係の展開を行う．以降，評価対象となる元のスキーマを $\mathcal{S}^*$，継承によって暗黙的に含まれる情報を各オブジェクト型に明示化したスキーマを $\mathcal{S}^+$ と表記し，この操作を $\mathcal{S}^+ = \mathbf{Flat}(\mathcal{S}^*)$ と定義する．$\mathbf{Flat}(\cdot)$ は各オブジェクト型について，そのすべての祖先型に定義された属性およびエッジの場合は端点制約を推移的にコピーし，当該オブジェクト型に明示的に付与する操作である．またエッジ型については，当該エッジ型の始点（終点）型として指定されたノード型についてそのすべての子孫型を列挙し，列挙された始点・終点型の組に対して対応するエッジ型を生成することで，暗黙的に含まれる端点制約を明示化する．

なお本研究では定義 3.4.1 に示すように継承関係が有向非巡回であることを前提としているため，$\mathbf{Flat}(\cdot)$ は同一の $\mathcal{S}^*$ に対して常に一意な $\mathcal{S}^+$ を与える．また $\mathbf{Flat}(\cdot)$ の形式的定義および具体的なアルゴリズムは付録 1.2 に示す．

4.5 オブジェクト型の類似度

オブジェクト型の類似度は (i) ラベル集合の一致度，(ii) プロパティ集合の一致度，および (iii) エッジ型の場合は端点ノード型の一致度に基づき定義する．

定義 4.5.1 (ノード型の類似度)**.** ノード型 $v^{\mathcal{G}} \in \mathcal{S}^{\mathcal{G}}$ と $v^+ \in \mathcal{S}^+$ の類似度を以下のように定義する．

$$\mathbf{sim}_V(v^{\mathcal{G}}, v^+) = \begin{cases} 0 & \lambda(v^{\mathcal{G}}) \cap \lambda(v^+) = \varnothing, \\ D_{\mathrm{attr}}(v^{\mathcal{G}}, v^+) & \text{otherwise.} \end{cases}$$

$$D_{\mathrm{attr}}(v^{\mathcal{G}}, v^+) = \alpha\, D_\lambda(v^{\mathcal{G}}, v^+) + (1-\alpha)\, D_\kappa(v^{\mathcal{G}}, v^+)$$

ただし $\alpha \in [0,1]$ はラベル一致度とプロパティ一致度の重みを調整するパラメータである．また D_λ および D_κ は Dice 係数に基づく一致度であり，以下のように定義する．

$D_\lambda(v^{\mathcal{G}}, v^+) = D(\lambda(v^{\mathcal{G}}), \lambda(v^+))$

$D_\kappa(v^{\mathcal{G}}, v^+) = \dfrac{D(\mathbf{req}(v^{\mathcal{G}}), \mathbf{req}(v^+)) + D(\mathbf{opt}(v^{\mathcal{G}}), \mathbf{opt}(v^+))}{2}$

ただし $D(A,B) = \dfrac{2|A \cap B|}{|A| + |B|}$である．

定義 4.5.1 では型同士のラベル集合が一致しない場合に類似度を 0 とすることで，意味的に明らかに異なる型が偶然に共通するプロパティキーを持つことによる誤った高類似度評価を防止している．

定義 4.5.2 (エッジ型の類似度)**.** エッジ型 $e^{\mathcal{G}} \in \mathcal{S}^{\mathcal{G}}$ と $e^+ \in \mathcal{S}^+$ の類似度を以下のように定義する．

$$\mathbf{sim}_E(e^{\mathcal{G}}, e^+) = \begin{cases} 0, & \lambda(e^{\mathcal{G}}) \cap \lambda(e^+) = \varnothing, \\ D_{\mathrm{edge}}(e^{\mathcal{G}}, e^+), & \text{otherwise.} \end{cases}$$

$$D_{\mathrm{edge}}(e^{\mathcal{G}}, e^+) = \beta\, D_{\mathrm{attr}}(e^{\mathcal{G}}, e^+) + (1-\beta)\, D_\sigma(e^{\mathcal{G}}, e^+)$$

ただし $\beta \in [0,1]$ はエッジ型の属性の一致度と端点ノード型の一致度の重みを調整するパラメータである．また D_σ は端点ノード型の一致度であり，以下のように定義する．

$$D_\sigma(e^{\mathcal{G}}, e^+) = \frac{\mathbf{sim}_V(\mathbf{src}(e^{\mathcal{G}}), \mathbf{src}(e^+)) + \mathbf{sim}_V(\mathbf{dst}(e^{\mathcal{G}}), \mathbf{dst}(e^+))}{2}$$

4.6 網羅性（Coverage）

網羅性は，インスタンスに観測される各オブジェクト型に対しスキーマ中で最も類似する型との一致度を用いることで，スキーマがインスタンス構造をどの程度説明できているかを評価する指標である．インスタンス由来の型を十分に表現できる型がスキーマに存在する場合には高い類似度が得られ，対応する型がスキーマに存在しない場合には類似度は低い値となる．このような対応関係に基づく類似度を集約することで，スキーマがインスタンス構造をどの程度網羅しているかを定量化する．

定義 4.6.1 (網羅性)**.** 評価対象のスキーマを $\mathcal{S}^*$，インスタンスを $\mathcal{G}$ とする．ノード網羅性およびエッジ網羅性を

$$\mathbf{Cvg}_V(\mathcal{G}, \mathcal{S}^*) = \frac{1}{|V(\mathcal{S}^{\mathcal{G}})|} \sum_{v^{\mathcal{G}} \in V(\mathcal{S}^{\mathcal{G}})} \max_{v^+ \in V(\mathcal{S}^+)} \mathbf{sim}_V(v^{\mathcal{G}}, v^+)$$

$$\mathbf{Cvg}_E(\mathcal{G}, \mathcal{S}^*) = \frac{1}{|E(\mathcal{S}^{\mathcal{G}})|} \sum_{e^{\mathcal{G}} \in E(\mathcal{S}^{\mathcal{G}})} \max_{e^+ \in E(\mathcal{S}^+)} \mathbf{sim}_E(e^{\mathcal{G}}, e^+)$$

と定義する．ただし，$\mathcal{S}^{\mathcal{G}} = \mathbf{Abs}(\mathcal{G})$，$\mathcal{S}^+ = \mathbf{Flat}(\mathcal{S}^*)$ である．

4.7 簡潔性（Concision）

簡潔性は，インスタンス構造の説明に本質的に寄与しない冗長なオブジェクト型がスキーマにどの程度含まれているかを測る指標である．簡潔性の計算では，各スキーマオブジェクト型を 1 つずつ除去し，その除去が網羅性に与える影響に基づいて冗長性を判定する．除去しても網羅性の低下が十分に小さい場合，当該オブジェクト型はインスタンス構造の説明に本質的には寄与しておらず冗長であるとみなす．このような冗長オブジェクト型の割合に基づいて簡潔性を定義する．

まず，「網羅性の低下が十分に小さい」ことを定量化するため寄与下限を定義する．

定義 4.7.1 (寄与下限)**.** 評価対象のスキーマを $\mathcal{S}^*$，インスタンスを $\mathcal{G}$ とし，$\mathcal{S}^+ = \mathbf{Flat}(\mathcal{S}^*)$ とする．ノード網羅性およびエッジ網羅性に対する寄与下限を

$$\epsilon_V = \frac{\mathbf{Cvg}_V(\mathcal{G}, \mathcal{S}^*)}{|V(\mathcal{S}^+)|} \cdot \gamma,$$

$$\epsilon_E = \frac{\mathbf{Cvg}_E(\mathcal{G}, \mathcal{S}^*)}{|E(\mathcal{S}^+)|} \cdot \gamma$$

と定義する．ただし $\gamma \in (0,1]$ は冗長性判定の厳しさを調整するパラメータである．

次に，定義 4.7.1 で定義した寄与下限を用いて冗長オブジェクト型を定義する．

定義 4.7.2 (冗長オブジェクト型)**.** オブジェクト型 o を評価対象スキーマ $\mathcal{S}^*$ から除去したときのノード網羅性およびエッジ網羅性の低下量を以下のように定義する．

$$\Delta_V(o) = \mathbf{Cvg}_V(\mathcal{G}, \mathcal{S}^*) - \mathbf{Cvg}_V(\mathcal{G}, \mathcal{S}^* \setminus \{o\})$$

$$\Delta_E(o) = \mathbf{Cvg}_E(\mathcal{G}, \mathcal{S}^*) - \mathbf{Cvg}_E(\mathcal{G}, \mathcal{S}^* \setminus \{o\})$$

このとき，$\Delta_V(o) < \epsilon_V$ かつ $\Delta_E(o) < \epsilon_E$ を満たすオブジェクト型 o を**冗長オブジェクト型**と定義する．さらに冗長オブジェクト型の集合を以下のように定義する．

$$R_V(\mathcal{G}, \mathcal{S}^*) = \{\, v^* \in V(\mathcal{S}^*) \mid \Delta_V(v^*) < \epsilon_V \wedge \Delta_E(v^*) < \epsilon_E \,\}$$

$$R_E(\mathcal{G}, \mathcal{S}^*) = \{\, e^* \in E(\mathcal{S}^*) \mid \Delta_V(e^*) < \epsilon_V \wedge \Delta_E(e^*) < \epsilon_E \,\}$$

ここで，オブジェクト型の除去操作は o がノード型である場合には当該ノード型とそれを端点として参照するエッジ型を同時に除去し，o がエッジ型である場合には当該エッジ型のみを除去する．このため，ノード型を除去した場合には対応するエッジ型も失われることでエッジ網羅性が低下し得る．一方，エッジ型を除去した場合でもそれが継承関係を表すエッジ型であるときには属性や制約の継承が行われなくなるため，展開後スキーマにおけるノード型の属性集合が変化し，ノード網羅性が低下し得る．したがって，冗長オブジェクト型の判定ではノード型・エッジ型の別に関わらず Δ_V と Δ_E の両方を考慮する．

最後に，冗長オブジェクト型の割合に基づいて簡潔性を定義する．

定義 4.7.3 (簡潔性)**.** 評価対象のスキーマを $\mathcal{S}^*$，インスタンスを $\mathcal{G}$ とする．ノード簡潔性およびエッジ簡潔性を以下のように定義する．

$$\mathbf{Con}_V(\mathcal{G}, \mathcal{S}^*) = 1 - \frac{|R_V(\mathcal{G}, \mathcal{S}^*)|}{|V(\mathcal{S}^*)|}$$

$$\mathbf{Con}_E(\mathcal{G}, \mathcal{S}^*) = 1 - \frac{|R_E(\mathcal{G}, \mathcal{S}^*)|}{|E(\mathcal{S}^*)|}$$

4.8 C2 スコア

C2 スコアは網羅性と簡潔性を単一の指標として統合するための指標であり，いずれか一方が低い場合にスコアが低下するように両者の調和平均として定義する．

定義 4.8.1 (C2 スコア). 評価対象のスキーマを $\mathcal{S}^*$，インスタンスを $\mathcal{G}$ とする．ノード C2 スコアおよびエッジ C2 スコアを以下のように定義する．

$$\mathbf{C2}_V(\mathcal{G},\mathcal{S}^*) = \frac{2\cdot \mathbf{Cvg}_V(\mathcal{G},\mathcal{S}^*)\cdot \mathbf{Con}_V(\mathcal{G},\mathcal{S}^*)}{\mathbf{Cvg}_V(\mathcal{G},\mathcal{S}^*)+\mathbf{Con}_V(\mathcal{G},\mathcal{S}^*)},$$

$$\mathbf{C2}_E(\mathcal{G},\mathcal{S}^*) = \frac{2\cdot \mathbf{Cvg}_E(\mathcal{G},\mathcal{S}^*)\cdot \mathbf{Con}_E(\mathcal{G},\mathcal{S}^*)}{\mathbf{Cvg}_E(\mathcal{G},\mathcal{S}^*)+\mathbf{Con}_E(\mathcal{G},\mathcal{S}^*)}$$

計算量 提案手法全体の計算量は $O\big(|V(\mathcal{G})|+|E(\mathcal{G})|+|V(\mathcal{S}^{\mathcal{G}})||V(\mathcal{S}^+)|+|E(\mathcal{S}^{\mathcal{G}})||E(\mathcal{S}^+)|\big)$ である．アルゴリズムの擬似コードと簡単な計算量解析は付録に示す．

4.9 提案指標の計算例

図 1 に示す PG インスタンスを $\mathcal{G}$，図 2 に示すスキーマを評価対象スキーマ $\mathcal{S}^*$ とし，提案指標の計算例を示す．以下ではノードの網羅性・簡潔性・C2 スコアについての計算過程を示す．パラメータは $\alpha=0.5$，$\beta=0.5$，$\gamma=0.15$ とする．

(1) インスタンス由来スキーマの生成 インスタンス $\mathcal{G}$ に観測される構造を抽象化し，$\mathcal{S}^{\mathcal{G}}=\mathbf{Abs}(\mathcal{G})$ を構成する．得られたインスタンス由来スキーマを図 3 に示す [1]．

本例では，インスタンス $\mathcal{G}$ のノードはラベル集合に基づいて $\{v_1,v_2\}$，$\{v_3,v_4\}$，$\{v_5,v_6\}$ の 3 つの部分集合に分割され，それぞれからノード型 $v_1^{\mathcal{G}},v_2^{\mathcal{G}},v_3^{\mathcal{G}}$ が生成される．エッジについても同様に（始点ラベル集合，エッジラベル，終点ラベル集合）のタプルに基づいて分割され，4 種類のエッジ型が生成される．

(2) 継承関係の展開 評価対象スキーマ $\mathcal{S}^*$ に対して継承関係を展開し，$\mathcal{S}^+=\mathbf{Flat}(\mathcal{S}^*)$ を構成する．$\mathcal{S}^*$ を展開したスキーマを図 5 に示す．この操作により，親型から暗黙的に継承されていたラベルやプロパティおよび端点制約が明示化される．

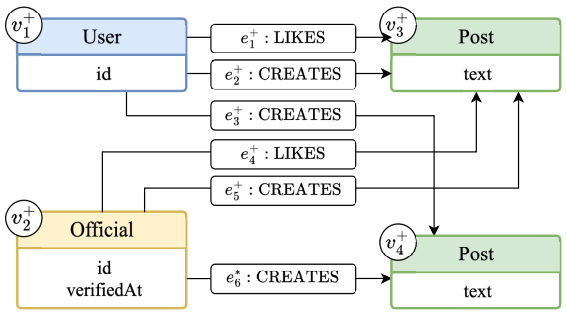


図 5: 展開後スキーマ $\mathcal{S}^+$

(3) 類似度の計算 定義 4.5.1 に基づき，(1) のインスタンス由来スキーマ $\mathcal{S}^{\mathcal{G}}$ の各ノード型と (2) の展開後スキーマ $\mathcal{S}^+$ の各ノード型との間で類似度を計算する．表 3 にノード型の類似度行列を示す．

(4) ノード網羅性の計算 定義 4.6.1 および表 3 に基づき，ノード網羅性は以下のように計算される．

1：1 節では図 3 は Xue らの手法によって抽出されたスキーマとして紹介しているが，本節ではインスタンス $\mathcal{G}$ に対して **Abs** を適用して得られるインスタンス由来スキーマとして用いている．本稿では両者が一致しているため同一の図を用いているが，一般には両者が一致するとは限らない．

表 3: ノード型の類似度行列

	v_1^+	v_2^+	v_3^+	v_4^+
$v_1^{\mathcal{G}}$	**0.75**	0.5	0.0	0.0
$v_2^{\mathcal{G}}$	0.5	**0.75**	0.0	0.0
$v_3^{\mathcal{G}}$	0.0	0.0	**1.0**	**1.0**

表 4: ノード型の冗長性評価

除去したノード型	Δ_V	Δ_E	$\Delta_V<\epsilon_V$	$\Delta_E<\epsilon_E$	冗長ノード型
v_1^*	0.33	0.93	False	False	False
v_2^*	0.08	0.03	False	False	False
v_3^*	0.00	0.47	True	False	False
v_4^*	0.00	0.00	True	True	True

$$\mathbf{Cvg}_V(\mathcal{G},\mathcal{S}^*)=\frac{1}{3}(0.75+0.75+1.0)=\boxed{0.83}$$

(5) ノード簡潔性の計算 まず，定義 4.7.1 に基づき寄与下限を計算するとそれぞれ $\epsilon_V=0.031$，$\epsilon_E=0.023$ となる．次に，各ノード型を $\mathcal{S}^*$ から除去した場合の網羅性低下量 Δ_V,Δ_E を計算し，冗長性を判定する．表 4 に各ノード型の判定結果を示す．表より，ノード型 v_4^* のみがノード網羅性およびエッジ網羅性のいずれに対しても寄与が小さく，冗長な型と判定される．したがって定義 4.7.3 に基づき，ノード簡潔性は以下のように計算される．

$$\mathbf{Con}_V(\mathcal{G},\mathcal{S}^*)=1-\frac{1}{4}=\boxed{0.75}$$

(5) ノード C2 スコアの計算 定義 4.8.1 に基づき，ノード C2 スコアは以下のように計算される．

$$\mathbf{C2}_V(\mathcal{G},\mathcal{S}^*)=\frac{2\cdot 0.83\cdot 0.75}{0.83+0.75}=\boxed{0.79}$$

5 評価実験

5.1 評価実験の概要

本研究では，提案指標がプロパティグラフのスキーマ品質評価指標として実運用に耐えうるかを検証するため，以下に示す 4 つのリサーチクエスチョン（RQ）を設定し，それぞれに対応する評価実験を行う．

RQ1：大規模グラフに対しても実行可能であるか 大規模なプロパティグラフに対しても，提案指標が現実的な計算時間で実行可能であるかを検証する．複数の実データセットを用い，ノード数およびエッジ数の増加に伴う評価時間のスケーリング挙動を測定する．

RQ2：スキーマに含まれる誤りに応じて評価値が変化するか スキーマに欠損や冗長を段階的に導入した場合に，誤りの種類および量に応じて評価値が適切に変化するかを検証する．

RQ3：異なるスキーマ間の構造的差異を区別できるか 同一データセットに対して複数の既存スキーマ抽出手法を適用し，生成されたスキーマ間の構造的差異が提案指標によって区別可能かを検証する．

表 5: データセット一覧

データセット	カテゴリ	ノード数	エッジ数	ノードラベル数	エッジラベル数	ノードプロパティ数	エッジプロパティ数	正解スキーマ	継承	任意プロパティ	複数ラベル	実データ／人工データ
FinDKG [29]	知識グラフ	13,645	223,983	12	15	2	1	–	–	–	–	実データ
LDBC-SNB [30]	ソーシャルグラフ	30,191	44,742	14	15	16	3	✓	✓	✓	✓	人工データ
NeuPrint MB6 [31]	生物学	486,267	961,571	5	3	32	3	✓	✓	✓	✓	実データ
IT-Management [32]	IT	83,847	181,995	17	12	17	0	–	✓	✓	✓	人工データ
Northwind [33]	物流	1,035	3,139	5	4	38	5	✓	–	–	–	人工データ
Spotify [34]	音楽	64,458	155,430	4	4	16	0	✓	–	–	–	実データ
Steam(this work)	ゲーム	229,372	438,264	28	9	20	4	✓	✓	✓	✓	実データ
TPC-H [35]	物流	78,805	205,150	7	7	30	3	✓	–	–	–	人工データ
Twitter [36]	ソーシャルグラフ	43,325	56,403	6	8	14	0	–	✓	✓	–	実データ
WordNet [37]	知識グラフ	689,929	1,013,075	12	31	35	0	–	✓	✓	✓	実データ

RQ4：人間による品質判断と評価結果が整合するか ユーザスタディを通じて，人間によるスキーマ品質の主観的な順位付けと提案指標による評価結果との整合性を検証する．

実験環境 本実験は，Apple M1 Ultra（20 コア）および 128 GB メモリを搭載した Mac Studio 上で実行した．OS には macOS 15.5 を使用し，グラフデータベースは Neo4j Enterprise 2025.11.2 を用いた．アルゴリズムは Python 3.12.4 で実装し，すべての実験アルゴリズムはシングルスレッドで実行した．

データセット 本実験では，表 5 に示す 10 種類のデータセットを用いた．本実験では実世界データと人工データの双方を使用し，また継承関係，任意プロパティ，マルチラベルといった PG 特有の構造を含むデータセットとそれらを含まないデータセットの双方を用いることで，データ構造の多様性を広くカバーしている．なお，Steam データセットは本研究のために著者らが独自に収集・構築したものである．

5.2 RQ1：大規模グラフに対する実行可能性

概要 本実験では，入力グラフの規模を段階的に拡大した場合の実行時間を測定し，提案指標が大規模なプロパティグラフに対しても実用的な時間で評価を実行可能であるかを検証する．提案手法の計算量は $O\big(|V(\mathcal{G})|+|E(\mathcal{G})|+|V(\mathcal{S}^{\mathcal{G}})||V(\mathcal{S}^{+})|+|E(\mathcal{S}^{\mathcal{G}})||E(\mathcal{S}^{+})|\big)$ である．一般にスキーマサイズはインスタンスサイズに比べて十分小さいため，実行時間は主として $|V(\mathcal{G})|$ および $|E(\mathcal{G})|$ の走査コストに支配され，入力規模に対して概ね線形に増加すると期待される．

データセットは，スキーマ構造を固定したままインスタンスの規模のみを制御可能な LDBC-SNB データセットを用いる．インスタンス規模を調整する変数 scale factor を段階的に変更して異なる規模のデータセットを生成し，各データセットに対して Lbath らのスキーマ抽出手法 [2] により得られたスキーマを入力として提案手法によるスキーマ評価を実行した．実行時間にはデータ読み込みおよびスキーマ評価計算に要する時間を含み，スキーマ抽出に要する時間は含まない．各条件について 10 回実行し平均実行時間を記録した．

実験結果 表 6 に各 scale factor におけるノード数，エッジ数，および平均実行時間を示す．表より，提案手法は入力規模に対してほぼ線形にスケールし，数千万〜2 億エッジ規模のグラフに対してもスキーマ品質評価を現実的な計算時間で実行可能であることが示された．またこの結果は理論的な計算量解析とも整合している．

表 6: スキーマ評価の実行時間

Scale Factor	ノード数	エッジ数	実行時間 [秒]
0.1	0.4M	2M	9
0.3	1M	6M	15
1.0	4M	21M	29
3.0	11M	63M	118
10.0	34M	200M	354

5.3 RQ2：スキーマ誤りに対する評価値の変化

概要 本実験では，提案指標がスキーマに含まれる誤りの種類および程度に応じた直感的に妥当な評価値の変化を示すかを検証する．本実験では正解スキーマに対して人為的に誤りを挿入し，誤り数の増加に伴う評価値の挙動を観測する．

NeuPrint MB6 および Steam データセットに付属する正解スキーマを起点とし，(a) オブジェクト型の削除，(b) ラベルの削除，(c) プロパティの削除，(d) オブジェクト型の重複追加，(e) エッジ方向の反転，(f) (a) – (e) からのランダム選択の計 6 種類の誤り挿入操作を行う．ノード型に対しては (a) – (d) および (f)，エッジ型に対しては (a) および (c) – (f) を適用した．なお Neo4j はラベルなしエッジを許容しないため，(b) はエッジに対して適用していない．誤り数は 0 から 5 までの 6 段階とし，各（誤り種別，誤り数）の組に対して 100 個のスキーマを生成・評価し，評価値の平均を算出した．

実験結果 図 6〜図 9 に NeuPrint MB6 および Steam データセットにおける誤り数および誤り種別ごとのスキーマ評価結果を示す．図より，いずれのデータセットにおいても正解スキーマ（誤り数 0）が最も高いスコアを示し，誤り数の増加に伴ってスコアが単調に低下する傾向があることが分かる．したがって，提案指標はスキーマの誤り数に応じて直感的に妥当な形で評価値を低下させることが確認された．

また誤りの種類によってスコア低下の度合いが異なることも確認できる．例えばノードプロパティの欠損に対しては比較的緩やかなスコアの低下が見られる一方，ノードラベルの欠損に対しては急激なスコア低下が生じている．一般的にラベルの欠損はプロパティの欠損に比べてオブジェクト型の識別力を大きく損なうため，提案手法はスキーマに含まれる誤りの性質に応

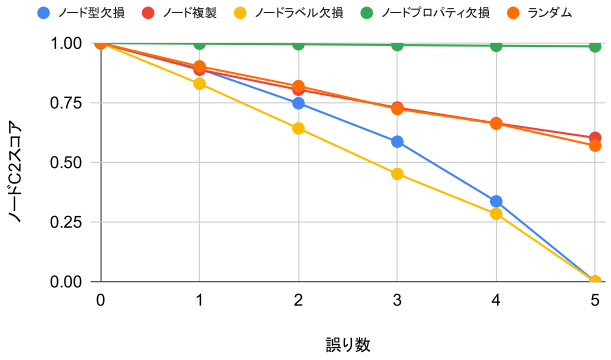


図 6: ノード C2 スコア (MB6)

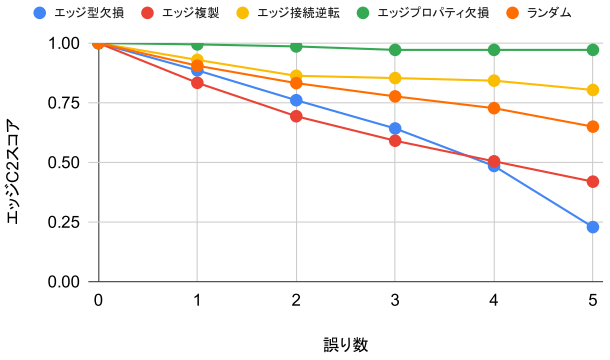


図 7: エッジ C2 スコア (MB6)

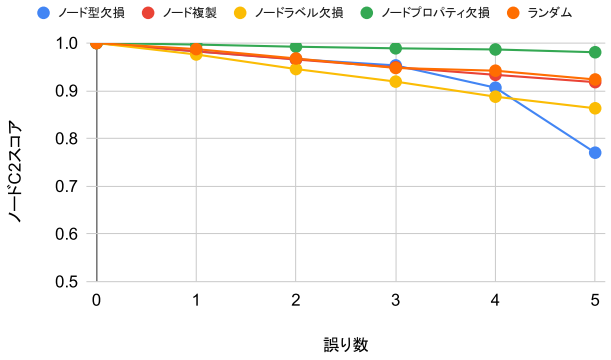


図 8: ノード C2 スコア (Steam)

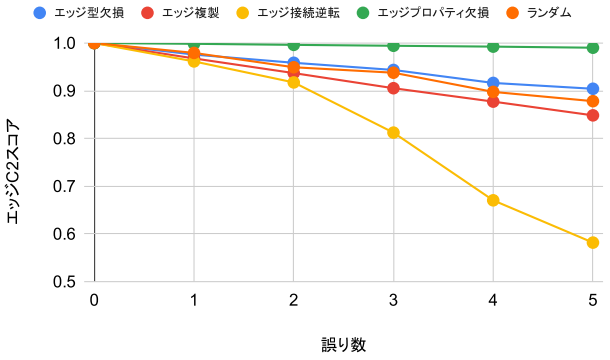


図 9: エッジ C2 スコア (Steam)

じて適切に評価値を変化させていることが示唆される．

さらにオブジェクト型やラベルの欠損では主として網羅性が低下するのに対し，冗長なオブジェクト型の追加では簡潔性が大きく低下する傾向が観測された．これらの結果は，提案指標が欠陥の種類に応じて網羅性と簡潔性を適切に区別して評価できていることを示している．

以上より，提案手法は正しいスキーマに対して高い評価を与え，誤ったスキーマに対しては誤り数および誤りの性質に応じて直感的に妥当な形で各評価値を低下させることが確認された．

5.4 スキーマ抽出手法間比較実験

概要 本実験では異なるスキーマ抽出手法によって得られたスキーマ間の評価値の差が，スキーマの構造的特徴として人間にとって説明可能な形で現れるかを検証する．なお本実験は抽出手法の優劣を決定することを目的とするものではない．

本実験では表 5 に示す全てのデータセットに対して類似度ベース手法 [1]（以下，手法 1），ルールベース手法 [2]（手法 2），および階層型クラスタリング手法 [3]（手法 3）の 3 種類のスキーマ抽出手法を適用し，得られたスキーマを評価した．

実験結果 各データセットおよび各手法で抽出したスキーマの評価結果を表 7 に示す．表より，手法 3 による抽出スキーマは他の手法と比較して全体的に網羅性がやや低い傾向が確認できる．これは，手法 3 がインスタンス全体を走査せずサンプリングに基づいてスキーマを抽出するため，希少なラベルやプロパティがスキーマに反映されにくいことに起因すると考えられる．例えば WordNet データセットでは，手法 1 および手法 2 が 12 種類のノードラベルを抽出しているのに対し手法 3 は 10 種類にとどまっており，このことがノード網羅性の低下の一因として評価値に反映されていた．

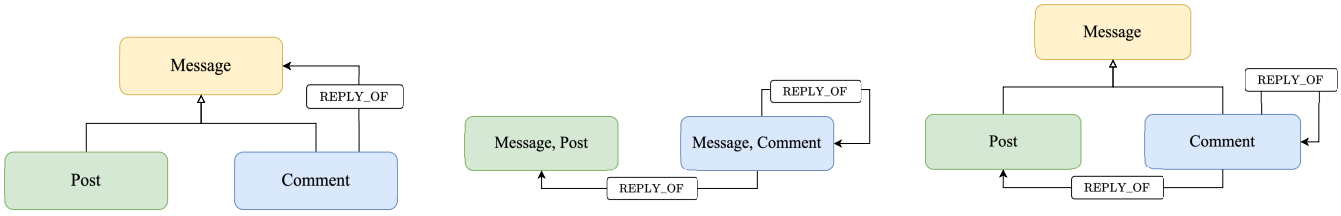
一方，手法 1 および手法 2 による抽出スキーマは高い網羅性とノード簡潔性を示す一方で，エッジ簡潔性が相対的に低くなる傾向が観測された．この傾向も各手法の設計上の特性と整合している．例えば手法 2 では継承関係を抽出するものの，親オブジェクト型へのエッジの集約を行わないため意味的に同一のエッジが子ノード型の組ごとに分散して定義されやすい．また手法 1 は継承関係を扱わないため親型による関係の集約が行われず，結果としてエッジ型の冗長性が生じやすい．図 10 は LDBC-SNB データセットにおける正解スキーマと抽出スキーマの一部を示したものである（ただしプロパティは省略している）．正解スキーマでは `Message` に対して集約的に定義されている `REPLY_OF` エッジが，手法 1 および手法 2 による抽出スキーマでは `Post` や `Comment` ごとに分散して定義されており，これがエッジ簡潔性の低下として評価値に反映されていることが分かる．以上より，提案指標はスキーマ抽出手法の設計上の特性やそれに起因するスキーマ構造の違いを評価値として反映できることが確認された．

5.5 ユーザスタディ

概要 本ユーザスタディの目的は (1) 人間がスキーマ品質を評価する際に重視する欠陥の種類とその相対的重要度を明らかにす

表 7: 網羅性，簡潔性，C2 スコアの評価結果

データセット	手法 1						手法 2						手法 3					
	網羅性		簡潔性		C2 スコア		網羅性		簡潔性		C2 スコア		網羅性		簡潔性		C2 スコア	
	V	E	V	E	V	E	V	E	V	E	V	E	V	E	V	E	V	E
FindKG	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.88	1.00	1.00	1.00	0.93
LDBC-SNB	1.00	1.00	1.00	0.44	1.00	0.61	1.00	1.00	1.00	0.56	1.00	0.72	0.98	0.96	0.77	0.45	0.86	0.61
NeuPrint MB6	1.00	0.88	1.00	0.33	1.00	0.48	0.98	0.99	1.00	0.40	0.99	0.57	0.78	0.93	0.50	0.35	0.61	0.51
IT-Management	1.00	1.00	1.00	0.41	1.00	0.58	1.00	1.00	0.96	0.48	0.98	0.65	0.94	0.89	0.76	0.36	0.84	0.52
Northwind	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.97	1.00	1.00	1.00	0.98
Spotify	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Steam	1.00	1.00	1.00	0.04	1.00	0.08	1.00	1.00	0.97	0.19	0.98	0.31	0.99	0.93	0.77	0.02	0.87	0.03
TPC-H	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.98	1.00	1.00	1.00	0.99
Twitter	1.00	1.00	1.00	0.64	1.00	0.78	0.94	0.96	1.00	0.42	0.97	0.58	0.93	0.80	0.42	0.23	0.58	0.35
WordNet	0.93	0.88	1.00	0.31	0.97	0.46	0.86	0.93	0.91	0.34	0.88	0.50	0.71	0.84	0.58	0.09	0.64	0.16



(a) 正解スキーマの一部　(b) 手法 1 による抽出スキーマの一部　(c) 手法 2 による抽出スキーマの一部

図 10: LDBC データセットにおける正解スキーマと抽出スキーマの比較（プロパティは省略）

ること，および (2) 提案手法による評価結果が人間の集合的判断とどの程度一致するかを定量的に検証することである．

ユーザスタディでは，12 種類の PG インスタンスそれぞれに対し正解スキーマ 1 件と，意図的に欠陥を含めた誤りスキーマ 3 件の計 4 件を用意した．誤りスキーマは，(i) 型・ラベル・プロパティの欠損（**構造の欠損**），(ii) エッジ接続誤りや継承関係の逆転（**意味論的破綻**），(iii) 継承構造の過剰展開や型複製（**冗長な構造**），(iv) インスタンスに存在しない要素の導入（**不要な構造**）のいずれか，または複数を含むよう設計した．

評価者は情報系分野の大学生・大学院生・教員からなる 30 名である．各設問について，明確な判断基準を与えず 4 件のスキーマを直感的に良いと思う順に 1 位から 4 位まで順位付けさせた．なお同順位は認めなかった．

ゴールデンランキング 各設問の順位結果は Borda Count により集約し，ゴールデンランキングを構築した．さらに，評価者間の順位分布が一様分布から有意に逸脱しているかをカイ二乗検定により検証した結果，すべての設問で $p < 0.001$ が確認され，統計的に有意な合意が存在することを確認した．

5.5.1 人間のスキーマ評価軸の分析

本実験では明示的な評価基準を与えずに得られた順位結果から，人間がどの欠陥を相対的に重大と捉えるかを分析する．各誤りスキーマには，意味論的破綻，構造の欠損，不要な構造，冗長な構造のいずれか 1 つの代表的欠陥タグを付与した．なおスキーマが誤りを複数含む場合は，意味論的破綻 >構造の欠損 >不要な構造 >冗長な構造 の優先順でタグを付与した．

同一設問内の誤りスキーマ同士の順位関係に基づき，ある評価者が欠陥タグ t を持つスキーマを他の欠陥より重大と判断した確率 $p(t > \textbf{others})$ を算出し，全評価者・全設問について集約した．結果を表 8 および図 11 に示す．

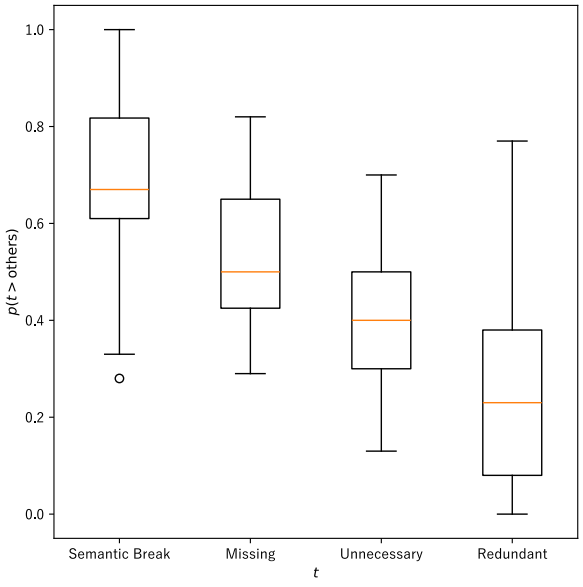


図 11: 欠陥タグごとの $p(t > \textbf{others})$ の分布

表 8: 欠陥タイプごとの相対的重大度の比較

Name	p(意味破綻 > その他)	p(欠損 > その他)	p(不要 > その他)	p(冗長 > その他)
評価者平均	0.68	0.54	0.42	0.25
ゴールデンランキング	0.83	0.59	0.40	0.00
提案手法	0.83	0.06	0.30	0.77

表 8 より，人間のスキーマ評価では，ゴールデンランキングおよび評価者平均の双方において意味論的破綻が最も重大と評価された．次いで構造の欠損，不要な構造，冗長な構造の順に重大と評価される一貫した傾向が確認された．特に意味論的破綻は他の欠陥タイプと比較して著しく高い割合で重大と判断されており，スキーマの正当性を評価する上で最も支配的な評価軸であることが示唆される．

一方で図 11 に示すように，いずれの欠陥タイプについても評価者間には一定のばらつきが存在することが分かる．その中でも冗長な構造はゴールデンランキングでは重大と判断されない一方で，評価者間の判断の分散が最も大きい欠陥タイプであることが確認された．この結果は，特定の欠陥タイプに限らずスキーマ品質の評価という行為そのものが評価者ごとに異なる判断基準を含み，一定の属人性を内在していることを示唆している．

このような属人性はスキーマの比較や選択を行う際の再現性を低下させる要因となる．したがって，人間の主観的判断に依存せず一貫した基準でスキーマを比較可能とする定量的かつ自動的な評価指標を提供することの重要性が，本ユーザスタディの結果からも裏付けられたと言える．

5.5.2 提案手法とゴールデンランキングの比較

本実験では提案手法によるスキーマ順位とゴールデンランキングとの一致度を評価する．提案手法による順位は，各スキーマ案に対して算出したノードおよびエッジの C2 スコアの平均に基づき決定した．

まず Kendall の順位相関係数 τ を用いて各設問における提案手法の順位とゴールデンランキングとの一致度を測定した．その結果，12 設問中 11 設問で $\tau > 0$，1 設問で $\tau = 0$ が得られ，平均 $\tau = 0.523$（95% 信頼区間は $[0.39, 0.67]$）であった．また，最上位スキーマの一致率は 100% であった．これらの結果は，提案手法が人間の集合的判断と同方向の順位付けを一貫して再現していることを示している．

また表 8 を参照して欠陥タイプ別に評価傾向を比較すると，提案手法と人間評価はいずれも意味論的破綻を最も重大な欠陥として評価する点で一致していた．一方で，構造の欠損と冗長な構造に対する評価の重み付けには明確な差が見られた．具体的には，人間評価では構造の欠損に対してより大きなペナルティが与えられる傾向があるのに対し，提案手法では冗長な構造が相対的に重大と評価される傾向が確認された．

この評価傾向の差は，冗長性判定の強弱を決定するハイパパラメータ γ の値を小さくしたり，C2 スコアを算出する際に網羅性と簡潔性の重み付けを調整することである程度制御可能である．一方で，前節で示した通り人間のスキーマ評価自体にも評価者間のばらつきが存在する．したがって，各欠陥をどの程度重く扱うべきかという点については今後さらなる検討が必要な課題である．

6 まとめと今後の課題

本研究では，スキーマレスに運用される PG に対して複数のスキーマ候補を客観的かつ定量的に比較するためのスキーマ品質評価手法を提案した．具体的には，インスタンスに観測される構造をどの程度捉えているかを測る**網羅性**と不要な冗長性や過剰な細分化を抑えた記述であるかを測る**簡潔性**を定義し，それらを統合した指標として **C2 スコア**を導入した．

また評価実験では提案指標が大規模な PG インスタンスに対しても実用的な計算時間で算出可能であることを確認した．加えて，提案手法がスキーマに含まれる欠損や冗長といった誤りの種類および程度に応じて評価値を直感的に妥当な形で変化させることや，既存のスキーマ抽出手法によって生成されたスキーマ間の構造的差異を説明可能な形で評価値に反映できることを示した．

さらに，ユーザスタディを通じて人間のスキーマ評価における主要な評価軸とその相対的重要度を明らかにし，提案指標による評価結果が人間による集合的なスキーマ品質判断と整合する傾向を持つことを確認した．一方で，人間のスキーマ評価には欠陥タイプによらず評価者間のばらつきが存在し，スキーマ評価という行為自体が一定の属人性を内在することも明らかとなった．この結果は，スキーマの比較や選択を人間の主観的判断のみに依存することの限界を示すものであり，再現可能で一貫した基準に基づく自動的なスキーマ評価手法の必要性を裏付けるものである．

今後の課題として，本研究では評価対象から除外したプロパティの値域や型，エッジの多重度などを評価指標に組み込むことが挙げられる．また，データ特性や利用目的に応じて自動的に各種ハイパーパラメータを調整する手法の検討も重要な課題である．これらの課題に取り組むことで，多様な運用要件やデータ特性を持つ PG に対しても，一貫した基準に基づくスキーマ品質評価を可能とする汎用的な評価基盤を確立できると考える．

謝　　辞

本研究の一部は JST 創発的研究支援事業 JPMJFR232P，ならびに，JSPS 科研費 若手研究 22K17894 の支援を受けたものである．

文　献

[1] Xue Lei. Property graph schema extraction. Master's thesis, Eindhoven University of Technology, 2021.

[2] Hanâ Lbath, Angela Bonifati, and Russ Harmer. Schema inference for property graphs. In Proceedings of the 24th International Conference on Extending Database Technology (EDBT 2021), pp. 499–504, 2021.

[3] Angela Bonifati, Stefania Dumbrava, and Nicolas Mir. Hierarchical clustering for property graph schema discovery. In 25th Proceedings of the International Conference on Extending Database Technology (EDBT 2022), pp. 449–453, 2022.

[4] Angela Bonifati, Stefania Dumbrava, Emile Martinez, Fatemeh Ghasemi, Malo Jaffré, Pacôme Luton, and Thomas Pickles. Discopg: property graph schema discovery and exploration. Proc. VLDB Endow., Vol. 15, No. 12, p. 3654 – 3657, August 2022.

[5] Svetlozar Nestorov, Serge Abiteboul, and Rajeev Motwani. Inferring structure in semistructured data. ACM SIGMOD Record, Vol. 26, No. 4, pp. 39–43, 1997.

[6] Tushar Sharma, Marios Fragkoulis, Stamatia Rizou, Magiel Bruntink, and Diomidis Spinellis. Smelly relations: measuring and understanding database schema quality. In Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice, ICSE-SEIP '18, p. 55 – 64, New York, NY, USA, 2018. Association for Computing Machinery.

[7] Roy Goldman and Jennifer Widom. Dataguides: Enabling query formulation and optimization in semistructured databases. In Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB 1997), pp. 436–445. Morgan Kaufmann, 1997.

[8] Geert Jan Bex, Wim Martens, Frank Neven, and Thomas Schwentick. Expressiveness of xsds: From practice to theory, there and back again. In Proceedings of the 14th International Conference on World Wide Web (WWW 2005), pp. 712–721, 2005.

[9] Felipe Pezoa, L. Reutter, Juan Fernando Suárez, Martín Ugarte, and Domagoj Vrgoc. Foundations of json schema. In Proceedings of the 25th International Conference on World Wide Web (WWW '16), pp. 263–273, Geneva, Switzerland, 2016. International World Wide Web Conferences Steering Committee.

[10] Nicholas Gibbins and Nigel Shadbolt. Resource description framework (RDF). In Encyclopedia of Library and Information Sciences, pp. 1–30. Taylor & Francis, 2009.

[11] Grigoris Antoniou and Frank van Harmelen. Web ontology language: OWL. In Handbook on Ontologies, pp. 91–110. Springer, 2009.

[12] Renzo Angles. The property graph database model. In Proceedings of the 12th Alberto Mendelzon International Workshop on Foundations of Data Management (AMW 2018), 2018.

[13] Angela Bonifati, Peter Furniss, Alastair Green, Russ Harmer, Eugenia Oshurko, and Hannes Voigt. Schema validation and evolution for graph databases. In Proceedings of the 38th International Conference on Conceptual Modeling (ER 2019), pp. 448–456, 2019.

[14] Nimo Beeren and George Fletcher. A formal design framework for practical property graph schema languages. In Proceedings of the 26th International Conference on Extending Database Technology (EDBT 2023), pp. 478–484, 2023.

[15] Renzo Angles, Angela Bonifati, Stefania Dumbrava, George Fletcher, Alastair Green, Jan Hidders, Bei Li, Leonid Libkin, Victor Marsault, Wim Martens, Filip Murlak, Stefan Plantikow, Ognjen Savković, Michael Schmidt, Juan F. Sequeda, Sławek Staworko, Dominik Tomaszuk, Hannes Voigt, Domagoj Vrgoč, Mingxi Wu, and Dušan Živković. PG-Schema: Schemas for property graphs. Proceedings of the ACM on Management of Data, Vol. 1, No. 2, pp. 1–25, 2023.

[16] Renzo Angles, Angela Bonifati, Stefania Dumbrava, George Fletcher, Keith W. Hare, Jan Hidders, Victor E. Lee, Bei Li, Leonid Libkin, Wim Martens, Filip Murlak, Josh Perryman, Ognjen Savković, Michael Schmidt, Juan F. Sequeda, Sławek Staworko, and Dominik Tomaszuk. PG-Keys: Keys for property graphs. In Proceedings of the 2021 ACM SIGMOD International Conference on Management of Data (SIGMOD 2021), pp. 2423–2436, 2021.

[17] Philipp Skavantzos and Sebastian Link. Normalizing property graphs. Proceedings of the VLDB Endowment, Vol. 16, No. 11, pp. 3031–3043, 2023.

[18] Philipp Skavantzos and Sebastian Link. Third and boyce–codd normal form for property graphs. The VLDB Journal, Vol. 34, No. 2, 2025.

[19] Sam Skartsis, Alexey Volkov, and Olaf Hartig. Transforming RDF graphs to property graphs using standardized schemas. In Proceedings of the 2025 ACM SIGMOD International Conference on Management of Data (SIGMOD 2025) – Companion Volume, 2025.

[20] Chandan Sharma, Pierre Genevès, Nils Gesbert, and Nabil Layaïda. Schema-based query optimisation for graph databases. Proceedings of the ACM on Management of Data, Vol. 3, No. 1, pp. 1–29, 2025.

[21] Minos N. Garofalakis, Aristides Gionis, Rajeev Rastogi, S. Seshadri, and Kyuseok Shim. Xtract: A system for extracting document type descriptors from XML documents. In Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data (SIGMOD 2000), pp. 165–176. Association for Computing Machinery, 2000.

[22] Geert Jan Bex, Frank Neven, Thomas Schwentick, and Karl Tuyls. Inference of concise DTDs from XML data. In Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB 2006), pp. 115–126. VLDB Endowment, 2006.

[23] Mohamed-Amine Baazizi, Dario Colazzo, Giorgio Ghelli, and Carlo Sartiani. Parametric schema inference for massive json datasets. The VLDB Journal, Vol. 28, No. 4, pp. 497–521, 2019.

[24] Redouane Bouhamoum, Kenza Kellou-Menouer, Stéphane Lopes, and Zoubida Kedad. Scaling up schema discovery for rdf datasets. In Proceedings of the 2018 IEEE 34th International Conference on Data Engineering Workshops (ICDEW), ICDEW '18, pp. 84–89, Piscataway, NJ, USA, 2018. IEEE.

[25] Artem Lutov, Soheil Roshankish, Mourad Khayati, and Philippe Cudré-Mauroux. Statix – statistical type inference on linked data. In Proceedings of the 2018 IEEE International Conference on Big Data (Big Data '18), Big Data '18, pp. 2253–2262, Piscataway, NJ, USA, 2018. IEEE.

[26] Kenza Kellou-Menouer and Zoubida Kedad. Schema discovery in rdf data sources. In Proceedings of the 34th International Conference on Conceptual Modeling (ER 2015), Vol. 9381 of Lecture Notes in Computer Science, pp. 481–495, 2015.

[27] Aldo Gangemi, Andrea G. Nuzzolese, Valentina Presutti, Francesco Draicchio, Alberto Musetti, and Paolo Ciancarini. Automatic typing of DBpedia entities. In Proceedings of the 11th International Semantic Web Conference (ISWC 2012), Vol. 7649 of Lecture Notes in Computer Science, pp. 65–81, 2012.

[28] Information technology — Database languages — GQL, 2024.

[29] Xiaohui Victor Li and Francesco Sanna Passino. Findkg: Dynamic knowledge graphs with large language models for detecting global trends in financial markets. In Proceedings of the 5th ACM International Conference on AI in Finance, ICAIF '24, p. 573 – 581. ACM, November 2024.
[30] Orri Erling, Alex Averbuch, Josep Larriba-Pey, Hassan Chafi, Andrey Gubichev, Arnau Prat, Minh-Duc Pham, and Peter Boncz. The ldbc social network benchmark: Interactive workload. In Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, SIGMOD '15, p. 619 – 630, New York, NY, USA, 2015. Association for Computing Machinery.
[31] Takemura et al. A connectome of a learning and memory center in the adult *Drosophila* brain. eLife, Vol. 6, p. e26975, jul 2017.
[32] Neo4j. Network management graph example dataset. `https://github.com/neo4j-graph-examples/network-management`, 2025. Accessed 2025-01-04.
[33] Microsoft. Northwind sample database. `https://github.com/microsoft/sql-server-samples/tree/master/samples/databases/northwind-pubs`, 2025. Accessed 2025-01-04.
[34] Hamed Zamani, Markus Schedl, Paul Lamere, and Ching-Wei Chen. An analysis of approaches taken in the acm recsys challenge 2018 for automatic music playlist continuation. ACM Trans. Intell. Syst. Technol., Vol. 10, No. 5, September 2019.
[35] Transaction Processing Performance Council. Tpc benchmark™ h (decision support) standard specification. `https://www.tpc.org/tpc_documents_current_versions/pdf/tpc-h_v2.17.3.pdf`, 2017. Accessed 2025-01-04.
[36] Neo4j. Twitter v2 graph example dataset. `https://github.com/neo4j-graph-examples/twitter-v2`, 2025. Accessed 2025-01-04.
[37] George A. Miller. Wordnet: a lexical database for english. Commun. ACM, Vol. 38, No. 11, p. 39 – 41, November 1995.

付　　録

1 アルゴリズム

1.1 インスタンス由来スキーマの抽出アルゴリズム

アルゴリズム 1 は，プロパティグラフのインスタンス $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \lambda, \kappa, \sigma)$ から，インスタンス由来スキーマ $\mathcal{S}^{\mathcal{G}} = \mathbf{Abs}(\mathcal{G})$ を構成する．このアルゴリズムでは，構造が類似するインスタンスオブジェクトをまとめるために，ノードとエッジをそれぞれ互いに素な部分集合へ分割し，各部分集合から 1 つのノード型またはエッジ型を生成する．

(1) ノードの型の生成 アルゴリズム 1 の 4-16 行目がノード型の生成に対応する．ノードは，ラベル集合が完全に一致するもの同一のグループとしてまとめ，各グループからノード型を 1 つ生成する．

そのために，まずラベル集合 L を持つノードの集合 $\mathcal{V}_L$ を作成する（5 行目）．次に各グループ $\mathcal{V}_L$ ごとに新しいノード型 v_T を作成し，ラベル集合 L をノード型のラベル集合として設定する（6-8 行目）．さらに，グループ内のノードが持つプロパティの和集合をノード型のプロパティ集合として設定する（9 行目）．また，プロパティ k の必須性は，グループ内のすべてのノードが k を持つかどうかで決定し，その結果をノード型の必須性制約として設定する（10-14 行目）．継承関係は導入せず（15 行目），次のエッジ型の生成のためにラベル集合 L とノード型 v_T の対応を記憶しておく（16 行目）．

Algorithm 1: インスタンス由来スキーマの生成アルゴリズム

Input: $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \lambda, \kappa, \sigma)$
Output: $\mathcal{S}^{\mathcal{G}} = (\mathcal{V}_T^{\mathcal{G}}, \mathcal{E}_T^{\mathcal{G}}, \lambda^{\mathcal{G}}, \kappa^{\mathcal{G}}, \sigma^{\mathcal{G}}, \mu^{\mathcal{G}}, \rho^{\mathcal{G}})$

1: **function** Abs($\mathcal{G}$)
2: $\quad \mathcal{V}_T^{\mathcal{G}} \leftarrow \varnothing$;
3: $\quad \mathcal{E}_T^{\mathcal{G}} \leftarrow \varnothing$;
$\quad$ // ノード型の生成
4: $\quad$ **foreach** $L \in \{\lambda(v) \mid v \in \mathcal{V}\}$ **do**
5: $\quad\quad \mathcal{V}_L \leftarrow \{v \in \mathcal{V} \mid \lambda(v) = L\}$;
6: $\quad\quad v_T \leftarrow$ `createNewNodeType`();
7: $\quad\quad \mathcal{V}_T^{\mathcal{G}} \leftarrow \mathcal{V}_T^{\mathcal{G}} \cup \{v_T\}$;
8: $\quad\quad \lambda^{\mathcal{G}}(v_T) \leftarrow L$;
9: $\quad\quad \kappa^{\mathcal{G}}(v_T) \leftarrow \bigcup_{v \in O_L} \kappa(v)$;
10: $\quad\quad P_{\mathbf{req}} \leftarrow \bigcap_{v \in O_L} \kappa(v)$;
11: $\quad\quad$ **foreach** $k \in \kappa^{\mathcal{G}}(v_T)$ **do**
12: $\quad\quad\quad \mu^{\mathcal{G}}(v_T, k) \leftarrow$ `True`;
13: $\quad\quad\quad$ **if** $k \notin P_{\mathbf{req}}$ **then**
14: $\quad\quad\quad\quad \mu^{\mathcal{G}}(v_T, k) \leftarrow$ `False`;
15: $\quad\quad \rho^{\mathcal{G}}(v_T) \leftarrow \varnothing$;
16: $\quad\quad M[L] \leftarrow v_T$; $\quad$ // ラベル集合 $\mapsto$ ノード型の対応表
$\quad$ // エッジ型の生成
$\quad$ // $\tau(e) = (\lambda(e), \lambda(\mathbf{src}(e)), \lambda(\mathbf{dst}(e)))$
17: $\quad$ **foreach** tpl $\in \{\, \tau(e) \mid e \in \mathcal{E} \,\}$ **do**
18: $\quad\quad \mathcal{E}_{\text{tpl}} \leftarrow \{\, e \in \mathcal{E} \mid \tau(e) = \text{tpl} \,\}$;
19: $\quad\quad e_T \leftarrow$ `createNewEdgeType`();
20: $\quad\quad \mathcal{E}_T^{\mathcal{G}} \leftarrow \mathcal{E}_T^{\mathcal{G}} \cup \{e_T\}$;
21: $\quad\quad (L_e, L_{\mathbf{src}}, L_{\mathbf{dst}}) \leftarrow$ tpl;
22: $\quad\quad \lambda^{\mathcal{G}}(e_T) \leftarrow L_e$;
23: $\quad\quad \kappa^{\mathcal{G}}(e_T) \leftarrow \bigcup_{e \in \mathcal{E}_{\text{tpl}}} \kappa(e)$;
24: $\quad\quad P_{\mathbf{req}} \leftarrow \bigcap_{e \in \mathcal{E}_{\text{tpl}}} \kappa(e)$;
25: $\quad\quad$ **foreach** $k \in \kappa^{\mathcal{G}}(e_T)$ **do**
26: $\quad\quad\quad$ **if** $k \in P_{\mathbf{req}}$ **then**
27: $\quad\quad\quad\quad \mu^{\mathcal{G}}(e_T, k) \leftarrow$ `True`;
28: $\quad\quad\quad$ **else**
29: $\quad\quad\quad\quad \mu^{\mathcal{G}}(e_T, k) \leftarrow$ `False`;
30: $\quad\quad \sigma^{\mathcal{G}}(e_T) \leftarrow (M[L_{\mathbf{src}}], M[L_{\mathbf{dst}}])$;
31: $\quad\quad \rho^{\mathcal{G}}(e_T) \leftarrow \varnothing$;
32: $\quad$ **return** $(\mathcal{V}_T^{\mathcal{G}}, \mathcal{E}_T^{\mathcal{G}}, \lambda^{\mathcal{G}}, \kappa^{\mathcal{G}}, \sigma^{\mathcal{G}}, \mu^{\mathcal{G}}, \rho^{\mathcal{G}})$;

(2) エッジの型の生成 アルゴリズム 1 の 17-31 行目がエッジ型の生成に対応する．エッジは，$(\lambda(e), \lambda(\mathbf{src}(e)), \lambda(\mathbf{dst}(e)))$ の組が完全に一致するものを同一のグループとしてまとめ（18 行目），ノード型と同様に各グループからエッジ型を 1 つ生成する（19-27 行目）．最後に，エッジ型の端点ノード型を設定する（28 行目）．

1.2 継承関係の展開アルゴリズム

継承関係の展開アルゴリズムをアルゴリズム 2 に示す．このアルゴリズムでは，まず元のスキーマの属性・制約を展開後スキーマにコピーする（2-10 行目）．次に，各オブジェクト型に対して親・先祖オブジェクト型から属性・制約を継承する（11-17 行目）．この操作は，図 2 に示すスキーマのノード型 v_1^* のラベ

ル User や必須プロパティ id を，図 5 に示す展開後スキーマの v_2^+ に継承させる操作に対応する．続いて，各エッジ型に対して端点ノード型の子孫ノード型を考慮した新エッジ型を生成する（18-31 行目）．この操作は，図 2 に示すスキーマのエッジ型 e_1^* を図 5 に示す展開後スキーマの e_4^+ に展開する操作に対応する．最後に，継承関係を空にする（32-34 行目）．

Algorithm 2: 継承関係の展開アルゴリズム

Input: $\mathcal{S}^* = (\mathcal{V}_T^*, \mathcal{E}_T^*, \lambda, \kappa, \sigma, \mu, \rho)$：評価対象スキーマ
Output: $\mathcal{S}^+ = (\mathcal{V}_T^+, \mathcal{E}_T^+, \lambda^+, \kappa^+, \sigma^+, \mu^+, \rho^+)$：展開後スキーマ

1: **function** Flat($\mathcal{S}^*$)
    // 元のスキーマをコピー
2: $\mathcal{V}_T^+ \leftarrow \mathcal{V}_T^*$;
3: $\mathcal{E}_T^+ \leftarrow \mathcal{E}_T^*$;
4: **foreach** $o \in (\mathcal{V}_T^* \cup \mathcal{E}_T^*)$ **do**
5: $\lambda^+(o) \leftarrow \lambda(o)$;
6: $\kappa^+(o) \leftarrow \kappa(o)$;
7: **foreach** $k \in \kappa(o)$ **do**
8: $\mu^+(o,k) \leftarrow \mu(o,k)$;
9: **if** $o \in \mathcal{E}_T^*$ **then**
10: $\sigma^+(o) \leftarrow \sigma(o)$;
    // 親・先祖からの属性・制約を継承する
11: **foreach** $o \in (\mathcal{V}_T^* \cup \mathcal{E}_T^*)$ **do**
12: **foreach** $a \in \mathtt{Anc}(o)$ **do**
13: $\lambda^+(o) \leftarrow \lambda^+(o) \cup \lambda(a)$;
14: $\kappa^+(o) \leftarrow \kappa^+(o) \cup \kappa(a)$;
15: **foreach** $k \in \kappa(a)$ **do**
        // 子オブジェクト型が既に k の必須性制約を持つ場合は上書きしない
16: **if** $\mu^+(o,k)$ is undefined **then**
17: $\mu^+(o,k) \leftarrow \mu(a,k)$;
    // エッジ型の展開
18: **foreach** $e \in \mathcal{E}_T^*$ **do**
19: $(v_{\mathbf{src}}, v_{\mathbf{dst}}) \leftarrow \sigma(e)$;
20: $D'_{\mathbf{src}} \leftarrow \{v_{\mathbf{src}}\} \cup \mathtt{Desc}(v_{\mathbf{src}})$;
21: $D'_{\mathbf{dst}} \leftarrow \{v_{\mathbf{dst}}\} \cup \mathtt{Desc}(v_{\mathbf{dst}})$;
22: **foreach** $d_s \in D'_{\mathbf{src}}$ **do**
23: **foreach** $d_t \in D'_{\mathbf{dst}}$ **do**
24: **if** $(d_s, d_t) \neq (v_{\mathbf{src}}, v_{\mathbf{dst}})$ **then**
            // 端点ノード型のみが異なる新エッジ型を作成
25: $e' \leftarrow \mathtt{createNewEdgeType}()$;
26: $\mathcal{E}_T^+ \leftarrow \mathcal{E}_T^+ \cup \{e'\}$;
27: $\lambda^+(e') \leftarrow \lambda^+(e)$;
28: $\kappa^+(e') \leftarrow \kappa^+(e)$;
29: **foreach** $k \in \kappa^+(e)$ **do**
30: $\mu^+(e',k) \leftarrow \mu^+(e,k)$;
31: $\sigma^+(e') \leftarrow (d_s, d_t)$;
    // 継承関係を空にする
32: **foreach** $o \in (\mathcal{V}_T^+ \cup \mathcal{E}_T^+)$ **do**
33: $\rho^+(o) \leftarrow \varnothing$;
34: **return** $(\mathcal{V}_T^+, \mathcal{E}_T^+, \lambda^+, \kappa^+, \sigma^+, \mu^+, \rho^+)$;

// 補助関数
// $\mathtt{Anc}(o)$：ρ に基づき o から親方向に到達可能な全ての型を返す
// $\mathtt{Desc}(o)$：ρ に基づき o から子方向に到達可能な全ての型を返す

1.3 網羅性評価アルゴリズム

網羅性の評価アルゴリズムをアルゴリズム 3 に示す．なお，アルゴリズム 3 ではスキーマのノード網羅性 $\mathbf{Cvg}_V(\mathcal{G}, \mathcal{S}^*)$ を計算する場合を示しているが，エッジも同様の手順で計算できる．

Algorithm 3: ノード網羅性の評価アルゴリズム

Input: $\mathcal{G}$：インスタンス，$\mathcal{S}^*$：評価対象スキーマ
Output: $\mathbf{Cvg}_V(\mathcal{G}, \mathcal{S}^*)$

1: **function** CalcCvgV($\mathcal{G}, \mathcal{S}^*$)
2: $\mathcal{S}^{\mathcal{G}} \leftarrow \mathtt{Abs}(\mathcal{G})$; // インスタンス由来スキーマの抽出
3: $\mathcal{S}^+ \leftarrow \mathtt{Flat}(\mathcal{S}^*)$; // 継承関係の展開
4: **foreach** $v^{\mathcal{G}} \in V(\mathcal{S}^{\mathcal{G}})$ **do**
5: $\mathrm{max\_sim}[v^{\mathcal{G}}] \leftarrow \max_{v^+ \in V(\mathcal{S}^+)} \mathtt{sim}_V(v^{\mathcal{G}}, v^+)$;
6: **return** avg $([\,\mathrm{max\_sim}[v^{\mathcal{G}}] \mid v^{\mathcal{G}} \in V(\mathcal{S}^{\mathcal{G}})\,])$;

1.4 簡潔性評価アルゴリズム

簡潔性の評価アルゴリズムをアルゴリズム 4 に示す．なお，アルゴリズム 4 ではスキーマのノード簡潔性 $\mathbf{Con}_V(\mathcal{G}, \mathcal{S}^*)$ を計算する場合を示しているが，エッジの場合も同様の手順で計算できる．

Algorithm 4: ノード簡潔性の評価アルゴリズム

Input: $\mathcal{G}$：インスタンス，$\mathcal{S}^*$：評価対象スキーマ
Output: $\mathbf{Con}_V(\mathcal{G}, \mathcal{S}^*)$

1: **function** CalcConV($\mathcal{G}, \mathcal{S}^*$)
    // 元のスキーマの網羅性を計算
2: $\mathbf{cvg}_V \leftarrow \mathtt{CalcCvgV}(\mathcal{G}, \mathcal{S}^*)$;
3: $\mathbf{cvg}_E \leftarrow \mathtt{CalcCvgE}(\mathcal{G}, \mathcal{S}^*)$;
    // 寄与下限を計算
4: $\mathcal{S}^+ \leftarrow \mathtt{Flat}(\mathcal{S}^*)$;
5: $\epsilon_{\mathcal{V}} \leftarrow \dfrac{c_{\mathcal{V}}}{|\,\mathcal{V}_T^+\,|} \times \gamma$;
6: $\epsilon_{\mathcal{E}} \leftarrow \dfrac{c_{\mathcal{E}}}{|\,\mathcal{E}_T^+\,|} \times \gamma$;
7: $R \leftarrow 0$; // 冗長オブジェクト数
8: **foreach** $v^* \in \mathcal{V}_T^*$ **do**
        // ノード型 v^* を除去したスキーマで網羅性を再計算
9: $\mathbf{cvg}'_V \leftarrow \mathtt{CalcCvgV}(\mathcal{G}, \mathcal{S}^* \setminus \{v^*\})$;
10: $\mathbf{cvg}'_E \leftarrow \mathtt{CalcCvgE}(\mathcal{G}, \mathcal{S}^* \setminus \{v^*\})$;
        // 網羅性の低下量を計算
11: $\Delta_V \leftarrow \mathbf{cvg}_V - \mathbf{cvg}'_V$;
12: $\Delta_E \leftarrow \mathbf{cvg}_E - \mathbf{cvg}'_E$;
        // 網羅性の低下量が寄与下限以下なら冗長とみなす
13: **if** $\Delta_{\mathcal{V}} < \epsilon_{\mathcal{V}} \ \wedge \ \Delta_{\mathcal{E}} < \epsilon_{\mathcal{E}}$ **then**
14: $R \leftarrow R + 1$;
15: **return** $1 - \dfrac{R}{|\,\mathcal{V}_T^*\,|}$;

2 計算量の解析

本節では，提案手法の計算量について整理する．以降，インスタンスを $\mathcal{G}$，評価対象スキーマを $\mathcal{S}^*$，展開後スキーマを $\mathcal{S}^+ = \mathbf{Flat}(\mathcal{S}^*)$，インスタンス由来スキーマを $\mathcal{S}^{\mathcal{G}} = \mathbf{Abs}(\mathcal{G})$ とする．

まず，前処理である $\mathbf{Abs}(\mathcal{G})$ は，インスタンス中の各ノードおよびエッジを一度ずつ走査し，ラベル集合およびプロパティ集合に基づいて分類を行うため，計算量は $O(|V(\mathcal{G})| + |E(\mathcal{G})|)$ である．同様に，$\mathbf{Flat}(\mathcal{S}^*)$ は，スキーマの継承関係を推移的に展開する操作であり，スキーマサイズに対して多項式時間で実行可能である．

次に，網羅性（Coverage）の計算では，$\mathcal{S}^{\mathcal{G}}$ に含まれる各ノード型（エッジ型）に対して，$\mathcal{S}^+$ に含まれるすべてのノード型（エッジ型）との類似度を計算し，最大値を求める．したがって，ノード網羅性およびエッジ網羅性の計算量は，それぞれ $O(|V(\mathcal{S}^{\mathcal{G}})||V(\mathcal{S}^+)|)$，$O(|E(\mathcal{S}^{\mathcal{G}})||E(\mathcal{S}^+)|)$ である．

簡潔性（Concision）の計算では，スキーマ中の各オブジェクト型 $o \in \mathcal{S}^*$ を 1 つずつ除去し，その都度，網羅性の低下量を評価する．素朴に実装した場合，各オブジェクト除去ごとに網羅性を再計算する必要があるため，計算量は $O(|V(\mathcal{S}^*)||V(\mathcal{S}^{\mathcal{G}})||V(\mathcal{S}^+)|)$（エッジについても同様）となる．

しかし，本研究では，$\mathcal{S}^{\mathcal{G}}$ と $\mathcal{S}^+$ の間で計算される類似度および最大類似度の結果をメモ化し，オブジェクト除去のたびに再利用することで，重複する計算を省いている．この最適化により，簡潔性計算において新たに必要となる計算は，除去対象に対応する一部の寄与の更新に限られ，網羅性全体を再計算する必要はない．その結果，簡潔性計算を含む全体の計算量は，実効的に $O(|V(\mathcal{S}^{\mathcal{G}})||V(\mathcal{S}^+)| + |E(\mathcal{S}^{\mathcal{G}})||E(\mathcal{S}^+)|)$ に抑えられる．

以上をまとめると，提案手法の計算量は

$$O(|V(\mathcal{G})| + |E(\mathcal{G})| + |V(\mathcal{S}^{\mathcal{G}})||V(\mathcal{S}^+)| + |E(\mathcal{S}^{\mathcal{G}})||E(\mathcal{S}^+)|)$$

である．一般に，スキーマサイズは対応するインスタンスサイズに比べて十分に小さいことが多いため，実運用においては $|V(\mathcal{G})|$ および $|E(\mathcal{G})|$ による走査コストが支配的になると考えられる．実際に，本研究のスケーラビリティ評価実験においても，実行時間はインスタンスサイズの増加に対してほぼ線形に増加しており，本解析と整合した挙動が確認された．

3 実験設定

3.1 提案手法のハイパーパラメータ

本研究で用いたハイパパラメータの値は $\alpha = 0.5$，$\beta = 0.5$，$\gamma = 0.15$. である．

α はノード型およびエッジ型の類似度計算において，ラベル一致とプロパティ一致の相対的な重要度を調整するパラメータである．$\alpha = 0.5$ とすることで，ラベルとプロパティの双方を同程度に考慮する中立的な設定とした．

β はエッジ型の類似度計算において，エッジ本体の属性と，接続先ノード型の一致度の寄与を調整するパラメータである．本研究では，エッジの意味と構造のいずれも同様に重要であると考え，$\beta = 0.5$ に固定した．

γ は簡潔性の計算において，冗長オブジェクトを判定する際の厳しさを制御するパラメータである．$\gamma = 0.15$ は，LDBC，Northwind，Spotify，TPC-H の各データセットにおける正解スキーマを冗長と判定しない上限値として経験的に決定した．これよりも大きな値を用いると，LDBC のように比較的スキーマサイズの大きい正解スキーマにおいて，明らかにインスタンスのデータ構造を説明するために必要と判断されるオブジェクト型が冗長と判定される事例が確認された．本研究では実験条件を単純化するため，冗長判定に対して比較的緩やかな値である $\gamma = 0.15$ を全スキーマに対して一律に用いた．

インスタンスや人間の評価特性に応じた適切なハイパーパラメータの自動調整手法の検討は今後の課題である．

3.2 スキーマ抽出手法のハイパーパラメータ

本研究で用いたスキーマ抽出手法のハイパーパラメータ設定を表 A·1 に示す．

表 A·1: 各スキーマ抽出手法のハイパーパラメータ設定

手法	ハイパーパラメータ	値
類似度ベース クラスタリング手法 [1]	l	1.0
	p	0.0
	t	0.0
	k	0
	θ	0.5
ルールベース手法 [2]	なし	–
階層型 クラスタリング手法 [3]	sampling rate	0.7
	top_k_props	1
	num_clusters	2

機械学習によるサブグラフマッチングアルゴリズム選択

倉谷　元琉† Skitsas Konstantinos†† Panagiotis Karras††† 天方　大地† 佐々木 勇和†

† 大阪大学大学院情報科学研究科　〒 565–0871 大阪府吹田市山田丘 1-5
†† Aarhus University　Nordre Ringgade 1, 8000 Aarhus C, Denmark
††† Copenhagen University　Nørregade 10, 1165 København K, Denmark
E-mail: †{kuraya.genryu,amagata.daichi,sasaki}@ist.osaka-u.ac.jp, ††skitsas@cs.au.dk,
†††piekarras@gmail.com

あらまし　サブグラフマッチングは, データグラフからクエリグラフと呼ばれる特定の構造を有するグラフと同じ構造を有する部分グラフ (埋込み) を全て列挙する, グラフ解析において基本的な問題である. より効率的な方法を目指し, 今までに数多くのアルゴリズムが提案されてきた. しかし, クエリグラフやデータグラフによって, 最も性能が良いアルゴリズムは異なる. 先行研究では, クエリグラフやデータグラフの特徴に基づいたルールベースのアルゴリズム選択手法を示しているが, ルール作成時に想定されていないグラフに対して性能が悪くなってしまう. 本研究では, クエリグラフとデータグラフに基づき, 最適なサブグラフマッチングアルゴリズムを選択する機械学習に基づいた手法を提案する. 実験では, サブグラフマッチング, サブグラフカバレッジにおいて, 提案手法がベースラインよりも 36.0% 多くの単位時間あたりの埋込み数, 46.3% 多くの単位時間あたりのカバレッジを達成したことを示す.

キーワード　サブグラフマッチング, アルゴリズム選択, 機械学習

1 序　　論

サブグラフマッチングとは, 大規模なデータグラフ内にクエリグラフと同じ構造を有する部分グラフ (埋込み) を列挙する, グラフ解析における基本的な問題である. これは, 不正検出 [14] やソーシャルネットワーク [4] など, 幅広く応用されている. しかし, その計算の難しさから, NP 困難 [7] な問題として知られている. そのため, ここ数十年で多くのアルゴリズムが提案されてきた [2], [9], [20]. しかし, それぞれのアルゴリズムにはそれぞれの長所や短所があり, 全てのデータグラフやクエリグラフに対して最適な 1 つの手法が存在しないことが現状である.

図 1 は, 7 つの実データセットにおいて, 75 種類のアルゴリズムの中で各アルゴリズムが単位時間あたりに発見された埋込み数 (EPS) の観点で最大の値を達成した回数を示すヒートマップである. 図 2 は, 各データセットにおいて 1 から 75 のそれぞれの順位における EPS の平均値を表している. これらの結果から, 最適なアルゴリズムはデータセットに影響を受け, 与えられたクエリグラフとデータグラフに対して不適切なアルゴリズムを選択した場合, 効率が大きく低下することを示している.

この問題を解決するために, 与えられたクエリグラフやデータグラフに対して最適なアルゴリズムを選択する必要がある. 先行研究 [18], [24] では実験で得た各アルゴリズムの性能評価に基づき, クエリグラフやデータグラフの特徴を用いてアルゴリズムを選択する, ルールベースの方法が提案されている. しかし, これらの方法では, 性能評価を行ったデータと異なるデータに対して適切なアルゴリズムの選択が難しい. そこで, 与えられたクエリグラフ、データグラフに対してより強固に最適なアルゴリズムが選択される方法が必要である.

本研究では, 与えられたクエリグラフ, データグラフに対し, 効率的な処理を行うことができるアルゴリズムを機械学習を用いて自動的に選択する, 新しい手法を提案する. 提案手法では, データグラフ, クエリグラフ, アルゴリズム, そしてその性能を含んだ, 経験的データを用いて機械学習モデルを訓練する. この手法の主要素は, 経験的データから得られるアルゴリズムの性能に基づいて, 各アルゴリズムに正のラベル付けを行い訓練データを構築するラベリング戦略と, クエリグラフとデータグラフからアルゴリズム選択に有効な特徴量を抽出する特徴量抽出器である. ラベリング戦略として,3 つの TopX, IncY, Weighted を提案する. TopX はそれぞれのクエリグラフとデータグラフの組ごとに上位 X 件の性能を有するアルゴリズムに, IncY では, 最も性能が良いアルゴリズムに対して, その性能が Y% 以内であるアルゴリズムに等しく正解ラベルを与える. Weighted では, 最良の性能に基づいて正解ラベルが与えられる. 特徴量抽出器では, クエリグラフとデータグラフの両方から, 計 13 個の特徴量が抽出される. ラベリング戦略, 特徴量抽出器により, 与えられたクエリグラフとデータグラフの特徴に対して, どのアルゴリズムを選択するべきか, という問題を分類問題として機械学習モデルを訓練する. また推論時には, 訓練済みのモデルを用いて与えられたクエリグラフ, データグラフの組に対して最も良い性能を持つと期待されるアルゴリズムを選択し, それを用いたサブグラフマッチングを実現する.

評価実験では, 7 つの実データグラフに対して 75 通りのアルゴリズムを用いる. 提案手法がベースラインよりも EPS による評価では 35.96% 高く, MRR による評価では 42.75% 高い性能を達成したことを示す. さらに, モデルの訓練時とは異な

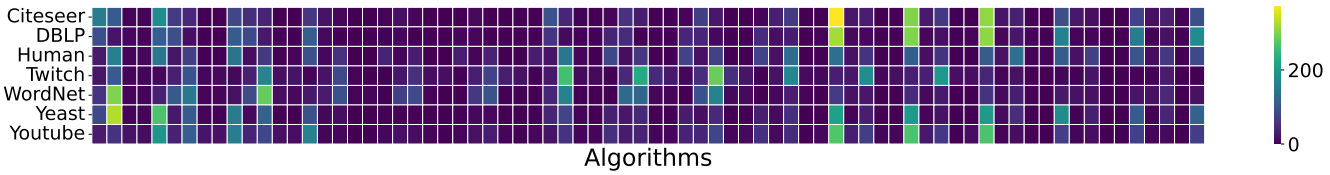


図 1: データセットごとの, 最大 EPS を達成したアルゴリズムの分布を示すヒートマップ.

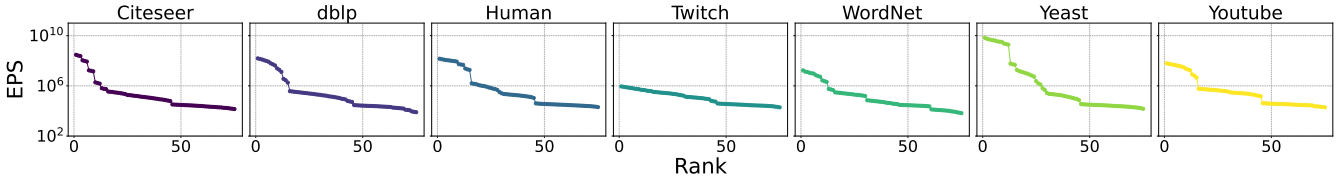


図 2: 順位ごとの平均 EPS; 各クエリに対して i 位のアルゴリズムを選択することで, ランク i 位における EPS を得ている.

るデータやクエリを用いたときの用いたときの, 分布外データに対する性能も示す. 加えて, サブグラフカバレッジ [15], [17] に対しても, 提案手法が適用可能であることも示す.

本研究の貢献は以下のとおりである.

- **新規性のある問題:** 選択すべきサブグラフマッチングアルゴリズムを機械学習を用いて選択する問題に取り組む. この問題に取り組んでいる先行研究は存在しない.
- **新規性のある枠組み:** ラベリング戦略と特徴量抽出器を用いて, 機械学習によってサブグラフマッチングアルゴリズムを選択する包括的な枠組みを提案する.
- **幅広い堅牢な実験:** 7 つの実データを用いた幅広い実験は, 提案手法が幅広い状況でもベースラインより優れていることを示す.

2 事 前 知 識

ここでは, 基本的な定義と関連研究について述べる.

2.1 グラフとサブグラフマッチング

節点ラベル付き無方向グラフを $g = (V(g), E(g), L_g)$ と定義する. ただし, $V(g)$ は節点集合, $E(g)$ は枝集合である. 枝 $e(v, v') \in E(g)$ は, g における $v, v' \in V(g)$ が接続していることを示す. L_g は, 各節点に対し節点ラベル集合 Σ への写像である.

クエリグラフ $q = (V(q), E(q), L_q)$ とデータグラフ $G = (V(G), E(G), L_G)$ に対して, G における q の埋込みとは次の 3 つの条件を満たす写像 $M : V(q) \to V(G)$ である: (i) M は単射である. すなわち, $u \neq u' \in V(q)$ ならば, $M(u) \neq M(u')$ である. (ii) 任意の $u \in V(q)$ について, $L_q(u) = L_G(M(u))$ である. (iii) 任意の枝 $e(u, u') \in E(q)$ について $e(M(u), M(u')) \in E(G)$ である. G 内に q の埋込みが存在するとき, q は G にサブグラフ同型であるという. サブグラフマッチングとは, データグラフ G にクエリグラフ q の全ての埋込みを見つけることである. 多くの場合, データグラフ G はクエリグラフ q に対して非常に大きい構造である.

2.2 サブグラフカバレッジ

サブグラフマッチングと似た問題として, サブグラフカバレッジがある. これは, 制限時間内に発見された埋込みに含まれる重複しない節点数を最大化することを目的とする. また, この重複しない節点数をここではカバレッジと呼ぶ [15], [17].

2.3 関 連 研 究

サブグラフマッチングアルゴリズム. サブグラフマッチングアルゴリズムは, フィルタリング, 順序決定, 埋込みと呼ばれる 3 段階の技術で構成される [18], [24]. フィルタリングは, グラフの構造情報や節点ラベル情報に基づき, クエリグラフの埋込みになる可能性のあるデータグラフ内の節点や枝へと探索空間を絞り込む. 順序決定は, クエリグラフの節点をデータグラフへ効率的に対応付けるための探索順序を決定する. 埋込みは, クエリグラフの全ての埋込みを出力する.

数多くの既存手法がある中, LDF [20], NLF [25] は節点ラベルや次数を用いてフィルタリングを行い, GQL [8], RI [3] は候補節点サイズや接続性に基づいた順序決定を行う. LFTJ [21] は計算量の改善を, RM [19] はリレーショナルデータベースの問題として扱う. また, DPiso [5], VEQ [9], KSS [23] はそれぞれ高度な枝刈りや分解手法で探索空間を削減する. Pilos [16] はラプラシアン行列を用いたスペクトルフィルタリングを行う.

サブグラフマッチングと機械学習. サブグラフマッチングに機械学習を適用させた先行研究 [11]?[13], [22] はいくつか存在する. NeuroMatch [13] では二値分類による解の有無判定を行い, Sub-GMN [11] は単一解の特定を行う. 順序決定に関しては RSM [12] は強化学習を用い, NeuSO [22] は GNN でコスト等を予測して決定する.

これらは全て直接解を求めているが, 本研究のような, 既存のアルゴリズムの中から最適なアルゴリズムを選択するために機械学習を用いている先行研究はない.

サブグラフカバレッジ. サブグラフカバレッジには既存手法に加えて, カバレッジを効率よく探索する手法がある. 例として, MatCo [15] は局所データ構造と枝刈りを, Mix&Match [17] はグラフの広範囲を探索する手法をとっている.

ベンチマーク. ベンチマーク研究 [18], [24] により, 最良のアル

ゴリズムはデータグラフ, クエリグラフによって異なることがわかっている. 先行研究ではアルゴリズムを分解, 再構成して性能向上を図っているが, 本研究では, 再構成したアルゴリズム群の中から, 状況に応じて最適なアルゴリズムを選択する手法を提案する.

3 研究動機と問題定義

図 1 は 7 つのデータセットに対して 75 個のアルゴリズムの中で最も高い EPS を達成したアルゴリズムの分布を表したヒートマップである. ただし, それぞれのデータセットには 2800 個のクエリを用意している (詳細は 5.9 項). この図から, 最も高い EPS を達成するアルゴリズムの傾向は, データセットの種類によって大きくばらつきがあることがわかる.

一方で, 図 2 からはアルゴリズムの順位とその性能には単純な比例関係ではないことが示される. 例えば, Yeast では上位のアルゴリズムと下位のアルゴリズムの間には, 6 桁ほどの EPS の差が見られることに対し, Twitch では上位のアルゴリズムと下位のアルゴリズムの差は大きくない. これらの結果は, 常に最上位のアルゴリズムを選択しなければならないということではないことを示しつつ, 下位のアルゴリズムを選択してしまうと著しくその性能を下げてしまう可能性を示している. そこで我々は, 実験結果から得られる経験的な性能データに基づいてアルゴリズムが選択されることが実用的となり得ると考える. ここで次の定義を行う:

経験的データ. アルゴリズムの集合を $\mathcal{A}$ としたときに, 経験的データとは, $< G, q,$ アルゴリズム, 性能 $>$ の 4 つの組である. ただし, アルゴリズムとは $\mathcal{A}$ に含まれる任意の要素であり, G および q はそれぞれデータグラフ, クエリグラフのことを示す.

本研究では性能の評価指標に EPS [24] やカバレッジ [15], [17] を用いているが, 実行時間など, 他の指標を用いることもできる. 各データグラフ, クエリグラフ, そしてアルゴリズムについて, その性能が既知であることを想定する. そのため, G と q に対して最も性能が良いアルゴリズムを特定できることになる. そこで, 今回の問題を以下に定義する:

サブグラフマッチングアルゴリズム選択. クエリグラフ q, データグラフ G, アルゴリズム集合 $\mathcal{A}$ が与えられた時に, サブグラフマッチングアルゴリズム選択とは q と G に対して $\mathcal{A}$ の中から最も性能が良いアルゴリズムを選択することである.

単純な方法として, 経験的データに基づいて概ね高い性能を達成するアルゴリズムを選択することが考えられる. 例えば, 全てのデータセット, またはそれぞれのデータセットに含まれる全てのクエリグラフに対して最も頻繁に最高性能を達成するアルゴリズムを選択することができる. しかし, その方法ではクエリグラフごとの最適化を行わないためそれぞれのクエリごとに最良なアルゴリズムを選択できていない. そこで我々は, 類似した未知のクエリに対しても信頼性の高い推測を可能にするため, 構造的特徴量を用いて過去のクエリから機械学習モデルを学習させる方法を提案する.

4 提案手法

ここでは, 機械学習によって最良なサブグラフマッチングアルゴリズムを推測し, そのアルゴリズムでサブグラフマッチングを行う枠組みを提案する. 図 3 は概要を示している. 提案手法では多クラス分類を行っており, 各クラスはアルゴリズムを表している. また, 入力はクエリグラフとデータグラフの特徴量である. 機械学習モデルは, 与えられたクエリグラフとデータグラフの特徴量から最良なアルゴリズムが推測できるように学習する.

モデルの訓練時には, クエリ, データグラフから特徴量を抽出し, ラベリング戦略に従って各アルゴリズムの性能に対応したラベル付けを行う. 推論時には, 与えられたクエリ, データグラフから特徴量を抽出し, それらを入力として訓練済みの機械学習モデルに与える. モデルは最良なアルゴリズムを選択し, 提案手法ではそのアルゴリズムを用いてサブグラフマッチングを実行する.

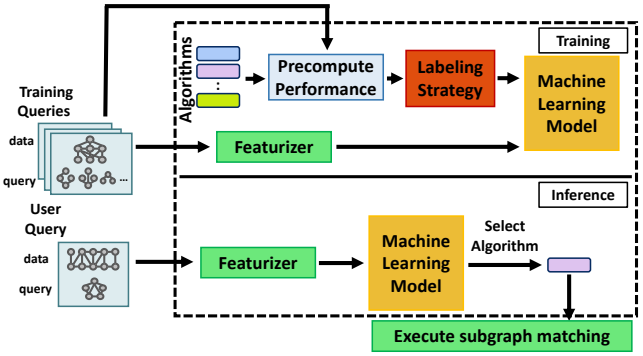


図 3: 提案手法の概要図.

4.1 ラベリング戦略

経験的データをモデルの訓練に使用できる形式にするために, ラベリング戦略ではそれらにラベルを割り当てる. あるクエリに対してアルゴリズムのラベル付けを行う際に, 3 章の分析結果に基づき, 最高性能を示すただ 1 つのアルゴリズムだけを選択するのではなく, 高い性能を示す複数のアルゴリズムに対してラベルを付与する.

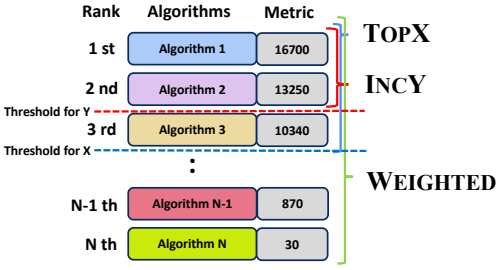


図 4: 3 つのラベリング戦略.

3 種類のラベリング戦略を提案する. TopX: 性能上位 X 個のアルゴリズムに対して正のラベルを割り当てる. IncY: 最も

良いアルゴリズムの性能に対して, $Y\%$ 以内の性能であるアルゴリズムに対して正のラベルを割り当てる. WEIGHTED: 相対的な性能として算出される値に基づき, 各アルゴリズムに正のラベルを割り当てる.

図 4 はそれぞれのラベリング戦略を表している. $X = 3$ とした TOPX では, 性能上位 3 つのアルゴリズムにラベル 1 を, それ以外には 0 を割り当てる. $Y = 0.7$ とした INCY では, EPS が $0.7 \cdot 16700 = 11690$ よりも高いアルゴリズムにラベル 1 を, それ以外に 0 を与える. WEIGHTED では, ラベル $1.0, 0.79, 0.62, 0.05, 0.002$ を 1 位, 2 位, 3 位, $(N-1)$ 位, N 位のアルゴリズムにそれぞれ割り当てる.

4.2 特徴量抽出器

特徴量抽出器は, クエリグラフとデータグラフから, それらを表現できる 13 個の構造的特徴量を抽出する. 抽出される特徴量には, クエリグラフとデータグラフの節点数 ($|V(q)|$, $|V(G)|$), クエリグラフの枝数 ($|E(q)|$), クエリグラフの平均次数, クエリグラフの直径, クエリグラフとデータグラフの最大コア数 (*degeneracy*), クエリグラフとデータグラフの密度 ($\frac{2|E(q)|}{|V(q)|(|V(q)|-1)}$, $\frac{2|E(G)|}{|V(G)|(|V(G)|-1)}$), 木幅, 節点ラベル集合のサイズ ($|\Sigma|$), データグラフの候補節点サイズ, そしてクエリグラフとデータグラフ間の節点ラベル比率が含まれる. 最後 2 つの特徴量は本研究で特有の特徴量であるため, 以下で定義する.

候補節点サイズとは, LDF フィルタリングアルゴリズム [20] によって計算される, サブグラフマッチングに関与する候補節点数の平均値である: $\frac{1}{|V(q)|}\sum_{u \in V(q)} |C(u)|$, ただし $|C(u)|$ は G における u の候補節点数である. 候補節点サイズの算出には, 単純なアルゴリズムで軽量であることから, LDF を用いている. 節点ラベル比率とは, クエリグラフとデータグラフにおける節点ラベル分布の内積であり, 次式で定義される: $\frac{\sum_{\ell \in \Sigma} |V(q)_\ell| \cdot |V(G)_\ell|}{|V(q)| \cdot |V(G)|}$, ただし $V(q)_\ell$ と $V(G)_\ell$ はそれぞれ節点ラベルが ℓ であるクエリグラフ, データグラフ内の節点集合を表している.

13 つの特徴量のうちデータグラフの密度, 節点ラベル集合のサイズ, 候補節点数, 木幅の 4 つは先行研究 [24] においてアルゴリズム選択のために活用されている.

4.3 モデルの訓練

ここでは, 機械学習モデルの訓練方法について記述する. 提案手法では, 以下に示される標準的な交差エントロピー損失を用いる: $\mathcal{L} = -\frac{1}{|Q|}\sum_{q \in Q}\sum_{i=1}^{|\mathcal{A}|} y_{q,i} \log \frac{\exp(\hat{y}_{q,i})}{\sum_{j=1}^{|\mathcal{A}|}\exp(\hat{y}_{q,j})}$ ただし, クエリ q に対するクラス $c \in [1, |\mathcal{A}|]$ に対して $y_{q,c}$ は正解ラベルを, $\hat{y}_{q,c}$ はモデルの出力値を表している. Q は訓練データに含まれるクエリの集合を表しており, $|Q|$ はその数を表している.

事前に設定したエポック数の間で, 損失関数を対象とした誤差逆伝播法を用いてパラメータを更新する. 各エポックの終了後, モデルの汎化性能を監視するために独立した検証セットを用いて性能評価を行う. さらに過学習を防ぐため, 学習の早期終了を採用する. すなわち, 検証セットにおける損失が一定のエポック数にわたって減少しない場合, 学習を終了する. 最終的なモデルは, 学習中に検証セットに対して最も低い損失値を達成したモデルを採用する.

5 評価実験

本実験では, 提案手法がサブグラフマッチングおよびサブグラフカバレッジにおいて優れた性能を発揮することを示し, 分布外 (Out-of-Distribution, OOD) のクエリグラフやデータセットに対する堅牢性を検証する. 加えて, 学習及び準備にかかるコストや, 各特徴量の重要性についても評価する.

5.1 実験設定

アルゴリズムと実験環境. サブグラフマッチングの実験において, アルゴリズムは各段階から表 1 に示されたアルゴリズムを用い, $5 \cdot 3 \cdot 5 = 75$ 通りの組合せを含むアルゴリズム群を対象とする. PILOS [16] は最新の強力なフィルタリング手法であり, 他の手法は全て先行研究 [24] で用いることを推奨されていたアルゴリズムである.

表 1: サブグラフマッチング: 各段階におけるアルゴリズム.

各段階	アルゴリズム
フィルタリング	LDF [20], NLF [25], DPISO [5], VEQ [9], PILOS [16]
順序決定	RI [3], RM [19], GQL [8]
埋込み	EXPLORE [20], LFTJ [21], QFILTER [6], KSS [23], VEQ [9]

サブグラフカバレッジの実験において, 表 2 に示されたアルゴリズムを用いる. MATCO [15] では, フィルタリングではクエリグラフとデータグラフの各節点のラベルが一致するかどうか, また順序決定ではクエリ節点の次数だけに着目して計算している. そこで, ここでは MATCO で用いられているそれらの単純なアルゴリズムを LAB, DEG と表記し, MIX&MATCH [17] で使用している枠組みで実装し, 合わせて埋込みを行う MATCO の中心的技術も実装する. また, MIX&MATCH で採用されているため, フィルタリングには VEQ [9], 順序決定には CFL [2] を用いる. 計 $2 \cdot 2 \cdot 3 = 12$ 通りの組合せからなるアルゴリズム群を対象とする.

表 2: サブグラフカバレッジ: 各段階におけるアルゴリズム.

各段階	アルゴリズム
フィルタリング	LAB [15], VEQ [9]
順序決定	DEG [15], CFL [2]
埋込み	MATCO [15], MIX&MATCH [17], MIX&MATCH-I [17]

データセット. 表 3 に示す 7 つのデータセットを用いる. これらは既存研究 (e.g., [18], [24]) で広く用いられている. Twitch は節点ラベル無しグラフであるため, 先行研究 [19] に従って無作為に一様に節点ラベルを付与した.

クエリセット. 部分誘導グラフ Q_I とスター型グラフ Q_S をクエリセットとして用いる. Q_I は, データグラフ内の 1 つの節点から開始し, 接続されている隣接節点へと枝を伸ばして移動するという操作を, 生成したいグラフの節点数に達するまで繰

表 3: データセット; k: 最大コア数.

データセット	記法	$\|V\|$	$\|E\|$	k	$\|\Sigma\|$	種類
Citeseer	cs	3,264	4,536	7	6	Citation
DBLP	db	317,080	1,049,866	113	14	Collab
Human	hm	4,674	86,282	148	43	Protein
Twitch	tw	168,114	6,797,557	149	45	Social
WordNet	wn	76,853	120,399	31	4	Lexical
Yeast	ys	3,112	12,519	10	70	Protein
Youtube	yt	1,134,890	2,987,624	51	23	Social

り返す. その後, 抽出された節点間を接続する全ての枝を抽出することで生成する. Q_S は, データグラフ内の 1 つの節点から開始し, その隣接節点へと枝を伸ばす. その後, 未訪問の隣接節点へと移動し, 生成したいグラフの節点数に達するまで繰り返すことで生成する. それぞれのクエリセットに対して, クエリに含まれる節点数を $i \in \{4, 8, 16, 32, 64, 96, 128\}$ としてクエリを生成する. 節点数 i を持つクエリの集合を Q_i と表記する.

各データセットから, 各 i の Q_I 及び Q_S についてそれぞれ 200 個のクエリを用意し (1 データセットあたり 2800 個), 合計で 19600 個 (2800×7) のクエリグラフ, データグラフの組を生成する. ここで, 全てのアルゴリズムが埋込みを見つけられなかったクエリは除外する. サブグラフカバレッジの実験では, 節点数 $i \in \{16, 32, 48, 64, 80, 96, 112, 128\}$ を対象とする. 節点数が少ないクエリは軽量であるため除外した.

全てのアルゴリズムは C++ で実装し, 一方で機械学習の実装は全て Python3 で開発した. C++ のコンパイルには, CMake 3.22.1 及び g++ 11.4.0 を使用した. 全ての実験は Ubuntu 22.04 LTS, Intel Xeon E5-2640 v4 @ 2.40GHz CPU を 2 基 (計 20 物理コア, 40 論理コア), 及び 1 TB の RAM を搭載したサーバ上で実施した.

経験的データの収集. 経験的データを収集するために事前に全てのクエリクエリグラフに対して全てのアルゴリズムを用いてサブグラフマッチングを実行する. その際, 埋込みには 10 秒の制限時間を設ける. 収集したデータはの一部は訓練データに, 残りは検証データ, テストデータとして使用する. 具体的には各データセットにおいて, 学習, 検証, テスト用データを任意に 6 : 2 : 2 の比率で分割する. これらのデータは一度の収集で良いため, 実験結果にはこの収集に費やした時間を考慮していない.

機械学習モデルとハイパーパラメータ. 本実験では, 簡単であり高速な学習が可能であることから, 機械学習モデルとして 2 層の多層パーセプトロン (MLP) を用いる. 学習には最適化手法として Adam [10] を用い, 隠れ層の活性化関数には ReLU を, 出力層には SoftMax を用いる. また, 学習の最大エポック数は 10000 とし, 学習の早期終了は 20 エポック以内に検証セットでの損失値が改善しない場合に行う. ハイパーパラメータは, 学習率は $[5e-4, 1e-3, 5e-3, 1e-2, 5e-2]$, ドロップアウト率は $[0.0, 0.1, 0.2, 0.3, 0.4, 0.5]$, 荷重減衰は $[0.0, 1e-5, 1e-4, 1e-3, 1e-2]$, 隠れ層のユニット数は $[16, 32, 64, 128]$ の中から調整する. また, それぞれのラベリング戦略にも固有のハイパーパラメータを採用している. TopX では X の値を $[1, 2, 3, 4, 5, 6, 7]$ の中から, IncY では Y の値を $[1.0, 0.95, 0.9, 0.85, 0.8, 0.75, 0.7, 0.65, 0.6, 0.55, 0.5]$ の中から調整する. また, ハイパーパラメータの調整には Optuna [1] を用いて, 検証セットに対する損失を最小化するように選択している.

各実験において, 学習データを用いてハイパーパラメータ調整を行い, 前述の探索範囲の中から, 検証セットにおける損失値を最小化する組合わせを選択する.

ベースライン. 本実験では以下のベースラインを用いる:

- Voted-D: 与えられたデータセットにおいて, 最も頻繁に最高性能を示したアルゴリズムを選択.
- Voted: 全てのデータセットにおいて, 最も頻繁に最高性能を示したアルゴリズムを選択.
- RecAlgo [24] : グラフの特徴に基づいて, 手動で設計されたルールをもとにフィルタリング, 順序決定, 埋込みの手法を選択する.

評価指標. サブグラフマッチングの実験において, 平均 EPS (Embeddings per Second) [24] を用いて評価する. これは, 単位時間あたりに処理された埋込み数を示すものであり, 実行時間には, フィルタリング, 順序決定, 埋込みにかかる時間が含まれている. さらに提案手法や RecAlgo には特徴量抽出にかかる時間やモデルの推論時間が実行時間に加わる. サブグラフカバレッジの実験においては, 平均カバレッジを用いて評価する. これは, 制限時間内に発見された埋込みに含まれる, 重複しない節点数の平均値を表す. また, EPS もカバレッジもクエリによってその値が大きく変動する. そのため, これらの値の平均をとる前に, 各クエリについて観測された最大値で除算することで, 各測定値を正規化する.

加えて, MRR を用いて, ある手法が真に最高性能のアルゴリズムを選択することにどれだけ近づいているかを評価する指標であり, $MRR = \frac{1}{|Q|} \sum_{q \in Q} \frac{1}{rank_q}$ によって計算される. ここで, $rank_q$ はクエリ q に対してその手法が選択したアルゴリズムの経験的データ内での順位を表す. 全ての実験は 5 つの異なるシード値を用いて行い, その平均値を記録する.

表 4: 各データセットにおける MRR.

MRR	cs	db	hm	tw	wn	ys	yt
Voted-D	0.2793	0.2912	0.1565	0.2540	0.2183	0.2269	0.2461
Voted	0.2793	0.3004	0.1528	0.0545	0.0717	0.2111	0.2461
RecAlgo	0.0812	0.0644	0.0970	0.1183	0.0929	0.0675	0.0614
Ours TopX	0.3396	**0.3570**	**0.2234**	**0.2765**	0.3045	0.2929	**0.3043**
Ours IncY	**0.3415**	0.3527	0.2189	0.2734	**0.3099**	**0.2961**	0.2933
Ours Weighted	0.3191	0.3442	0.2106	0.2520	0.2973	0.2886	0.2889

表 5: 各データセットにおける EPS.

EPS	cs	db	hm	tw	wn	ys	yt
Voted-D	**0.6066**	0.5566	0.4015	0.6273	0.4838	0.4207	0.5432
Voted	**0.6066**	0.5687	0.4996	0.1946	0.1353	**0.5364**	0.5432
RecAlgo	0.2656	0.1866	0.3490	0.2852	0.3034	0.2012	0.1947
Ours TopX	0.5611	**0.6405**	0.5656	0.6818	0.6376	0.3399	**0.6667**
Ours IncY	0.5196	0.6402	0.5616	0.6774	0.6360	0.3324	0.6566
Ours Weighted	0.5665	0.6323	**0.5853**	**0.6887**	**0.6578**	0.4370	0.6659

5.2 ベースラインとの評価実験

提案手法とベースラインを比較する. 表 4 および表 5 に平均 MRR と正規化された平均 EPS の値をそれぞれ示す. MRR に

おいて, 提案手法は全てのデータセットにおいてベースラインを上回り, 全データセットにおいて平均 0.2945 の MRR を達成している. これは, 平均して上位 3 位のアルゴリズムを選択できていることを示す. さらに, 提案手法ではベースラインが大きく性能を落としてしまうようなデータセットに対しても, ベースラインよりも高い EPS を達成している. 特に hm においては高い EPS を達成するアルゴリズムが多岐にわたるため (図 1 参照), VOTED-D は性能を落としてしまっている一方, 提案手法では高い性能を発揮できている. tw においても, 提案手法は高い性能を発揮している一方で, VOTED は性能を落としている. これは, このデータセットにおいて高い EPS を達成するアルゴリズムの傾向が他のデータセットと比べて大きく異なってしまうためである. さらに, RECALGO では全てのデータセットにおいて性能が劣っている. これはいくつかのクエリにおいて最も高い EPS を達成する PILOS が選択肢に含まれていないためである.

表 6: サブグラフマッチングにおける平均実行時間.

Datasets	cs	db	hm	tw	wn	ys	yt
Running time	4.52	7.96	8.14	9.61	11.40	2.98	9.42

EPS の評価では提案手法は 5 つのデータセット (db, hm, tw, wn, yt) でベースラインを上回っている. cs や ys においては, MRR ではベースラインを上回っていたが, EPS ではベースラインの方が高い性能を達成している. これは, 推論コストがオーバーヘッドになってしまうことに起因すると考える. 表 6 はフィルタリング, 順序決定, 埋込みにかかる実行時間の平均値をデータセットごとに示している. これより, cs や ys では実行時間が他のデータセットよりも短く, 推論コストがオーバーヘッドになってしまっていることがわかる.

提案手法の中では, MRR においては TOPX が INCY, WEIGHTED の性能を上回っているが, EPS においては WEIGHTED が上回っている. これは, WEIGHTED が高い EPS を達成している全てのアルゴリズムに性能に応じたラベルを付与することに対して, TOPX では高い EPS を達成する少数のアルゴリズムにのみラベルを付与しているためである. WEIGHTED は高い EPS を目指して設計されているため, ys のようにアルゴリズム間で大きな差のあるデータセットにおいて (図 2 参照), TOPX を大きく上回る.

全体を通して, 提案手法はベースラインよりも高い MRR と EPS を達成している.

5.3 分布外データセットに対する評価

分布外 (Out-of-Distribution, OOD) データセットに対する手法の性能評価も行い, 提案手法の汎化性を評価する. 実験では, 全 7 つのデータセットのうち 6 つから抽出したクエリを用いて学習と検証を行い, 残る 7 つ目のデータセットのクエリを用いた推論の結果を評価する. また, テストデータに含まれるデータセットが訓練データから除いているため, VOTED-D の結果を除いている. 図 5 は平均 MRR と正規化された平均 EPS の結果を示す. 6 つのデータセット (cs, db, hm, tw, ys, yt) では, 提案手法はベースラインに劣っている. これは, 後の 5.7 項でもある通り, データセット固有の特徴がモデルの性能に重要な役割を持っていることに起因する. しかし, wn においては提案手法はベースラインを上回っている. wn におけるベースラインの MRR の値は表 4 にある通り低いが, これは他のデータセットと比べて, 高い EPS を達成するアルゴリズムの傾向が大きく異なるためである (図 1 参照). 全体として, 提案手法は OOD データセットに対して性能を落としてしまうことがわかる. ただし, 同様に分布が大きく異なるデータセットではベースラインも性能を落とすことがわかる.

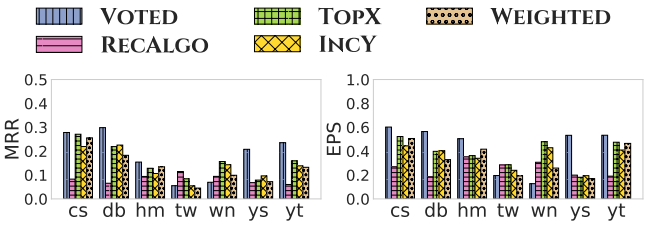


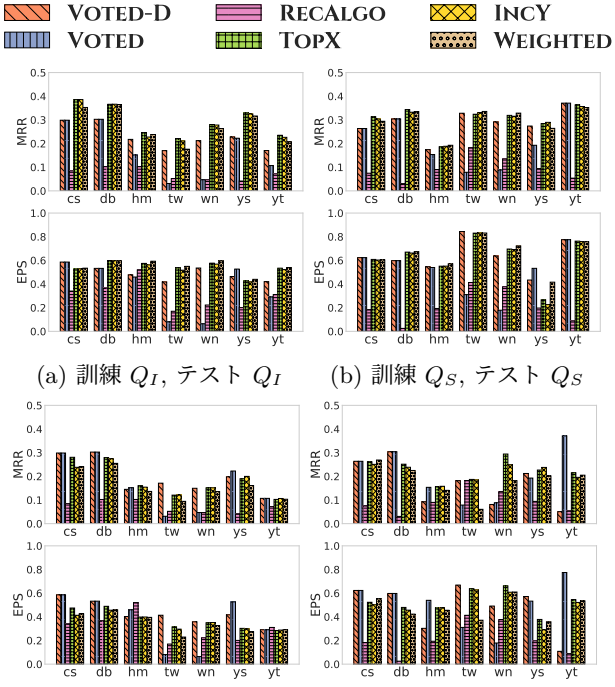
図 5: 分布外データセットにおける MRR と EPS.

5.4 訓練クエリセットの種類の影響評価

訓練データに含まれるクエリセットの影響を評価する. 図 6 は訓練データとテストデータに用いるクエリセットの種類 (Q_I, Q_S) の組合わせを入れ替えたときの平均 MRR と正規化された平均 EPS の結果を示す. 図 6 (a), (b) は訓練とテストデータのクエリセットの種類が一致する場合, 図 6 (c), (d) は種類が一致していない (すなわち OOD) 場合の結果を示す. 一致している場合, MRR においては yt 以外のデータセットでは提案手法がベースラインを上回っている. EPS においては Q_I では 5 つのデータセット (db, hm, tw, wn, yt) で提案手法がベースラインを上回っているが, Q_S では 3 つのデータセット (db, hm, wn) で提案手法がベースラインを上回っている. これは, Q_S では木幅が全て 1 となってしまうため, その分情報量が欠落していることに起因する. 一方, クエリの種類が一致しない場合, 提案手法は性能を落としてしまうことがわかる. これらの結果から, テストデータに含まれるクエリの構造的特徴が類似したクエリから訓練を行うことが重要であることがわかる.

5.5 分布外クエリサイズに対する評価

分布外 (OOD) クエリサイズに対する性能評価を行う. ここでは, 特定のサイズのクエリ ($i = 4, 8, 96, 128$) を訓練データから除き, 除かれたクエリをテストセットとする. 図 7 に平均 MRR と正規化された平均 EPS を示す. MRR の評価において, 提案手法は, 大きなクエリサイズ (Q_{96}, Q_{128}) をテストセットとしたときに db を除く 6 つのデータセットでベースラインを上回り, 小さなクエリサイズ (Q_4, Q_8) では 4 つ以上のデータセットでベースラインを上回る. 小さな OOD クエリサイズにおいて, VOTED-D や VOTED は, 大きな OOD クエリサイズに比べて性能が悪化する傾向にある. これは, 小さなクエリでは処理が容易になることが多いため, NLF のような軽量なアルゴリズムが高い EPS を達成する傾向にあるためである. ま



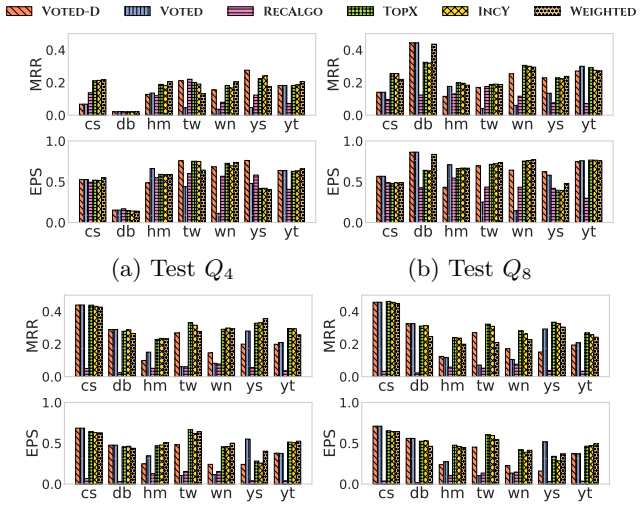
(a) 訓練 Q_I, テスト Q_I (b) 訓練 Q_S, テスト Q_S

(c) 訓練 Q_S, テスト Q_I (d) 訓練 Q_I, テスト Q_S

図 6: 訓練データとテストデータのクエリの種類による影響.

た EPS の評価においても, 特に大きなクエリサイズにおいて提案手法がベースラインを上回る. これは大きなクエリほど処理時間が増加し, 推論にかかる時間コストの影響が小さくなるためである.

これらの結果より, OOD クエリサイズに対して提案手法は優れた性能を示すことがわかる.



(a) Test Q_4 (b) Test Q_8

(c) Test Q_{96} (d) Test Q_{128}

図 7: 分布外クエリサイズに対する性能評価.

5.6 訓練データサイズの影響

訓練データのサイズを変化させたときの提案手法の性能を評価する. 基準として, 各データセットごとに訓練データにはおよそ 1600 個のクエリが含まれている. 実験では, テストセットのサイズを維持させた状態で訓練データサイズを $800, 400, 200, 10$ と減らす. 図 8 に各訓練データサイズにおける平均 MRR を示す. ただし, VOTED-D と VOTED は訓練データに基づいてアルゴリズムを選択するため訓練データサイズによって性能は変化することに対し, RECALGO は事前に決められたルールに基づいてアルゴリズムを選択するため性能は一定である.

MRR において, 訓練データサイズを 200 まで減らしても提案手法の性能は大きく変化しない. このことは, 訓練データの収集コストを削減できる可能性があることを示し, 5.8 項にてあらためて記述する. 訓練データサイズを 10 まで減少させたとき, 提案手法の性能は下がってしまうが, 3 つのデータセット (hm, wn, ys) においては依然としてベースラインを上回る. さらに, cs, db, tw ではベースラインとの性能差は最大でも 0.07% に収まる. これらの結果より, 提案手法は限られた訓練データを用いた場合でも高い性能を達成できることがわかる.

5.7 特徴量分析

RECALGO で用いられる特徴量の影響を調べるために, モデルの訓練に用いられる特徴量を 13 個から RECALGO で用いられる 4 つに絞った実験を行う. 図 9 では, 全ての特徴量を用いることでどれだけ性能が向上したかを表している. なお, 性能の向上率は $(MRR_{all} - MRR_{rec})/MRR_{rec}$ で計算され, MRR_{rec} は 4 つの特徴量を用いた時の MRR, MRR_{all} は全ての特徴量を用いた時の MRR を表す. tw を除く 6 つのデータセットでは, TOPX, INCY では全ての特徴量を用いたときに性能が大きく向上している. tw において, WEIGHTED ではわずかに性能が下がった. これらの結果からは, RECALGO で用いられている特徴量のみでは不十分であり, さらにルールベースの手法を手動で構築することの難しさを示している.

図 10 は Shapley 分析の結果を示している. 各特徴量がモデルの推論に与えた影響の大きさを表している. Shapley 値の計算において, 基準値の確立には計算の時間コストが高くなってしまう. そのため, 実験では およそ $10,000$ 個の訓練データから 100 個のサンプルを選択し, それをもとに基準値を確立した. この結果より, 我々が設計した節点ラベル比率が最も高い寄与度を持つことがわかる. また, クエリグラフから得られる特徴量よりもデータグラフから得られる特徴量の方が高い寄与度を持っていることがわかる. このことから, データグラフから得られる情報がアルゴリズムの選択により大きな影響を持つことを示している.

5.8 訓練と事前準備におけるコスト

機械学習の訓練に費やされる時間コストについて述べる. 表 7 には訓練にかかる平均時間と平均エポック数を示す. 提案手法では, 最大でも 100 秒程度の訓練時間に収まることがわかる. 訓練時間の分散が大きくなってしまうのは, ハイパーパラメータ調整の際に選択された学習率の違いが大きく影響している. 依然として, 提案手法における機械学習モデルの訓練が短い時間で終わることがわかる.

5.9 サブグラフカバレッジにおける実験

提案手法をサブグラフカバレッジの問題に適用する.

サブグラフカバレッジへの応用の動機付けとして, 7 つのデー

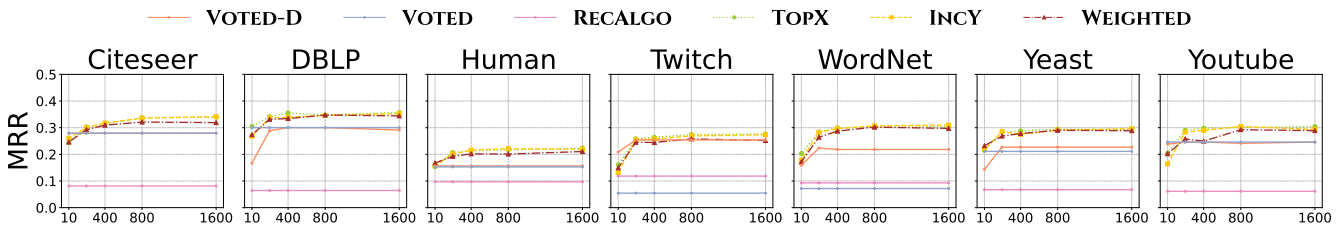


図 8: MRR に対する, 訓練データサイズの影響.

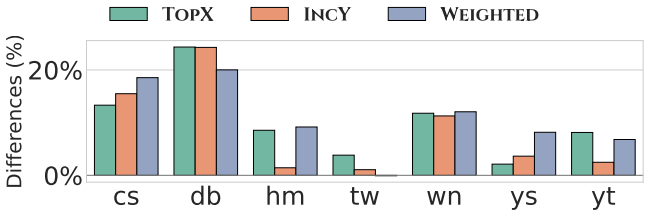


図 9: 特徴量の影響.

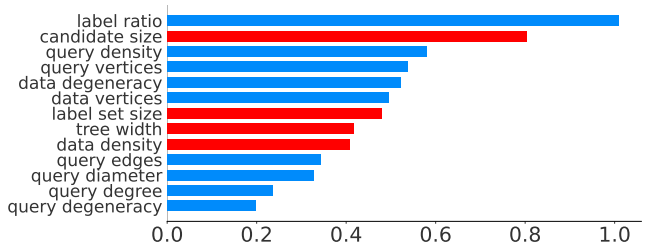


図 10: 平均 SHAP 値; 赤色は RECALGO での特徴量, 青色は追加した特徴量を表す.

表 7: 平均訓練時間と平均エポック数.

Ours	訓練時間 (秒)	エポック数
TOPX	$14.27_{\pm 4.83}$	$255.80_{\pm 92.61}$
INCY	$23.31_{\pm 5.31}$	$359.00_{\pm 82.88}$
WEIGHTED	$96.15_{\pm 140.39}$	$1629.40_{\pm 2473.25}$

図 11: データセットごとの, 最大カバレッジを達成したアルゴリズムの分布を示すヒートマップ; Yeast ではいくつかのアルゴリズムが最大カバレッジ数を達成している.

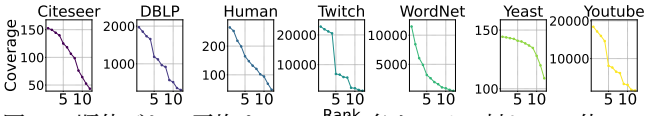


図 12: 順位ごとの平均カバレッジ; 各クエリに対して i 位のアルゴリズムを選択することで, ランク i におけるカバレッジを得ている.

タセットにおいて 12 種類のアルゴリズムがどのくらい最大カバレッジを達成したかを調べた. 図 11 は図 1 と同様に, その結果をヒートマップとして示している. 前処理後の制限時間 10 秒で複数のアルゴリズムがカバレッジを全て見つける可能性があるため, 同率 1 位が発生し, 結果として最良なアルゴリズムの数が多くなっている. それでも, cs や tw の間に見られるように, 最良アルゴリズムの傾向には違いが見られる. 図 12 は順位ごとの平均カバレッジを示す. EPS (図 2 参照) の場合と同様に, 平均カバレッジは順位に対して線形には低下せず, 上位と下位の差が大きくなる場合がある. 例えば, wn では 1 位のアルゴリズムと最下位のアルゴリズムの達成した平均カバレッジには, 10 倍以上もの差が生じている.

表 8 と 9 に平均 MRR と正規化された平均カバレッジの値をそれぞれ示す. MRR の評価では, 提案手法は 4 つのデータセットでベースラインを上回り, 0.6 以上の MRR を達成している. このことは, 提案手法が 1 位または 2 位のアルゴリズムを選択できていることを示す. また, 平均 MRR がサブグラフマッチングと比べて高いことは, アルゴリズムの選択肢が 12 と少なくなり, 選択することがより容易になったためである. さらに, カバレッジではいくつかのアルゴリズムが同率順位となる場合が多く, 特に ys では強くその傾向が見られる. 結果として, ほとんどの手法が非常に高い MRR を達成している. カバレッジでの評価では, 提案手法が 5 つのデータセット (cs, db, tw, wn, yt) でベースラインを上回る. hm, ys においても, ベースラインに近い, 高いカバレッジを達成している.

表 8: サブグラフカバレッジにおける MRR

MRR	cs	db	hm	tw	wn	ys	yt
VOTED-D	0.9276	0.5502	0.7880	0.5897	0.5654	0.9618	0.5527
VOTED	0.9177	0.4314	0.7880	0.3160	0.5654	0.9618	0.3208
Ours TOPX	**0.9350**	**0.6799**	**0.8258**	**0.8089**	**0.6536**	**0.9648**	**0.6935**
Ours INCY	0.9242	0.6667	0.8040	0.8039	0.6503	0.9567	0.6736
Ours WEIGHTED	0.9215	0.6692	0.7879	0.7983	0.6315	0.9539	0.6714

表 9: 正規化されたカバレッジ

Coverage	cs	db	hm	tw	wn	ys	yt
VOTED-D	0.9837	0.6432	**0.9211**	0.6285	0.6554	**0.9930**	0.6164
VOTED	0.9781	0.5918	**0.9211**	0.4408	0.6554	**0.9930**	0.5588
Ours TOPX	**0.9847**	0.8109	0.9013	**0.9198**	0.7188	0.9909	0.8360
Ours INCY	0.9791	0.5918	0.9059	0.9177	0.7228	0.9882	0.8340
Ours WEIGHTED	0.9811	**0.8225**	0.9158	0.9171	**0.7251**	0.9893	**0.8545**

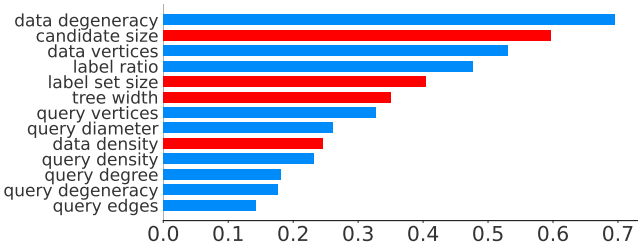


図 13: サブグラフカバレッジ: 平均 SHAP 値; 赤色は RECALGO での特徴量, 青色は追加した特徴量を表す.

図 13 は 5.7 項と同様に, Shapley 値を示している. サブグラフカバレッジの場合, データグラフのデジェネラシーがモデ

ルの推論に最も大きな寄与度を持っている一方, 候補節点サイズや節点ラベル比率の寄与度はサブグラフマッチングの場合と比べて, 小さくなっている. この結果は, 提案手法がそれぞれの問題ごとに特徴量と性能の間の特有の関係性を捉えていることを示している.

以上の結果から, 提案手法はサブグラフカバレッジにおいても多くの場合でベースラインを上回る性能であることがわかる. また, OOD データセットに対しても性能を維持できることを示した.

6 まとめ

本研究は, この問題を解く最初の研究であり, その可能性の調査に焦点を当てているため, より洗練されたモデルや, 学習データを生成するための方法を模索することで, 本研究を拡張する余地は大いにある. 将来の展望として, 機械学習モデル, 訓練データ, 特徴量の最適化に関する研究を行い, サブグラフマッチングやサブグラフカバレッジアルゴリズムの選択のためのモデルの構築を目指す.

謝辞

本研究は ASPIRE JPMJAP2328 および JSPS 科研費 JP23K28096 の支援によって行われた.

文献

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, pp. 2623–2631, 2019.

[2] Fei Bi, Lijun Chang, Xuemin Lin, Lu Qin, and Wenjie Zhang. Efficient subgraph matching by postponing cartesian products. In *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 2016.

[3] Vincenzo Bonnici, Rosalba Giugno, Alfredo Pulvirenti, Dennis Shasha, and Alfredo Ferro. A subgraph isomorphism algorithm and its application to biochemical data. *BMC Bioinformatics*, Vol. 14, pp. 1–13, 2013.

[4] Wenfei Fan. Graph pattern matching revised for social network analysis. In *Proc. Int. Conf. Database Theory*, pp. 8–21, 2012.

[5] Myoungji Han, Hyunjoon Kim, Geonmo Gu, Kunsoo Park, and Wook-Shin Han. Efficient subgraph matching: Harmonizing dynamic programming, adaptive matching order, and failing set together. In *Proc. ACM SIGMOD Int. Conf. Manag. Data*, pp. 1429–1446, 2019.

[6] Shuo Han, Lei Zou, and Jeffrey Xu Yu. Speeding up set intersections in graph algorithms using SIMD instructions. In *Proc. ACM SIGMOD Int. Conf. Manag. Data*, pp. 1587–1602, 2018.

[7] Juris Hartmanis. Computers and intractability: A guide to the theory of np-completeness. *SIAM Rev.*, Vol. 24, No. 1, p. 90, 1982.

[8] Huahai He and Ambuj K Singh. Graphs-at-a-time: Query language and access methods for graph databases. In *Proc. ACM SIGMOD Int. Conf. Manag. Data*, pp. 405–418, 2008.

[9] Hyunjoon Kim, Yunyoung Choi, Kunsoo Park, Xuemin Lin, Seok-Hee Hong, and Wook-Shin Han. Versatile equivalences: Speeding up subgraph query processing and subgraph matching. In *Proc. ACM SIGMOD Int. Conf. Manag. Data*, pp. 925–937, 2021.

[10] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. Int. Conf. Learn. Represent.*, 2015.

[11] Zixun Lan, Limin Yu, Linglong Yuan, Zili Wu, Qiang Niu, and Fei Ma. Sub-gmn: The neural subgraph matching network model. In *Proc. Int. Congr. Image Signal Process. BioMed. Eng. Inform.*, pp. 1–7, 2023.

[12] Ziming Li, Yuequn Dou, Youhuan Li, Xinhuan Chen, and Chuxu Zhang. Rsm: Reinforced subgraph matching framework with fine-grained operation based search plan. In *Proc. ACM Int. Conf. Web Search Data Min.*, pp. 475–483, 2025.

[13] Zhaoyu Lou, Jiaxuan You, Chengtao Wen, Arquimedes Canedo, Jure Leskovec, et al. Neural subgraph matching. *arXiv preprint arXiv:2007.03092*, 2020.

[14] Xiafei Qiu, Wubin Cen, Zhengping Qian, You Peng, Ying Zhang, Xuemin Lin, and Jingren Zhou. Real-time constrained cycle detection in large dynamic graphs. *Proc. VLDB Endow.*, Vol. 11, No. 12, pp. 1876–1888, 2018.

[15] Zhichao Shi, Youhuan Li, Ziming Li, Yuequn Dou, Xionghu Zhong, and Lei Zou. Matco: Computing match cover of subgraph query over graph data. *Proc. ACM Manag. Data*, pp. 1–24, 2025.

[16] Konstantinos Skitsas, Davide Mottin, and Panagiotis Karras. Pilos: Scalable large-subgraph matching by online spectral filtering. In *Proc. IEEE Int. Conf. Data Eng.*, pp. 1180–1193, 2025.

[17] Konstantinos Skitsas, Yuya Sasaki, Davide Mottin, and Panagiotis Karras. Mix & match: : Subgraph matching for absolute coverage. *Proc. VLDB Endow.*, Vol. 18, No. 13, pp. 5610–5622, 2025.

[18] Shixuan Sun and Qiong Luo. In-memory subgraph matching: An in-depth study. In *Proc. ACM SIGMOD Int. Conf. Manag. Data*, pp. 1083–1098, 2020.

[19] Shixuan Sun, Xibo Sun, Yulin Che, Qiong Luo, and Bingsheng He. Rapidmatch: A holistic approach to subgraph query processing. *Proc. VLDB Endow.*, Vol. 14, No. 2, pp. 176–188, 2020.

[20] Julian R Ullmann. An algorithm for subgraph isomorphism. *J. ACM*, Vol. 23, No. 1, pp. 31–42, 1976.

[21] Todd L. Veldhuizen. Leapfrog triejoin: A simple, worst-case optimal join algorithm. In *Proc. Int. Conf. Database Theory*, pp. 96–106, 2014.

[22] Linglin Yang, Lei Zou, and Chunshan Zhao. Neuso: Neural optimizer for subgraph queries. *arXiv preprint arXiv:2509.23775*, 2025.

[23] Rongjian Yang, Zhijie Zhang, Weiguo Zheng, and Jeffrey Xu Yu. Fast continuous subgraph matching over streaming graphs via backtracking reduction. *Proc. ACM Manag. Data*, Vol. 1, No. 1, pp. 15:1–15:26, 2023.

[24] Zhijie Zhang, Yujie Lu, Weiguo Zheng, and Xuemin Lin. A comprehensive survey and experimental study of subgraph matching: Trends, unbiasedness, and interaction. *Proc. ACM Manag. Data*, Vol. 2, No. 1, pp. 60:1–60:29, 2024.

[25] Gaoping Zhu, Xuemin Lin, Ke Zhu, Wenjie Zhang, and Jeffrey Xu Yu. Treespan: Efficiently computing similarity all-matching. In *Proc. ACM SIGMOD Int. Conf. Manag. Data*, pp. 529–540, 2012.

グラフモデルを基盤とする異種データベース統合システムの開発と評価

若林　拓†　常　穹†　宮崎　純†

† 東京科学大学情報理工学院情報工学系　〒 152–8550 東京都目黒区大岡山 2 丁目 12 − 1
E-mail: †wakabayashi.t.d410@m.isct.ac.jp, ††{q.chang,miyazaki}@comp.isct.ac.jp

あらまし　データの多様化に伴い、特性の異なる複数のデータベース（DB）を併用する機会が増えている一方で、それぞれが扱うクエリ言語やデータ型が異なるため、管理・操作の負荷増大が大きな課題となっている。本研究ではこの問題に対し、グラフデータベース、RDB、および主要な 3 種の NoSQL データベースを統合し、これらの一元的な操作を可能にするクエリ実行基盤を提案する。提案手法では、入力されたクエリを特定の言語に依存しない抽象化した演算子へと変換し、各 DB のネイティブクエリと結びつけるアプローチを採用した。これにより、個別の DB 実装に縛られることなく、異種 DB を跨ぐ複雑な連携を柔軟に組み立てることが可能となる。また、実行エンジン内に中間結果を保持・利用する仕組みを構築したことで、システム間の冗長なデータ移動を抑制し、複数の DB が関与する動的なクエリに対しても一貫した制御手順での実行を実現している。さらに、本システムは DB 間でデータを移行する機能を備えており、蓄積されたデータの特性や要件の変化に応じた再配置が可能である。本研究では、これら一連の機能を備えた実行基盤の実装を通じて、異種データベースが混在する分散環境においてもクエリが正しく動作する実行環境を構築し、クエリ実行順序の制御やデータ配置の変更を柔軟に行うことを可能とした。

キーワード　polystore system,NoSQL データベース, プロパティグラフ, データ統合技術

1 はじめに

情報社会の進展に伴い、テキストや時系列、グラフデータといった多種多様なデータが混在し、その規模も増大し続けている。取り扱うデータの性質や要件に応じて、構造化データを管理する RDB、製品カタログ等の文書データに適したドキュメント DB、友人関係等のつながりを表現するグラフ DB など、最適なデータモデルは異なる。

特にグラフデータベースはエンティティ間の複雑な関係性をノードとエッジで柔軟に表現できる一方、特有の課題も存在する。例えば、膨大な属性データがメモリを占有することで、グラフ探索に必要なメモリ展開領域が阻害されるという問題がある。また、グラフデータベースは構造中心のインデックス設計を採用しているため、大規模な属性フィルタリング処理には適していない。こうした単一データベースによる対応の限界を克服するため、複数のデータベースを併用し、データの性質に応じた適材適所の配置を実現する Polystore System が必要とされている。

しかし、Polystore System の運用においても問い合わせ最適化の困難さという課題が残されている。各データベースごとに性能特性が異なるため、単一 DB 向けの最適化手法が通用しない。特にグラフデータにおけるパス検索は計算の前後関係が構造的に固定されるため、他の DB とは異なる最適化アプローチが求められる。

これらの課題に対し、本研究ではグラフデータベースを主軸とし、肥大化した属性情報を他の異種データベースへ分散配置するという先行研究のアプローチを継承する。具体的には、関係探索に特化したグラフ構造のみをグラフデータベースで管理し、それに付随する膨大な属性情報を KVS や RDB 等へ切り出して保持させることで、グラフ探索におけるメモリ効率の向上を図るものである。

本研究ではこの構成をさらに発展させ、複雑な構造条件を含むユーザークエリを特定の言語に依存しない共通形式へと変換する実行基盤を導入した。この共通形式を介することで、異種データベース間を跨ぐ一連の処理順序を論理レベルで統合的に制御することが可能となる。これにより、将来的な実行計画の最適化を容易にする土台を築くとともに、異種分散環境における効率的なクエリ実行を可能とする管理基盤を実現する。

2 関連研究

本節では提案手法においてデータモデルとして活用する U-Schema と先行研究において提案された polystore system をいくつか紹介する。

2.1 U-Schema

U-Schema [1] は RDB と 4 つの主要な NoSQL データベースであるグラフ DB、ドキュメント DB、カラム指向 DB、キーバリューストアに対する統一的なスキーマモデルである。これは Entity-Relation モデルに基づいて構成されており、データオブジェクトをエンティティタイプ、エンティティの相互関係をリレーションタイプとして定義している。またエンティティの種類を示すデータラベルが同じだが、異なるプロパティを持つような状態 structural variation として記述する。これにより NoSQL データベースが持つスキーマレスな性質が反映されるようになっている。

2.2 CloudMdsQL

CloudMdsQL [2] は SQL をベースとした拡張クエリ言語を提供するクエリエンジンであり、RDB、ドキュメント DB、グラフ DB に対応している。SQL クエリの中にアクセス対象の DB のネイティブ関数をサブクエリとして直接埋め込むことで、統合的なアクセスを実現している。

しかしデータの保存先をクエリ内で明示する必要があり、データ配置の透過性は低い。またデータマイグレーションも導入されていない。

2.3 Léa らの手法

Léa ら [3] は RDB、ドキュメント DB、グラフ DB に対応している polystore system を提案している。このシステムでは、Entity-Relationship モデルに基づいて全体のデータモデルが構築されており、これを用いてシステム側が各データベースにおけるデータ配置を統一的に把握できるようになっている。これによりユーザがデータ配置を意識することのない透過的なクエリを実現している。また、対応する 3 種類のデータベースのネイティブクエリを受け付ける設計となっており、受け取ったクエリは問い合わせ木に変換される。この問い合わせ木を用いることで、実行順序の入れ替えによるクエリ処理最適化ができるようになっている。

一方で、このシステムにはデータマイグレーション機能が組み込まれておらず、異種データベース間でのデータ配置の最適化は考慮されていないという課題が残されている。

2.4 山下らの手法

山下らが提案したシステム [4] はグラフ DB とキーバリューストアに対応しているシステムであり、データモデル全体をグラフ構造として捉えるようにしている。

本手法では U-Schema [1] を利用してデータ配置を把握し、これに基づいてマッピング辞書というファイルを作成している。マッピング辞書は JSON 形式で記述され、各エンティティが持つプロパティのデータ配置についての情報を保持している。システムはクエリ実行時にこのマッピング辞書を参照することでアクセス対象のデータの位置を特定できる。これにより、分散されたデータベース間においても透過的なクエリを実現している。

また本手法では、キーバリューストアのデータモデルを統合するために転置キーを作成している。キーバリューストアでは、キーに対応するバリューを取得する方法でデータを抽出する。ここで用いるキーは、エンティティまたはリレーションシップのラベル、ID、およびプロパティのラベルから構成される複合キーである。一方で、分散されたプロパティを統合的に管理するためには、バリューを基点として対応する ID を取得する機能が求められる。そこでエンティティまたはリレーションシップのラベル、プロパティのラベル、およびバリューから構成される転置キーを作成し、これを対応する ID の集合と関連付ける。この転置キーを用いることでキーバリューストア上で特定のプロパティ値を持つエンティティやリレーションシップを効率的に抽出することが可能となる。

さらにデータマイグレーションも実装しており、この機能を用いてキーバリューストアにプロパティデータを分散配置させることでクエリ処理性能が向上することを実験的に示している。

本手法は U-Schema の設計上さらに RDB、ドキュメント DB、カラム指向 DB に対応を広げられると考えられる。またグラフ DB を軸とする設計上の観点からグラフクエリしか受け付けないが、分散配置されたプロパティにアクセスする際に生じるクエリの変換の機能は一部にとどまっており、より複雑な構文に対応するよう拡張が求められる。加えて現状のクエリ変換方式では実行順序をシステム側が制御することは困難であり、またグラフ探索途中に生成される中間結果の活用することができない。このため異種 DB をまたがる問い合わせ最適化を設計しづらいという課題が残されている。

3 提案手法

本研究では山下らの提案システム [4] を拡張し、U-Schema [1] を用いた統一的なスキーマ管理とデータ配置の透過性を実現する。既存手法 [2] [3] では困難であった透過的クエリ処理や異種データベース間のデータ移行機能を継承しつつ、本研究では対応データベースをドキュメント DB、カラム指向 DB、リレーショナル DB に拡充した。さらに、特定のクエリ言語に依存しない「抽象化オペレーター」と「中間結果保持機構」を新たに導入することで、論理レベルでの処理の共通化と冗長な計算コストの削減を図り、クエリ実行基盤を高度化させている。これにより、データの性質に応じた適材適所の分散配置を可能にし、大規模グラフデータに対する柔軟で拡張性の高い管理基盤を構築した。図 1 に提案システム全体の構成を示す。

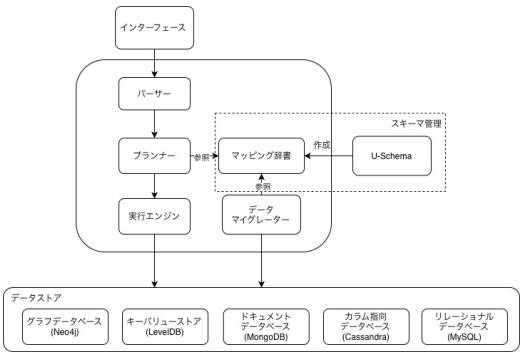


図 1: システムの構成

3.1 データ構造

このシステムにおいて、データ全体をプロパティグラフとして捉える。プロパティグラフとはグラフ構造を基盤としたデータモデルであり、特徴としてエンティティだけでなくリレーションシップ自体もプロパティを保持できる点が挙げられる。

提案手法では、エンティティおよびリレーションシップから構成されるグラフ構造についての情報をグラフ DB に格納し、それらが保持するプロパティについてはデータの性質や利用目

的に応じて異種データベース間に分散配置する。グラフ構造はRDB 等でも管理可能だが、探索時には再帰的な結合に伴う計算コストの増大が避けられない。加えて、密に結合したノード群を集約させるような最適分割は困難であり、結果として DB 間通信を頻発させる要因となる。こうした背景を踏まえ、本研究ではグラフ構造をグラフ DB だけで管理することでグラフ探索の効率を確保する。

また分散されたプロパティデータを統合的に扱うため、各エンティティおよびリレーションシップに対して ID を付与する。異種データベースに格納されたプロパティは、この ID を共通の参照キーとして管理される。これにより、ID を基点として同一のエンティティまたはリレーションシップに属するデータを特定でき、分散環境下においてもデータ全体を論理的に一貫したプロパティグラフとして把握することが可能となる。

3.2 マッピング辞書

山下らの手法 [4] においてマッピング辞書はエンティティを軸として構成されており、リレーションシップについてはどのエンティティ同士を接続しているかという情報しか保持されておらず、エンティティに付随するものとして管理されている。本システムにおいてはリレーションシップが持つプロパティも分散されることを想定しており、またグラフデータベースに対してクエリを発行する際ににエンティティとリレーションシップは異なる方法で記述する必要がある。以上の理由から本システムではリレーションシップを独立した要素としてマッピング辞書に記述する。

3.3 クエリ処理

提案するシステムはグラフデータベースである Neo4j のクエリ言語である Cypher によって書かれたクエリをユーザクエリとしている。クエリ処理は、プラン作成段階と実行段階の二つのフェーズから構成される。

3.3.1 プラン作成段階

プラン作成段階では、まずユーザークエリとして与えられた Cypher クエリをパースし、クエリ構造を抽象構文木として取得する。次にこのパース結果を基に実行プランを生成する。実行プランは Neo4j のクエリ処理方式を参考に設計しており、EntityScan、Expand、VarLengthExpand、Filter、Projection のオペレーターからなる線形なオペレーション列として表現される。またこの実行プラン生成の過程において、マッピング辞書を参照し各オペレーターがアクセスすべきデータベースを決定する。図 2 はクエリをオペレーターに変換した例を示している。

以下に、各オペレーターの挙動について説明する。

1.EntityScan

クエリ処理における探索の起点となるエンティティを取得するオペレーターである。システムに登録されたエンティティ全体の中から、クエリで指定されたエンティティのラベルやプロパティ条件に基づいて候補を選択する。本オペレーターは、探索対象となるエンティティを初期段階で限定することで、後続処理における探索範囲を削減する。出力は条件を満たすエンティティの ID 集合であり、以降のオペレーターはこの ID を基準として処理を行う。

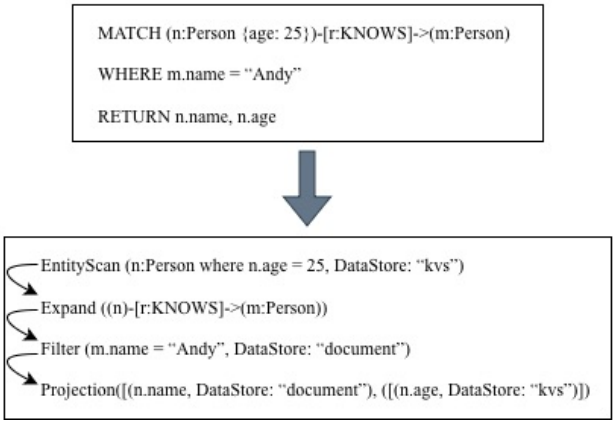


図 2: プラン作成の例

2.Expand

直前のオペレーションによって得られたエンティティを起点として、リレーションシップを 1 ホップ分探索するオペレーションである。エンティティ間の接続関係を辿ることで、新たなエンティティおよび対応するリレーションシップを取得する。これはグラフ構造に基づく探索を担うオペレーターであり、グラフ構造の情報を参照する必要がある。したがって本手法においては、グラフ構造を管理しているグラフデータベース上でのみ実行される。

3.VarLengthExpand

直前のオペレーターによって与えられたソースエンティティを起点として、可変長のマルチホップ探索を行うオペレーターである。単一ホップに限定されない探索を可能とすることで、複雑なグラフ構造に対するパス検索を可能とする。本オペレーターも Expand と同様、グラフデータベース上でのみ実行される。

挙動の詳細として、指定されたホップ数の探索を一括で実行する。その際、最終的な出力として始点と終点の重複のないペアを取得する仕様としており、この重複排除によって中間結果の削減を図っている。

一方で、本オペレーターは内部的には探索の過程で通過するエンティティおよびリレーションシップを取得しているため、これらの中間データに基づいた枝刈りの実装も可能である。ただし枝刈りの拡張を行う場合には、道中の情報を条件評価を用いるために経路情報を保持する必要があり、上述した始点と終点のみに基づいた重複排除という仕様が必ずしも適用できない場合がある。

4.Filter

それ以前のオペレーターによって生成された中間結果に対して、クエリで指定された条件を適用し、処理対象をさらに限定するオペレーターである。構造探索などによって得られた候補から、条件を満たすエンティティおよびリレーションシップのみを選択することで、後続のオペレーターの効率化を図る。

直前のオペレーターから受け取った中間結果を入力とし、そ

こに含まれる ID に対応するエンティティおよびリレーションシップにプロパティ条件を適用する。条件を満たさない ID は中間結果から除外され、評価後に残った ID 集合が次のオペレーターへ引き渡される。

5.Projection

クエリ処理の最終段階として実行され、先行するオペレーターによって選択されたエンティティおよびリレーションシップを対象に、処理結果をユーザーに提示する形式へと変換するオペレーターである。

挙動としては、先行するオペレーターから受け取った中間結果を入力とし、そこに含まれる ID に基づいて対応するエンティティおよびリレーションシップが保持する必要なプロパティ値を取得する。

3.3.2 実行段階

実行段階では、生成された実行プランに従って各オペレーターを順番に実行し、必要なデータを取得して最終的なクエリ結果を出力する。この際、expand 以外の各オペレーターについては、プラン作成段階で決定されたデータベースに対応するクエリが動的に生成され、実行される。一方で、Expand オペレーターと VarLengthExpand オペレーターはグラフ DB への直接アクセスとして実行される。各オペレーター間では、エンティティおよびリレーションシップの ID で構成されたテーブルを受け渡すことで中間結果を管理している。

3.4 データマイグレーション

さらに柔軟なデータ配置を実現するための機能として、山下らの提案システム [4] と同様にデータマイグレーションを導入する。本手法におけるデータマイグレーションとは、エンティティおよびリレーションシップが保持するプロパティを異なるデータベース間で移行可能とする仕組みであり、データ配置の変更による最適化を行うことを目的に導入している。提案手法では、グラフ構造自体はグラフデータベースに保持されたまま、プロパティのみが分散配置される。そのため、プロパティの配置を変更することで、クエリ特性や取り扱うデータの性質に応じた柔軟なデータ管理が可能となる。この仕組みをグラフデータベースとキーバリューストアの間だけでなく、RDB、ドキュメント DB、カラム指向 DB とも双方向にマイグレーションできるよう拡張している。データマイグレーションの実装は、U-Schema [1] における各データモデルとの対応づけをもとに行った。

4 評価実験

4.1 実験の目的

本実験では、提案手法に基づくシステム全体の基本的な動作を検証するとともに、現状における性能上のボトルネックを明らかにすることを目的とする。具体的には、分散配置されたデータが設計通りに正しく取得・統合されるかを確認した上で、処理段階ごとの実行時間を計測し、処理時間の集中箇所を把握する。さらに、将来的なデータ配置の最適化に向けた知見を得るため、データ配置ごとの性能特性についても分析する。これにより、現段階におけるシステムの性能特性を把握し、大規模データへの対応に向けた具体的な課題を抽出する。

4.2 使用したデータセットおよびクエリ

データセットとして、SNS を模した標準ベンチマークである LDBC Social Network Benchmark [5] を使用する。本研究が想定するデータ構成は大規模かつ多様な属性を有するものであるが、現時点における提案手法には、処理能力および対応可能なデータ規模に制約がある。本データセットは、パラメータ調整によって中間結果の生成量を容易に制御できる特性を有しており、手法の完成度が十分でない現状においても、負荷を適切に管理した状態で基本動作の検証が可能である。このような特性から、本研究では本データセットを実験用データとして選定した。

データセットの規模、およびラベルごとのデータ件数を表 1 に示す。

表 1: データセットの構成

要素	データ数
全ノード数	2,997,352
全エッジ数	34,393,552
ノード: Person	10,295
ノード: Organisation	7,955
ノード: Place	1,460
エッジ: KNOWS	346,028
エッジ: WORK_AT	44,088

また実験に用いるクエリについては、LDBC が定義する読み取り専用のワークロードに基づき、以下の条件を持つものを抽出して実行する。

- **探索範囲**: 特定の `personId` を持つノードを起点とする 1〜2 ホップ以内の知人 (`KNOWS`) 関係
- **構造条件**: 知人が `WORK_AT` 関係を持つ組織（`Company`）に所属し、その組織が特定の国（`countryName`）に対して `IS_LOCATED_IN` 関係を持つこと。
- **抽出条件**: 勤務開始年が指定された値（`workFrom`）未満である
- **出力**: 知人の ID、名、姓、会社名、入社年を射影し、入社年（昇順）、知人の ID（昇順）、会社名（降順）の優先順位でソートした上位 10 件を出力する

またこの条件に基づいて記述された Cypher クエリを図 3 に、このクエリを提案手法に基づいてオペレーターに分解した結果を表 2 に示す。

図 3: 実際に使用する Cypher クエリ

```
MATCH (p:Person {id:$personId})
-[:KNOWS*1..2]-(friend:Person)
-[work:WORK_AT]->(comp:Organisation {type: "Company"})
-[:IS_LOCATED_IN]->(:Place {type: "Country",
name: $countryName}) WHERE work.workFrom < $work-
FromYear RETURN friend.id, friend.firstName, friend.lastName,
comp.name, work.workFrom
ORDER BY work.workFrom ASC, friend.id ASC, comp.name
DESC LIMIT 10
```

4.3 評価手法

提案手法の評価にあたり、グラフ探索以外の演算で参照されるプロパティ（personId、workFrom 等）をすべて特定のデータベースへ配置し、それぞれの構成における実行性能を計測した。また比較用ベースラインとして、同一クエリを直接 Neo4j [7] へ送信した場合の処理時間を計測した。計測に際しては、OS および各データベースのキャッシュ機構に起因する計測値の変動を抑制するために同一条件下で 10 回の試行を実施した。本実験ではその算術平均を実行時間として採用する。

4.4 実験環境

提案システムは Go1.24.4 を用いて実装した。ユーザーがシステムに対して発行するクエリの記述言語には、主要なグラフデータベースサービスの一つである Neo4j [7] で用いられているクエリ言語 Cypher を採用している。システム内のパーサーには言語認識ツールである ANTLR4 [6] を用いている。

また表 3 に実験に使用したマシンの環境、表 4 に本システムの実装に伴い各データモデルごとに利用したデータベースを示す。

表 3: マシン環境

項目	仕様
OS	macOS 15.6 (Sequoia)
CPU	Apple M1 Pro (8-core CPU)
Memory	16 GB (Unified Memory)

表 4: 使用データベース環境

データモデル	データベース (バージョン)
グラフ	Neo4j 5.24.0 [7]
ドキュメント	MongoDB v8.0.10 [8]
キーバリューストア	LevelDB v1.0.0 [9]
カラム指向	Cassandra 6.2.0 [10]
リレーショナル	MySQL 9.3.0 [11]

4.5 実験結果

図 4 に、ベースラインおよび提案手法の各データ配置構成における全体の実行時間を示す。まず、提案手法に基づく全ての配置構成において、ベースラインと同等の出力結果が得られることを確認した。一方で実行時間については、全ての配置構成においてベースラインである Neo4j に直接クエリを送信する場合よりも大幅に増大していることが確認できる。

次に提案手法の各データ配置構成ごとの詳細を見ていく。表 5 は、実行プラン上の各ステップにおける処理時間をデータ配置ごとに示したものである。

計測結果を確認すると、グラフ構造の走査を担う Step2 (VarLengthExpand) および Step3・6 (Expand) においては、すべての構成で実行時間が概ね同等であった。

また、配置変更の影響を受ける Step1 (EntityScan)、Step4・5・7 (Filter)、および Step8 (Projection) においては、選択したデータモデルによって顕著な性能差が現れている。

例えば約 15000 件の中間結果に対してプロパティ評価を行う Step4 において最良値を示したキーバリューストア配置の 47.65ms に対し、最悪値となったカラム指向 DB 配置では 1357.54ms と約 28 倍の時間を要しており、このステップにおける処理時間が全体の大部分を占める結果となった。

また最終的な出力を生成する Step8 (Projection) においては、ドキュメント DB の 18.97 ms やキーバリューストアの 2.37 ms に対しカラム指向 DB は 0.29 ms と極めて低い値を記録した。

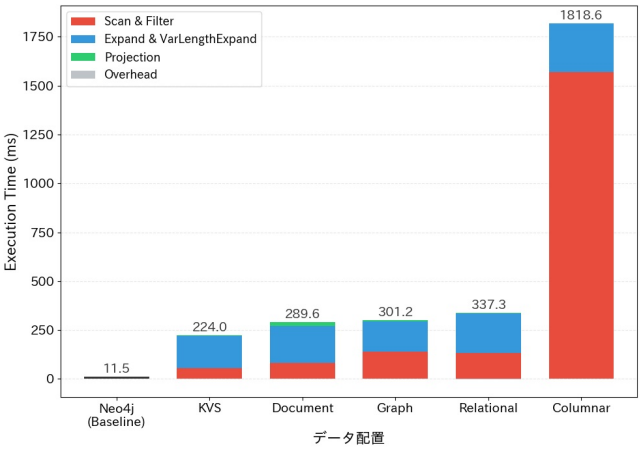


図 4: 全体実行時間比較

4.6 考　　察

まずベースラインから大幅に実行時間が増大した要因として、外部データベースとの通信コストに加え、各オペレーターの実行ごとに発生する中間結果保持のためのメモリ確保およびデータコピーのオーバーヘッドがあげられる。単一バッファ内で処理が完結するベースラインに対し、提案手法ではオペレーターごとに新しいメモリ領域を割り当てる構造であるため、中間結果が膨大になるほど管理負荷が実行時間を押し上げる結果となった。

各データ配置間の比較では、キーバリューストア配置が比較的安定した性能を示した。これは、外部キーに基づく参照がキーバリューストアの単一キーアクセスの特性と合致し、かつ複合キーの設計により取得範囲を物理レベルで最小化できたためと考えられる。

表 2: 各実行ステップにおける処理内容とアクセス先の対応

Step	オペレーター	役割	アクセス先
1	EntityScan	personId に基づく起点ノードの特定	各データベース
2	VarLengthExpand	KNOWS 関係に基づく可変長パス探索 (1-2 ホップ)	グラフ構造 (Neo4j)
3	Expand	WORK_AT 関係に基づく単一ホップの構造参照	グラフ構造 (Neo4j)
4	Filter	プロパティ値（workFrom）に基づく比較演算と削減	各データベース
5	Filter	プロパティ値 (type) に基づく比較演算と削減	各データベース
6	Expand	IS_LOCATED_IN 関係に基づく単一ホップの構造参照	グラフ構造 (Neo4j)
7	Filter	プロパティ値 (name) に基づく比較演算と削減	各データベース
8	Projection	最終的な出力の構成と出力プロパティの射影 (firstName, etc.)	各データベース

表 5: 提案手法における各データ配置構成のステップ別実行詳細

Step / Location	Graph (ms)	KVS (ms)	Document (ms)	Columnar (ms)	Relational (ms)	Rows
Total Latency	301.22	223.96	289.56	1818.63	337.32	-
1. EntityScan	2.18	0.03	4.05	73.45	4.45	1
2. VarLengthExpand	33.87	35.70	45.31	62.42	45.73	7,354
3. Expand	102.45	109.51	111.22	133.48	125.00	15,741
4. Filter	124.93	47.65	72.90	1357.54	115.27	10,459
5. Filter	11.98	5.94	5.02	124.03	8.81	10,459
6. Expand	16.86	21.38	30.31	53.09	30.93	10,459
7. Filter	1.39	0.91	1.31	13.73	1.38	381
8. Projection	6.94	2.37	18.97	0.29	5.26	10

対照的に、RDB 配置やドキュメント DB 配置、グラフ DB 配置では、特定のプロパティ抽出時においてもエンティティ単位のデータ管理構造がボトルネックとなった可能性がある。具体的には、目的外のプロパティを含むデータブロック全体の読み出しや、レコード単位の排他制御といった付随的な管理コストが発生する。こうした特性が、純粋なキー参照に特化した KVS と比較して実行時間を増大させた要因と考えられる。

一方、カラム指向 DB 配置においては、外部キーに基づくインデックス作成が Projection の効率化に大きく寄与した。インデックスを通じて必要なカラムの格納場所を直接特定し、不要なプロパティの I/O をスキップして目的の列データのみを選択的に読み出せたことが、高速化の主因と推測される。

しかし、Filter においては、インデックス未定義のプロパティに対する全行走査（ALLOW FILTERING）が致命的な遅延を招いた。これは、独立して格納された各列を突き合わせる際のデシリアライズ負荷が、処理件数に比例して累積したためと分析する。

5 おわりに

本研究では、U-Schema [1] に基づき分散配置されたプロパティグラフを統合管理し、異種データベース環境においてクエリを段階的に処理するシステムを構築した。先行研究 [4] のデータ移行や透過的クエリ機能を継承しつつ、新たに抽象化オペレーターを導入したことで、特定言語に依存しない実行基盤の高度化を実現している。本システムは、ユーザークエリをオペレーターへ変換し、中間結果を内部で再利用することで、データベース間を跨ぐ柔軟なクエリ実行を可能にした。また、対応データベースをドキュメント DB、カラム指向 DB、および RDB へと拡張し、より多くの性質を活用できるシステムを実現している。

一方、実験では中間結果の受け渡しを伴う逐次実行や、各データベースの特性を十分に活かせていない実装に起因する性能低下が確認された。そのため、中間処理の効率化や各データモデルに応じた実装の最適化を図る必要がある。

また、本手法はグラフ構造自体の分割管理を実装するまでには至っておらず、処理可能なクエリ表現も限定的な範囲に留まっている。今後は、データ規模の拡大や分析要求の高度化に適応するためのさらなる機能拡充に取り組む必要がある。

以上の課題に取り組んでいくことで、大規模グラフを異種分散環境において効率的に扱う Polystore system へと発展させていく予定である。

謝　辞

本研究は JST CREST JPMJCR22M2 ならびに JSPS 科研費 JP23K28091,JP23K28383,JP25H01167 の助成を受けたものである。

文　献

[1] Carlos J. Fernández Candel, Diego Sevilla Ruiz, and Jesús J. García-Molina. A unified metamodel for nosql and relational databases. *Information Systems*, Vol.104, p.101898, 2022.

[2] Boyan Kolev, Patrick Valduriez, Carlyna Bondiombouy, Ricardo Jiménez-Peris, Raquel Pau, and José Pereira. CloudMdsQL: querying heterogeneous cloud data stores with a common language. *Distributed and parallel databases*, Vol. 34, p.463–503, 2016.

[3] Léa El Ahdab, Imen Megdiche, André. Peninou, and Olivier Teste. Unified Models and Framework for Querying Distributed Data Across Polystores. *International Conference on Research Challenges in Information Science*, p.3–18. Springer, 2024.

[4] Fumihiro Yamashita, Qiong Chang, Jun Miyazaki. Unified Schema-Driven Graph Polystore: Achieving Transparency

in Multi-model Integration and Migration. Database and Expert Systems Applications: 36th International Conference, DEXA 2025, Proceedings, Part II. 2025, p. 136-143.
[5] Renzo Angles,János Benjamin Antal,Alex Averbuch,Altan Birler,Peter Boncz,Márton Búr,et al. The LDBC social network benchmark. arXivpreprint arXiv:2001.02299, 2020
[6] antlr4-go. ANTLR v4 Go runtime library. https://github.com/antlr4-go/antlr/v4 (accessed Jan 30, 2026)
[7] Neo4j, Inc. "Neo4j Graph Database." https://neo4j.com/ (accessed Jan 30, 2026).
[8] MongoDB, Inc. "MongoDB: The Developer Data Platform." https://www.mongodb.com/ (accessed Jan 30, 2026).
[9] Google. "LevelDB: A fast key-value storage library written at Google." https://github.com/google/leveldb (accessed Jan 30, 2026)
[10] The Apache Software Foundation. "Apache Cassandra." https://cassandra.apache.org/ (accessed Jan 30, 2026).
[11] Oracle Corporation. "MySQL: The world's most popular open source database." https://www.mysql.com/ (accessed Jan 30, 2026)

意味埋め込みを用いた連邦型知識グラフ問合せにおける情報源選択

林田　康太†　　天笠　俊之††

† 筑波大学 情報学群 情報科学類　〒 305–8573 茨城県つくば市天王台 1 丁目 1-1
†† 筑波大学 計算科学研究センター　〒 305–8577 茨城県つくば市天王台 1 丁目 1-1
E-mail: hayashida@kde.cs.tsukuba.ac.jp, amagasa@cs.tsukuba.ac.jp

あらまし　連邦 SPARQL 問合せでは，クエリ実行前の情報源選択が性能に大きく影響する．従来手法は述語・型の有無に基づく構造的アプローチが主流だが，同一スキーマを持つ情報源が複数存在する場合，構造情報だけでは優先順位を付けることが困難である．本研究では，意味埋め込みに基づいて情報源をランキングする手法を提案し，RDF エンティティと SPARQL クエリを共通のテキスト規約で表現し，SBERT により意味埋め込み空間へ写像する．情報源ごとに階層的クラスタリングで要約し，クエリとの類似度とカバレッジに基づいてランク付けする．疑似 KG を用いた評価で，提案手法はベースライン手法と比較して，順位相関と網羅性の向上を確認した

キーワード　連邦型知識グラフ，RDF，SPARQL，情報源選択

1　序　　論

DBpedia や Wikidata に代表される大規模知識グラフ（KG）は RDF として公開され，SPARQL を通じて利用できる．知識グラフは推薦システムや質問応答など幅広い分野で活用されており，近年では LLM と知識グラフを組み合わせた GraphRAG [2] も注目されている．しかし，知識グラフは分散して存在し，データ量やアクセス制御の観点から単一の知識グラフとして統合することは難しい．さらに，分野別専門 KG や機関ごとの KG など，同一のオントロジーを共有しながらもトピックごとに分割された知識グラフ群が増えている．複数の KG を横断して問合せを行う連邦 SPARQL 問合せでは，全ての情報源に対して問合せを送る全探索はレイテンシや通信量の増大を招くため，クエリ実行前に関連する情報源を絞り込む情報源選択が不可欠である．

従来の情報源選択手法は，述語・型の有無に基づく構造的アプローチが主流である．HiBISCuS [5] や MAPS [6] は，クエリに含まれる構造情報を用いて候補情報源を絞り込む代表的手法である．しかし，構造ベース手法の多くは無関係な情報源の除外を主な目的としており，絞り込み後も複数の情報源が候補に残る場合に，どの情報源を優先すべきかを決定しにくい．例えば，すべての情報源が `dbo:Person` や `dbo:birthPlace` といった汎用的な型・述語を持つ場合，構造索引では多くが一様に候補として選ばれる一方で，どの情報源が「科学者」に特化し，どの情報源が「政治家」に特化しているかを判別できない．このような場合，情報源ごとにエンティティの意味分布が偏っていても，構造情報だけでは関連度の高い情報源を適切にランク付けできない．

そこで本研究では，構造情報では区別できない情報源間の優先順位付けを目的とし，意味埋め込みに基づく情報源ランキング手法を提案する．具体的には，RDF エンティティと SPARQL クエリを共通のテキスト規約で記述し，SBERT により同一埋め込み空間へ写像する．さらに，情報源ごとに階層的クラスタリングでエンティティ分布を要約し，クエリとの類似度とカバレッジに基づいて情報源をランク付けする．

2　前提知識および関連研究

2.1　連邦 SPARQL 問合せ

RDF は主語・述語・目的語のトリプルで知識を表現し，SPARQL はトリプルパターンの集合（Basic Graph Pattern; BGP）を用いて問合せを記述する．連邦 SPARQL 問合せでは，複数の SPARQL endpoint を横断して問合せを実行する．

2.2　BERT に基づく文埋め込み

BERT を基盤とする事前学習モデルは，文をベクトルに変換する文埋め込み手法として広く用いられている．中でも Sentence-BERT（SBERT）[4] は，文対タスクに基づいて文埋め込みを学習し，cos 類似度による類似検索に適することが報告されている．文を事前にベクトル化することで高速な近傍探索が可能となる．本研究では，cos 類似度に基づく近傍探索とクラスタリングのため SBERT を用いる．

2.3　構造情報に基づく情報源選択

連邦型 SPARQL 問合せにおける情報源選択では，RDF の構造情報に基づいて関連情報源を推定する手法が提案されてきた．HiBISCuS [5] は，クエリに含まれる述語・型・URI を保持する情報源を選択する手法であり，述語や型の有無に基づいて候補を絞り込む．一方，MAPS [6] は，述語間の隣接関係（接続パターン）を索引化し，クエリの BGP 形状と適合する情報源を選択する手法である．

2.4　意味的埋め込み手法

知識グラフ埋め込み手法は，グラフ構造ベースと言語モデルベースに大別される．代表的な手法に TransE [1] などがあるが，グラフ構造ベースの手法は意味情報を明示的に扱わず，言

語モデルベースの手法もトリプル単位の妥当性判定を目的としている．いずれも情報源全体の意味分布要約や適合度の効率計算に必要な索引構造を持たず，連邦型知識グラフ問合せの情報源選択への直接適用は困難である．

2.5 本研究の位置づけ

本研究は，構造ベース手法が苦手とする「同一スキーマ下での意味分布の偏り」に着目し，意味埋め込みに基づく情報源ランキング手法を提案する．SBERT と階層的クラスタリングにより情報源の意味分布を要約し，クエリとの類似度に基づいてランク付けすることで，従来の構造ベース手法を補完する．

3 問題設定

3.1 対象とする環境

関連研究で述べたように，従来の構造ベース手法は，述語・型・BGP 形状が共通である場合，情報源の絞り込みはできても適切な順位付けが難しい．本研究では，以下の条件を満たす環境を対象とする．

条件 1: 同一スキーマ（述語・型・BGP 形状が共通）

全ての情報源が同じ述語（例: `dbo:birthPlace`）と型（例: `dbo:Scientist`, `dbo:Artist`）を保持している．したがって，構造ベースの索引（HiBISCuS, MAPS 等）では，全情報源が「候補」として選ばれるが，優先順位を付けることができない．

条件 2: 意味分布の偏り（トピック混合比率が異なる）

各情報源は意味的トピック（例: 科学者，芸術家，競技者，政治家）の混合比率が異なる．同じトピックを含むが主トピックが異なるため，構造索引では区別できない．

3.2 情報源ランキング問題の定式化

上記の条件下では，構造情報だけでは情報源間の優先順位を決定できない．クエリが「科学者を探したい」という意図を持つ場合，科学者を 40%含む情報源 s_1 を上位に推奨すべきだが，構造索引では s_1 と s_2 の科学者混合比率を区別できない．そこで本研究では，情報源選択を「どの情報源をどの順で問合せるべきか」を決定する情報源ランキング問題として捉える．

入力: クエリ q，情報源集合 $V = \{s_1, s_2, \ldots, s_N\}$

出力: 情報源ランキング $\pi(q) = [s_{i_1}, s_{i_2}, \ldots, s_{i_N}]$

目的: クエリに対して「適合度の高い情報源」を上位に推奨する．適合度は，その情報源に含まれるクエリ条件を満たすエンティティ数として定義する．

4 提案手法

4.1 全体構成

提案手法は，索引構築フェーズとクエリ処理フェーズから構成される．索引構築フェーズ（図 1）では，各情報源のエンティティを共通テキスト規約に従ってテキスト化し，SBERT で埋め込みを生成後，階層的クラスタリングで要約する．クエリ処理フェーズ（図 2）では，クエリを同規約でテキスト化して埋め込み，階層探索でクエリに近いクラスタを選択し，類似度とカバレッジに基づいて情報源をランク付けする．

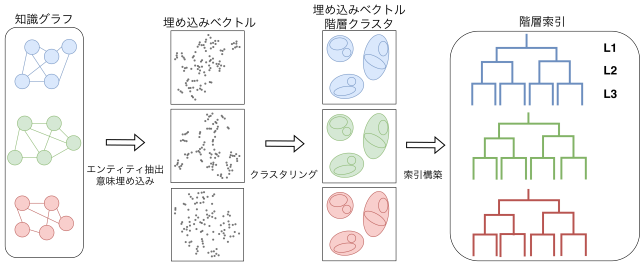


図 1 索引構築フェーズの処理

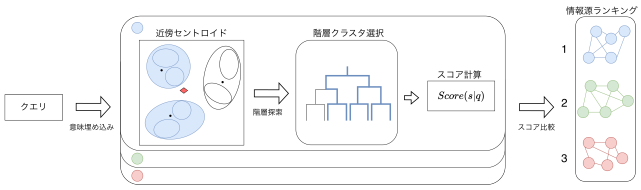


図 2 クエリ処理フェーズの処理

4.2 エンティティとクエリのテキスト化

エンティティとクエリを共通の埋め込み空間へ写像するため，共通テキスト規約を定義する．エンティティは，ラベル，型（最大 3 個），説明文，主要な述語–値の列の順で連結してテキスト化する：`{LABEL}. type {TYPE}. {DESC}`

例えば，Albert Einstein は次のようにテキスト化される：`Albert Einstein. type Scientist. German-born physicist. birth place Ulm.`

クエリ側も同様に，SPARQL から型・述語・値を抽出し，エンティティ側に近い語彙で記述する．例えば，「ロンドン生まれの科学者」を検索するクエリは `type Scientist. birth place London.` のようにテキスト化される．このように，エンティティと同じ文体に揃えることで，Transformer がクエリとエンティティの意味的類似度を適切に計算できるようにする．

4.3 索引構築

各情報源 s は，自身のエンティティ集合 E_s から生成した埋め込みベクトル群に対し，個別に階層的クラスタリングを適用して階層索引を構築する．具体的には，埋め込みベクトルに PCA により次元削減と L2 正規化を適用した後，階層的な Spherical k-means [3] クラスタリングを実行する．まず全エンティティを L1 レベルの粗いクラスタに分割し，次に各 L1 クラスタを L2 レベルの細かいクラスタへ再分割する．各レベルの各クラスタ c について，セントロイド $\boldsymbol{\mu}_c$（L2 正規化された平均ベクトル）とカバレッジ w_c（含まれるエンティティ数）を計算する．こうして，情報源 s は階層索引として要約される．

この索引構築は情報源ごとに完全に独立しており，並列化が容易である．さらに，特定情報源の更新が他へ影響しないため，分散環境に適した設計となっている．

4.4 クエリ処理

クエリ処理フェーズでは，図 2 に示すように，(1) クエリの

Algorithm 1: 階層探索と情報源スコアリング

Input : クエリ埋め込み $\mathbf{v}_q$，各情報源 s の階層セントロイド $C_s^{(1)}, C_s^{(2)}$，カバレッジ $w_s^{(2)}$，パラメータ k_{l1}, k_{l2}

Output: 情報源ランキング $\pi(q)$

1 **for** *each* **情報源** $s \in V$ **do**

2 $I_1 \leftarrow \mathrm{TopK}(\mathbf{v}_q^\top C_s^{(1)}, k_{l1})$;

3 $L_s \leftarrow \mathrm{TopK}(\mathbf{v}_q^\top C_{s,\mathrm{child}(I_1)}^{(2)}, k_{l2})$;

4 $\mathrm{Score}(s|q) \leftarrow \sum_{c \in L_s} \max(\mathbf{v}_q^\top \boldsymbol{\mu}_{s,c}^{(2)}, 0) \cdot w_{s,c}^{(2)}$;

5 $\pi(q) \leftarrow \mathrm{SortDesc}(\{(s, \mathrm{Score}(s|q)) \mid s \in V\})$;

意味埋め込み，(2) 各情報源での近傍セントロイド選択（階層クラスタ選択），(3) 情報源ごとのスコア計算，(4) スコア比較による情報源ランキング，の 4 段階で構成される．

まず，SPARQL クエリをテキスト化し，SBERT により意味埋め込み $\mathbf{v}_q$ を生成する．次に，各情報源 s に対して独立に階層探索を実行し，クエリに関連する L2 クラスタ（細かい階層）を選択する．アルゴリズム 1 に，この階層探索と情報源スコアリングの手順を示す．なお，$\mathbf{v}_q$ および各セントロイド $\boldsymbol{\mu}_c$ は L2 正規化されているため，内積 $\mathbf{v}_q^\top \boldsymbol{\mu}_c$ はコサイン類似度に等しい．

階層探索の詳細は以下の通りである．各情報源 s に対し，まず Level 1（粗い階層）において，クエリ埋め込み $\mathbf{v}_q$ と L1 セントロイド群 $C_s^{(1)}$ との類似度を計算し，上位 k_{l1} 個のクラスタ I_1 を選択する（アルゴリズム line 2）．次に，選択された L1 クラスタの子クラスタ（Level 2）を候補として収集し，その中からクエリとの類似度が最も高い上位 k_{l2} 個の L2 クラスタ L_s を選択する（line 3）．

選択された k_{l2} 個のクラスタに対し，クエリとの類似度（負の場合は 0）とカバレッジ w_c（含まれるエンティティ数）の積和により情報源スコアを計算する（line 4）：

$$\mathrm{Score}(s|q) = \sum_{c \in L_s} \max(\mathbf{v}_q^\top \boldsymbol{\mu}_c, 0) \cdot w_c \qquad (1)$$

このスコアは，「クエリに近い意味領域」に「多くのエンティティ」を持つ情報源ほど高く評価する．全情報源をこのスコア順に降順ソートし，ランキング $\pi(q)$ として出力する（line 5）．

5 評価実験

5.1 実験設定

提案手法の有効性を検証するため，意味的トピック分布を情報源ごとに偏らせた疑似知識グラフを用いて評価を行う．本研究では，情報源ランキングの精度を主眼とするため，SPARQL を実エンドポイントで実行せず，埋め込み空間上でのランキング品質を評価する．本実験では，構造索引では区別できない環境（同一スキーマ・異なる意味分布）を意図的に構築し，疑似 KG によりトピック混合比率と適合エンティティ数を厳密に制御することで，ランキング精度の定量評価を可能とする．

表 1 実験パラメータ

項目	値
モデル	`all-mpnet-base-v2`
次元数（Raw/PCA）	768 / 128
エンティティ数	4,000
情報源数	4
クラスタリング	L1(k=4) → L2(k=4)
探索パラメータ	k_{l1}=4, k_{l2}=7

表 2 疑似 KG の構成（トピック混合比率）

情報源 ID	科学者	芸術家	競技者	政治家
`kg_s`	**40%**	30%	20%	10%
`kg_a`	10%	**40%**	30%	20%
`kg_h`	20%	10%	**40%**	30%
`kg_p`	30%	20%	10%	**40%**

5.2 実験環境

実験環境を以下に示し，実験で用いた主なパラメータを表 1 に示す．

基本環境

- OS: macOS
- CPU: Apple M2 (8 cores)
- 実装言語: Python 3.13

GPU 環境（SBERT 埋め込み生成に使用）

- GPU: NVIDIA GeForce RTX 4080
- CUDA: 12.8

5.3 データセット

DBpedia 2022.12.01[1] から抽出した 4,000 エンティティを用い，4 つの疑似知識グラフ（各 1,000 エンティティ）を構築した．各知識グラフは，4 つのトピック（科学者，芸術家，競技者，政治家）を異なる混合比率で含む（表 2）．全ての情報源が同じ述語・型を持つため，構造索引では区別できないが，トピック混合比率の違いにより意味的適合度が異なる．

5.4 評価クエリ

各トピック（科学者，芸術家，競技者，政治家）に対し，特定の場所（birthPlace）で出生した人物を探すクエリを生成した．計 40 クエリ（4 トピック × 10 場所）を用意した．例えば，科学者 × ロンドンのクエリは以下の通りである．

```
SELECT ?p WHERE {
  ?p rdf:type dbo:Scientist .
  ?p dbo:birthPlace dbr:London .
}
```

このクエリに対し，kg_s が最も高い適合度（40%）を持ち，次いで kg_p（30%），kg_h（20%），kg_a（10%）の順となる．

5.5 ベースライン手法

提案手法と比較するため，グローバルセントロイド法をベー

1：https://databus.dbpedia.org/dbpedia/collections/dbpedia-snapshot-2022-12

スラインとして定義する．各情報源 s のエンティティ埋め込み集合を 1 本のベクトル（グローバルセントロイド）μ_s に要約し，クエリ埋め込み v_q との cos 類似度で情報源を順位付けする．

5.6 評価指標

情報源ランキングの精度を以下の指標で評価する．**NDCG@k**: 上位 k 件の累積利得を正規化した指標．情報源 s の利得 $\mathrm{rel}(s,q)$ は，その情報源に含まれるクエリ条件を満たすエンティティ数とする．**Kendall の順位相関係数** τ: 予測順位と正解順位のペア間の一致・不一致の比率に基づく指標で，値域は $[-1,1]$ である．**Spearman の順位相関係数** ρ: 予測順位と正解順位の順位差に基づく相関係数で，値域は $[-1,1]$ である．いずれも 1 に近いほど全順位の整合性が高いことを示す．**Recall**: 適合度 $\mathrm{rel}(s,q)>0$ の情報源のうち，$\mathrm{Score}(s|q)>0$ で返された情報源の割合．

5.7 実験結果

表 3　提案手法とベースライン手法の比較

手法	N@1	N@3	K-τ	S-ρ	Rec.
ベースライン	1.00	0.949	0.767	0.860	0.513
提案手法	1.00	1.00	0.992	0.995	1.00

表 3 に，提案手法とベースライン手法（グローバルセントロイド法）の比較を示す．

両手法とも最上位 1 件の推奨は完璧だが（NDCG@1=1.00），提案手法は全順位で高い整合性を示す．特に，順位相関指標では提案手法が Kendall τ=0.992，Spearman ρ=0.995 を達成し，ベースライン（τ=0.767, ρ=0.860）を大幅に上回る．また，提案手法は Recall=1.00 で関連する全情報源を漏れなく検出できるのに対し，ベースラインは Recall=0.513 と約半数の情報源を取りこぼす．

考察: グローバルセントロイド法は情報源を単一の平均ベクトルで表現するため，トピック混合比率の違いを捉えにくい．本実験では主トピック比率 40%の影響により平均ベクトルは主トピック方向に偏るが，混合比率の差（10%刻み）が十分大きいため，主トピックの違いだけでもある程度の順位付けが可能となり，高い順位相関（τ=0.767）を達成した．しかし，副トピック中心の情報源（トピック比率 20%, 10%）では，平均ベクトルとの類似度が閾値を下回り Score=0 となるケースが多く，Recall=0.513（約半数の情報源を取りこぼす）という低い網羅性にとどまった．また，3 位と 4 位（比率 20%と 10%）は平均ベクトルの差が小さく順位が逆転しやすい．

対して提案手法は，情報源を階層的クラスタ構造（L1=4, L2=16）で表現し，各クラスタのセントロイドとカバレッジ（エンティティ数）を保持する．クエリ処理では，全 L1 を探索して L2 候補（最大 16 個）を確保し，上位 k_{l2}=7 個のクラスタのみをスコア計算に使用する．この k_{l2}=7 は実験的に最適化された値であり，k_{l2}=3 では情報不足（NDCG@1=0.85），k_{l2}=10 では完璧（τ=1.00）だが，k_{l2}=7 でもほぼ同等（τ=0.992）の性能を達成する．階層構造により，主トピックだけでなく副トピック（20%, 10%）を含む多様な意味領域を個別のクラスタとして捕捉できるため，全情報源で Score >0 となり Recall=1.00 を達成した．

6 結論と今後の課題

本研究では，連邦 SPARQL 問合せにおける情報源選択に対し，意味埋め込みに基づいて情報源を優先順位付きでランキングする枠組みを提案した．RDF エンティティと SPARQL クエリを共通のテキスト規約で表現し，SBERT により意味埋め込みへ変換する．さらに，情報源ごとに階層的クラスタリングでエンティティ分布を要約し，クエリとの類似度とカバレッジに基づいて情報源をランク付けする．疑似 KG を用いた評価で，提案手法はベースライン（グローバルセントロイド法）と比較して，全情報源の検出（Recall=1.00 vs. 0.513）と全順位の一致度（Kendall τ=0.992, Spearman ρ=0.995）において大幅な向上を達成した．

今後の課題として，以下の 2 点が挙げられる．第一に，本研究では制御された疑似ベンチマークで評価したが，実世界の LOD データセット（DBpedia，Wikidata 等）での評価や，自然言語クエリ・複雑な SPARQL パターンへの対応が必要である．第二に，本研究は意味的類似度に基づくランキングに焦点を当てたが，構造ベース手法（HiBISCuS, MAPS 等）との統合により，構造と意味の両面から情報源を評価するハイブリッド戦略の構築が今後の発展方向である．

謝　辞

本研究の一部は，国立研究開発法人新エネルギー・産業技術総合開発機構（NEDO）「ポスト 5G 情報通信システム基盤強化研究開発事業」（JPNP20017），JST CREST（JPMJCR22M2），および科学研究費補助金（JP23K24949）の支援を受けて実施された．

文　献

[1] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Proc. NeurIPS 2013*, pp. 2787–2795, 2013.

[2] Yucheng Han, Yuxuan Wang, et al. Graph retrieval-augmented generation: A survey. *arXiv preprint arXiv:2408.08921*, 2024.

[3] Kurt Hornik, Ingo Feinerer, Martin Kober, and Christian Buchta. Spherical k-means clustering. *Journal of Statistical Software*, Vol. 50, No. 10, pp. 1–22, 2012.

[4] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proc. EMNLP-IJCNLP 2019*, pp. 3982–3992, 2019.

[5] Muhammad Saleem and Axel-Cyrille Ngonga Ngomo. HiBISCuS: Hypergraph-based source selection for SPARQL endpoint federation. In *The Semantic Web – ESWC 2014*, Vol. 8465 of *Lecture Notes in Computer Science*, pp. 176–191. Springer, 2014.

[6] 小倉勇大, 増田正, 天笠俊之. 連邦型 rdf 問合せのための隣接述語索引に基づく情報源選択手法の改良. 第 17 回データ工学と情報マネジメントに関するフォーラム（DEIM 2025）, 2025.