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

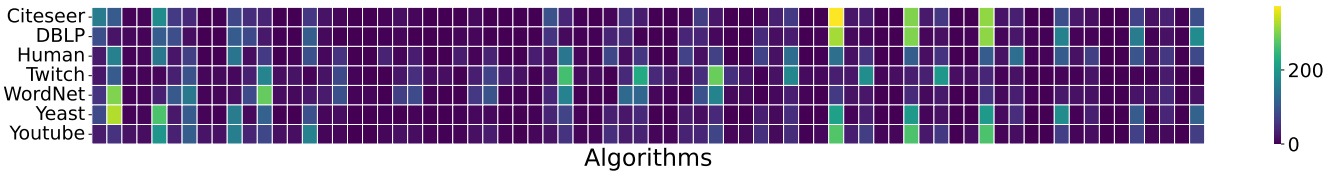


図 1: データセットごとの, 最大 EPS を達成したアルゴリズムの分布を示すヒートマップ.

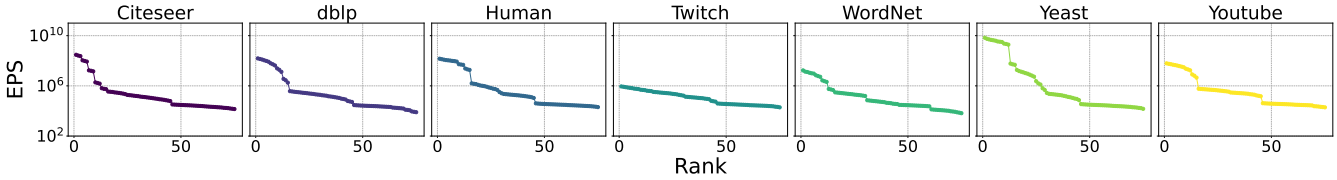


図 2: 順位ごとの平均 EPS; 各クエリに対して i 位のアルゴリズムを選択することで, ランク i 位における EPS を得ている.

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.

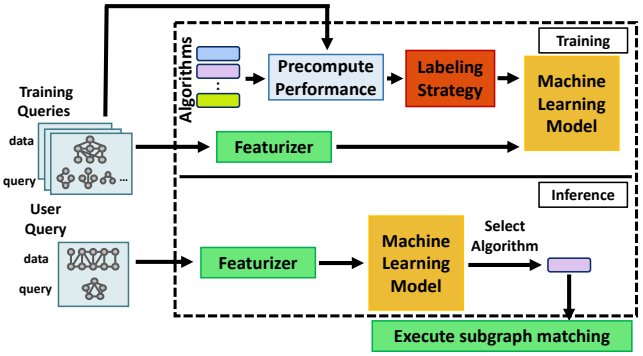


図 3: 提案手法の概要図.

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.

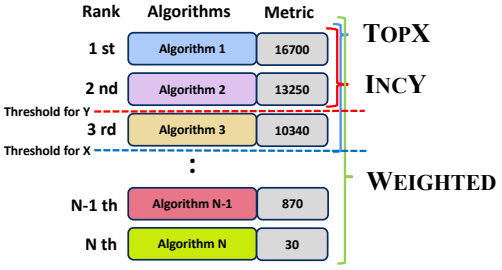


図 4: 3 つのラベリング戦略.

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.

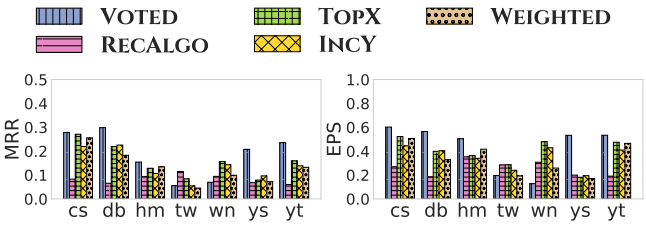


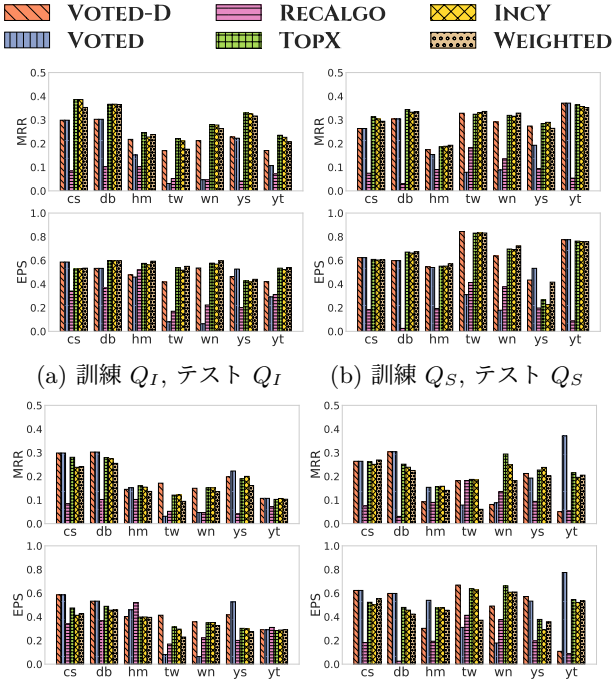
図 5: 分布外データセットにおける MRR と EPS.

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



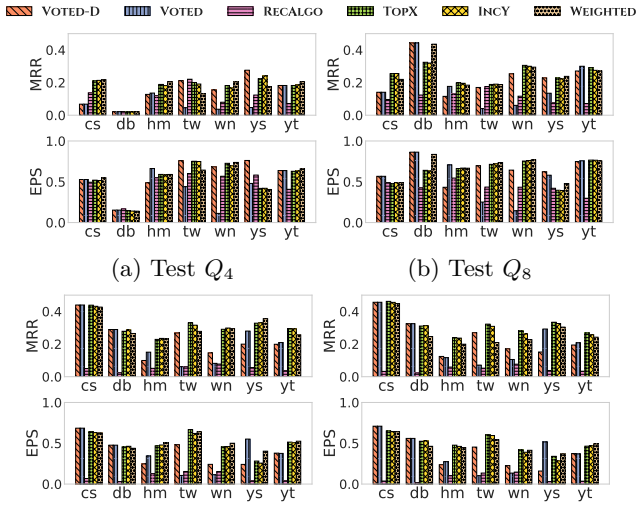
(a) 訓練 Q_I, テスト Q_I (b) 訓練 Q_S, テスト Q_S

(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

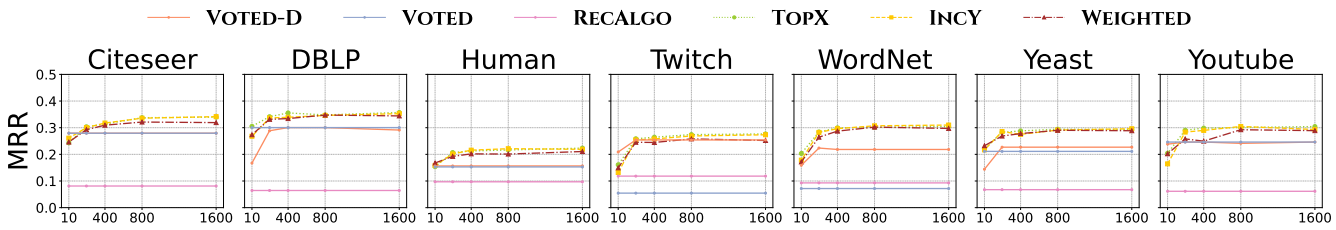


図 8: MRR に対する, 訓練データサイズの影響.

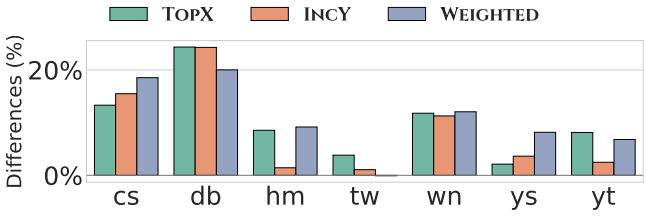


図 9: 特徴量の影響.

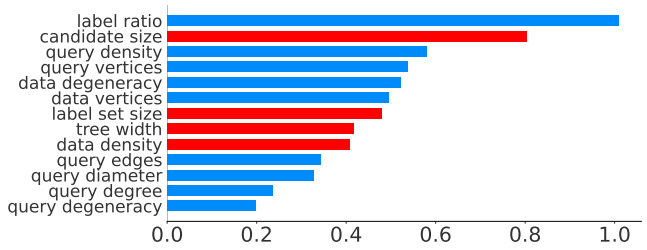


図 10: 平均 SHAP 値; 赤色は RECALGO での特徴量, 青色は追加した特徴量を表す.

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.

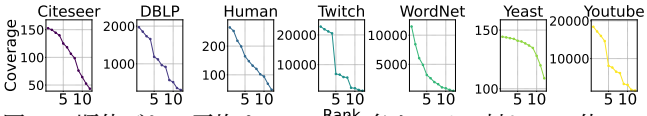


図 12: 順位ごとの平均カバレッジ; 各クエリに対して i 位のアルゴリズムを選択することで, ランク i におけるカバレッジを得ている.

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**

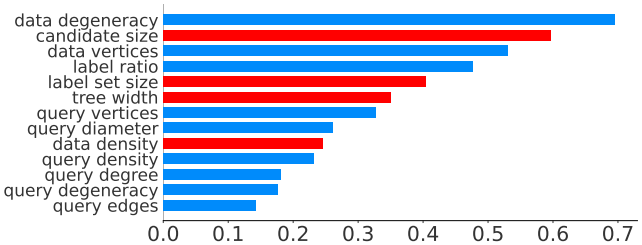


図 13: サブグラフカバレッジ: 平均 SHAP 値; 赤色は RECALGO での特徴量, 青色は追加した特徴量を表す.

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.