Efficient Partition-based Approaches for Diversified Top-k Subgraph Matching

Liuyi Chen
Hunan University
Changsha, China
liuyi.chen@hnu.edu.cn

Yuchen Hu
Hunan University
Changsha, China
shadowhyc@hnu.edu.cn

Zhengyi Yang
The University of New South Wales
Sydney, Australia
zhengyi.yang@unsw.edu.au

Xu Zhou
Hunan University
Changsha, China
zhxu@hnu.edu.cn

Wenjie Zhang
The University of New South Wales
Sydney, Australia
wenjie.zhang@unsw.edu.au

Kenli Li
Hunan University
Changsha, China
lkl@hnu.edu.cn

ABSTRACT

Subgraph matching is a core task in graph analytics, widely used in domains such as biology, finance, and social networks. Existing top-k diversified methods typically focus on maximizing vertex coverage, but often return results in the same region, limiting topological diversity. We propose the Distance-Diversified Top-k Subgraph Matching (DTkSM) problem, which selects k isomorphic matches with maximal pairwise topological distances to better capture global graph structure. To address its computational challenges, we introduce the Partition-based Distance Diversity (PDD) framework, which partitions the graph and retrieves diverse matches from distant regions. To enhance efficiency, we develop two optimizations: embedding-driven partition filtering and densest-based partition selection over a Partition Adjacency Graph. Experiments on 12 real world datasets show our approach achieves up to four orders of magnitude speedup over baselines, with 95% of results reaching 80% of optimal distance diversity and 100% coverage diversity.

PVLDB Reference Format:
Liuyi Chen, Yuchen Hu, Zhengyi Yang, Xu Zhou, Wenjie Zhang, and Kenli Li. Efficient Partition-based Approaches for Diversified Top-k Subgraph Matching. PVLDB, 19(4): 698 - 712, 2025.
doi:10.14778/3785297.3785310

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/Twilight-Shadow/DiversifiedTopkSubgraphMatching.

1 Introduction

Subgraph matching is one of fundamental tasks in graph analysis. Given a query graph and a data graph, it identifies all isomorphic subgraphs in the data graph. It serves as an effective means to extract valuable information from complex networks, supporting applications such as pattern recognition [30, 69], query optimization [13, 21, 51, 65, 67], and anomaly detection [7].

Proceedings of the VLDB Endowment, Vol. 19, No. 4 ISSN 2150-8097.
doi:10.14778/3785297.3785310

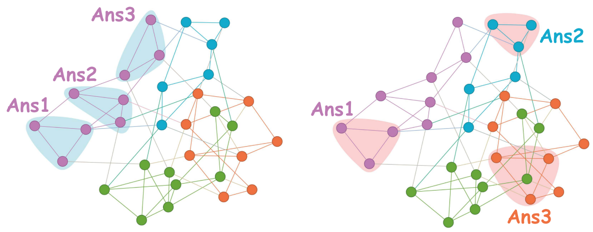
* Zhengyi Yang and Xu Zhou are the corresponding authors.

Numerous efforts [3–6, 11, 19, 33, 36, 38, 42, 45, 57, 82, 97, 99] have been devoted to addressing this problem in the literature. However, as the scale of data graphs grows, sometimes reaching billion-scale, two key issues arise: 1) Subgraph matching is NP-hard, making it computationally challenging to enumerate all possible matches. 2) The number of resulting subgraphs can be overwhelmingly numerous [28, 49, 94, 96, 102, 107], making them difficult to be analyzed effectively. A common practice in the literature is to enumerate the first k matches [17, 34, 62, 66, 83], where k is a user-defined constant that limits the maximum number of results. While this strategy reduces computational overhead, the returned matches are often redundant or overly similar. To address this, diversified top-k subgraph matching has been studied [3, 19, 57], which aims to return a set of k subgraphs that are not only isomorphic to the query but also diverse in structure or content, thereby enhancing interpretability and utility in downstream applications.

Motivation. Diversity metrics play a key role in top-k query processing by assessing the quality of returned results, aiming to reduce redundancy and enhance representativeness [23, 24]. This concept naturally extends to top-k subgraph matching. Existing works [57, 95] primarily aim to enhance result diversity by maximizing *vertex coverage*, which measures the number of distinct vertices covered by the selected set of subgraph matches. However, vertex coverage alone may fail to capture the true *structural dispersion* of subgraph matches, as it overlooks the *topological proximity* among the selected results in the data graph. That is, two subgraph matches may collectively cover many unique vertices, yet still be located in close proximity to each other, possibly within the same neighbourhood. In such cases, the result set may exhibit limited structural diversity, offering redundant or localized views.

Distance-based Diversity. To address this limitation, we introduce the problem of *Distance-Diversified Top-k Subgraph Matching* (DTkSM), which retrieves k subgraph matches that are not only isomorphic to the query graph but also maximally dispersed throughout the data graph. This formulation explicitly promotes diversity by maximizing the pairwise topological *distances* among the selected matches in the data graph, resulting in a more representative and interpretable answer set that better captures the broader structural landscape of the graph.

Example 1. *Figure 1 shows the top-k subgraph matching results on real-world dataset Human. When using vertex coverage, as shown*



(a) Coverage Based Diversity Matches in Human (b) Distance Based Diversity Matches in Human

Figure 1: Example of Top-3 diversified subgraph matches.

in Figures 1(a), the selected matches (highlighted in blue) are concentrated in specific local regions. Although these matches have little node overlap, they are clustered in the same local region of the graph, resulting in limited topological diversity and poor representation of the graph's global structure. In contrast, Figures 1(b) show the results using distance-based diversification. The selected matches (highlighted in red) are more widely distributed across the graph. This dispersion results in better diversity, allowing the results to more effectively represent the structural heterogeneity of the graph.

Applications. Beyond the motivating example, distance-based diversity has broad utility across domains.

<u>Biomedical Interaction Networks.</u> Subgraph matching facilitates disease pattern discovery in biomedical networks, such as protein–protein interaction graphs [1, 46, 55, 70, 71, 73]. In cancer research, a query graph may represent a metastasis-related structure, and its top-k matches can highlight potential metastatic pathways or target organs [1, 77]. However, coverage-based methods often return matches concentrated within a single region, whereas metastasis typically involves multiple, spatially distant organs [15, 76]. Distance-based diversity promotes spatially dispersed matches (e.g., lungs, liver), revealing independent metastatic routes and supporting prognosis, treatment planning [86, 91].

<u>Social Networks.</u> Subgraph matching helps identify patterns like influencer–follower motifs or tightly-knit triads [20, 40, 106]. Coverage-based methods often concentrate results in one community, limiting the detection of replicated behaviors such as misinformation campaigns [72, 103]. Distance-based diversification instead selects dispersed matches across distinct communities, enhancing the detection of distributed influence operations and supporting opinion analysis, anomaly detection, and moderation [18, 56, 63].

Challenges. While distance-based diversified top-k subgraph matching offers more informative and globally representative results, its computation remains highly non-trivial. Unlike coverage-based diversification, which operates under local constraints such as avoiding vertex overlap, distance-based diversification requires a global assessment of topological dispersion. Achieving optimal diversity demands complete enumeration of subgraph matches to evaluate all pairwise distances, thereby incurring substantial computational cost—especially on large graphs, where both subgraph enumeration and distance computation are resource-intensive. The primary challenges arise from two sources: *Subgraph Matching.* Subgraph matching is inherently NP-hard. This intrinsic complexity presents a fundamental bottleneck, especially when applied to large data graphs or complex query structures. *Distance Computation.* Ensuring structural diversity involves computing pairwise distances among a potentially large number of candidate matches. As the candidate space grows, distance evaluation becomes increasingly costly, particularly in large-scale graphs where such computations demand significant resources.

Research Question. This leads to the critical research question: <u>*how to efficiently find high-quality distance-diversified subgraph matching results while minimizing computational overhead?*</u>

Contributions. In this paper, we investigate distance-diversified top-k subgraph matching, which aims to find k isomorphic matches of a query graph in a data graph while maximizing pairwise topological distances. We propose the Partition-based Distance Diversity (PDD) framework, which partitions the data graph and retrieves matches from partitions that are topologically far apart. To enhance efficiency, PDD incorporates two optimizations: embedding-driven partition filtering, which leverages node embeddings to prioritize partitions likely to contain matches, and density-based partition selection, which formulates partition choice as a densest-subgraph problem on the Partition Adjacency Graph (PAG) to ensure that selected partitions are both structurally rich and well dispersed. The key contributions of this work are:

- **Novel Problem Formulation.** We define the *Distance-Diversified Top-k Subgraph Matching (DTkSM)* problem, which aims to select k isomorphic subgraph matches that maximize pairwise topological distances. We further prove that: 1) solutions to DTkSM also maximize vertex coverage as a natural consequence of structural dispersion, and 2) the DTkSM problem remains NP-hard, even when all subgraph matches are pre-computed.
- **Partition-Based Distance Diversity Framework.** To avoid exhaustive enumeration and costly distance computations, we propose the *Partition-based Distance Diversity (PDD)* framework. PDD partitions the data graph and strategically selects matches from distant regions to maximize topological diversity. This reduces the search space and computation while preserving the distance-diversity objective, and the partitioned structure naturally supports parallel matching for scalability.
- **Efficient Partition Filtering and Selection.** To enhance performance, we introduce two optimizations. *Embedding-driven partition filtering* leverages learned embeddings to identify partitions likely to contain valid matches, thereby pruning unpromising search spaces and reducing latency. *Densest-based partition selection* reformulates choosing k separated partitions as a densest subgraph problem on a *Partition Distance Graph (PDG).* This approach efficiently selects topologically dispersed partitions to ensure structural diversity.
- **Comprehensive Experimental Evaluation.** We conducted extensive experiments on both labeled and unlabeled graphs. The results demonstrate that our method achieves up to **four orders of magnitude** speedup over existing baselines and **one order of magnitude** on average. In terms of effectiveness, **95%** of our results achieve at least **80%** of the optimal distance diversity and consistently exhibit optimal coverage diversity. In contrast, only **3%** of cases from competing methods reach comparable quality.

2 PRELIMINARIES

In this section, we present the key definitions and propose the problem statement of this paper.

2.1 Key Concepts

We study an undirected and unweighted graph $G = (V, E)$, where V denotes the vertex set and $E \subseteq \{\{u, v\} \mid u, v \in V, u \neq v\}$ is the edge set. Each vertex $v \in V$ is associated with a categorical label $\ell(v)$. For a vertex v, the *degree*, denoted as $\deg(v)$, is defined as the number of edges incident to v. Two vertices u and v are said to be *adjacent* if $\{u, v\} \in E$, i.e., if they are connected by an edge.

Definition 1 (Shortest Path). *Given a graph G=(V, E), the shortest path between two vertices $u, v \in V$ is the path with the minimum total edge weight among all possible paths from u to v.*

Definition 2 (Subgraph Isomorphism). *Given a query graph $Q = (V_Q, E_Q, \ell_Q)$ and a data graph $G = (V_G, E_G, \ell_G)$, a subgraph $M \subseteq G$ is said to be* isomorphic *to Q if there exists an injective mapping $f : V_Q \rightarrow V_G$ such that $\ell_Q(u) = \ell_G(f(u))$ for all $u \in V_Q$, and $(f(u_1), f(u_2)) \in E_G$ for all $(u_1, u_2) \in E_Q$. The subgraph M, induced by the vertex set $f(V_Q)$, is referred to as a* match *of Q in G, and f is called a* subgraph isomorphism mapping.

Definition 3 (Subgraph Matching). *The goal of* subgraph matching *is to enumerate the complete set $\mathcal{M} = \{M_1, M_2, \ldots, M_t\}$ of subgraphs in G, where each M_i is isomorphic to Q. The set $\mathcal{M}$ is referred to as the* match set *of Q in G.*

Definition 4 (Diversified Top-k Subgraph matching). *Given a query graph Q, a data graph G, and an integer k, let $\mathcal{M}$ denote the match set of Q in G as defined in Definition 3. The* diversified top-k subgraph matching *problem aims to find a subset $\mathcal{R} \subseteq \mathcal{M}$ with $|\mathcal{R}| = k$ that maximizes a given diversity function $\mathcal{F}(\mathcal{R})$.*

Definition 5 (Coverage-based Diversity [95]). *Given a set of k subgraph matches $\mathcal{R} = \{M_1, M_2, \ldots, M_k\}$ in data graph G, the* coverage-based diversity function *is defined as*

$$\mathcal{F}_{\text{cov}}(\mathcal{R}) = \left| \bigcup_{i=1}^{k} V(M_i) \right|. \tag{1}$$

However, as illustrated in the introduction, the above definition fails to fully capture the essence of diversity. To address this limitation, we introduce a more comprehensive metric that emphasizes distance-based diversity among the results.

Definition 6 (Subgraph Distance). *Given two subgraphs M_i and M_j in a data graph G, their distance is defined as: $d(M_i, M_j) = \min_{u \in M_i,\, v \in M_j} \text{dist}_G(u, v)$, where $\text{dist}_G(u, v)$ denotes the shortest-path distance between nodes u and v in G.*

Definition 7 (Distance-based Diversity). *Given a set of k subgraph matches $\mathcal{R} = \{M_1, M_2, \ldots, M_k\}$ in a data graph G, the distance-based diversity is defined as: $\mathcal{F}_{\text{dis}}(\mathcal{R}) = \max_{\substack{i,j \\ i \neq j}} d(M_i, M_j)$*

Remark 1. *The proposed definition of distance-based diversity is theoretically grounded and consistent with established practices in top-k recommendation systems, where diversity is commonly promoted by maximizing the minimum pairwise distance among items. Similarly, our formulation adopts this principle by maximizing the minimum topological distance between matches, making the metric both intuitive and well-supported by prior theoretical frameworks.*

2.2 Problem Statement

Definition 8 (Distance-Diversified Top-k Subgraph Matching). *Given a query graph Q and a data graph G, the goal is to find a set of k subgraph matches such that each match is isomorphic to Q, and the result set maximizes the distance-based diversity objective defined in Definition 7.*

Remark 2. *The problem of DTkSM can be naturally decomposed into two stages: (1) enumerating all subgraph matches that are isomorphic to the query graph Q, and (2) selecting a subset of k matches that maximizes the distance-based diversity objective.*

Theorem 1. *Finding the top-k subgraph matches that maximize distance-based diversity is NP-Hard, even when all subgraph matches are pre-computed.*

Proof. Due to page limitations, the detailed proof is provided in the Appendix A [14]. □

3 The Proposed Method

In this section, we introduce the PDD framework that first selects the k most dispersed regions (graphs) via coarsen distance, and then performs subgraph matching in parallel within them.

3.1 Overview Framework

The new algorithmic framework is shown in Algorithm 1. The input graph G is divided into p disjoint partitions $\{G_1, \ldots, G_p\}$. To guide partition selection, a $p \times p$ distance matrix D for partitions is computed, where each entry $D_{i,j}$ represents the distance between partitions G_i and G_j (will be defined later). To ensure the diversity among the matches, the algorithm incrementally selects a subset $\mathcal{P}$ from the partition set. Starting from a random partition, each subsequent partition is selected to be the one which owns the maximum distance from the already selected ones, based on matrix D. This helps to spread the search across graph regions that are topologically far apart. Each selected partition $G_i \in \mathcal{P}$ is processed in parallel to search for occurrences of the query subgraph Q. If a match is found within a single partition, it is added to the match set $\mathcal{M}$. If the number of matched subgraphs is still less than k, an additional inter-partition matching phase is invoked, where the algorithm searches for matches that span multiple partitions in $\mathcal{P}$. Finally, the algorithm returns up to k-matched subgraphs that are expected to be both correct and topologically diverse.

Time Complexity Analysis: Excluding offline preprocessing (e.g., graph partitioning and distance computation), the online algorithm consists of two phases. In the first phase, selecting k dispersed partitions from n candidates requires $O(nk)$ time, as each round evaluates up to n neighbors. In the second phase, fine-grained subgraph matching is performed parallel on the k selected partitions. Letting T_{match} denote the total matching time, the overall online complexity is $O(nk + T_{\text{match}})$.

3.2 Partition-based Graph Preprocessing

Graph Partition. Partitioning the graph supports scalability by dividing a large graph into smaller, manageable subgraphs that

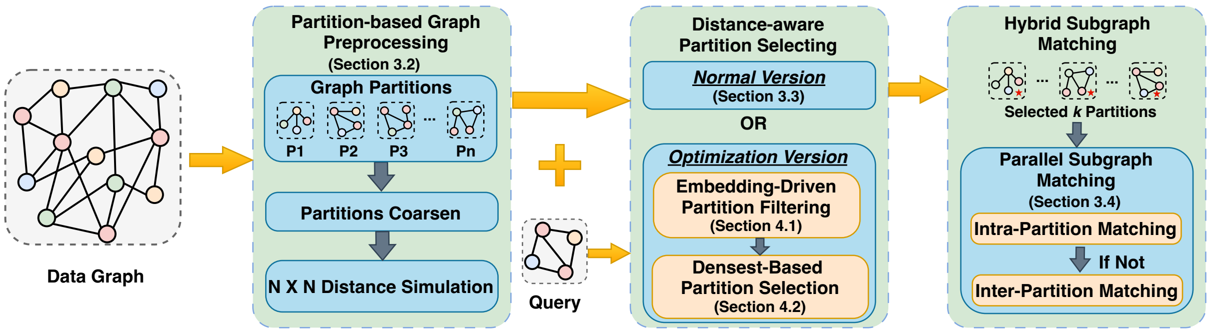


Figure 2: Overview Framework

Algorithm 1: The PDD Framework

Input: Data graph G, query graph Q, target size k
Output: Set $\mathcal{M}$ of k diverse subgraph matches

```
//Partition-based Graph Preprocessing
Partition G into {G_1, G_2, ..., G_n};
Compute n × n distance matrix D, where D_{i,j} is the distance in the
    Partition Adjacency Graph (PAG);
// Distance-aware Partition Selection
Initialize candidate list P ← ∅;
Randomly select a partition G_r and add to P;
while |P| < k do
    Identify G_j as the farthest valid neighbor from current P using
      D;
    P ← P ∪ {G_j};
// Hybrid Subgraph Matching
Initialize match set M ← ∅;
foreach G_i ∈ P in parallel do
    M ← HySM(G_i, Q) (Algorithm 2)
// Backtracking (if |M| < k);
if |M| < k then
    Let m =k−|M|;
    Select m most promising partitions not in P;
    Perform matching in these partitions to complete M;
return M
```

---

enable parallel processing and reduce memory overhead. It also facilitates efficient candidate filtering by localizing the search space. We adopt Distributed-NE [37], a parallel edge partitioning strategy that grows each partition from a seed vertex via greedy edge expansion. To maintain logical consistency, the scheme uses vertex replication: when an edge spans multiple partitions, its endpoints are duplicated across them. Thus, a vertex appearing in multiple partitions implicitly encodes a inter-partition connection.

Partition Distance. To improve retrieval diversity and minimize redundancy, we aim to select partitions that are distant from one another in the original graph. A straightforward approach is to define a distance metric between every pair of partitions and select those that are far apart. However, computing such distances directly on the original graph via shortest paths is computationally prohibitive for large-scale graphs.

To address this, we introduce a lightweight graph abstraction that captures essential inter-partition connectivity while significantly reducing distance computational complexity. Specifically, we construct an auxiliary graph where each vertex represents a partition and edges indicate adjacency based on shared (replicated) vertex. This allows inter-partition distances to be approximated efficiently using standard graph traversal techniques.

To formally define adjacency between partitions, we first state a lemma from the node-replication model in Distributed-NE:

Lemma 1. *If two partitions A and B are adjacent in the original graph G, then $V_A \cap V_B \neq \emptyset$.*

Proof. Suppose A and B are adjacent in G, so there exists an edge $(u, v) \in E$ with $u \in A$, $v \in B$. In edge partition methods, preserving this inter-partition edge requires replicating at least one of its endpoints into the other partition. Hence, either $u \in B$ or $v \in A$, which implies that $V_A \cap V_B \neq \emptyset$. □

Using this result, we define an auxiliary graph to model partition-level relationships:

Definition 9 (Partition Adjacency Graph). *Let $G = (V, E)$ be an undirected graph partitioned into $\mathcal{P} = \{P_1, \ldots, P_n\}$. The* Partition Adjacency Graph (PAG) *is a graph $G_H = (\mathcal{P}, E_H)$, where $E_H = \{(P_i, P_j) \mid P_i \cap P_j \neq \emptyset\}$.*

Definition 10 (Distance in Partition Adjacency Graph). *Let $G_H = (\mathcal{P}, E_H)$ be the PAG as defined above. The distance $d_H(P_i, P_j)$ between two partitions P_i and P_j is defined as the shortest path between P_i and P_j in G_H.*

This abstraction offers two key advantages. First, it drastically simplifies distance computation by avoiding costly traversal on the full graph. Second, it enables the identification of structurally distant partitions, guiding the system to favor less-connected regions of the graph. These properties jointly improve both retrieval efficiency and the diversity of the selected results. At the end of this phase, pairwise structural distances between all partitions are available to support the subsequent selection process.

Vertex Replication. When a graph employs edge-partitioned, some edges inevitably span multiple partitions. To preserve connectivity, the endpoints of these inter-partition edges are replicated into the corresponding partitions. This mechanism, known as vertex replication, is widely adopted in existing partitioning methods [9, 37, 58, 60, 101]. In the following, we add a discussion on the vertex replication impact. We guarantee that replication does not affect the correctness of our results: (1) no results are missed, and (2) no duplicates remain. First, although results spanning multiple partitions could be lost, we address this by designing an inter-partition matching procedure (see Section 3.4), which ensures completeness. Second, replication may create duplicate matches, but such cases are rare. This mainly occurs in two scenarios: (i) a partition contains only replicated vertices without any original part of the result, which is highly unlikely under balanced partitioning; or (ii) two neighboring partitions independently discover the same partial match. In the latter case, we apply a deduplication strategy so that only one copy is retained. Furthermore, we demonstrate in Exp 2 (Table 3) that vertex replication has no impact on either effectiveness or efficiency, as verified by varying replication vertices and different replication ratios.

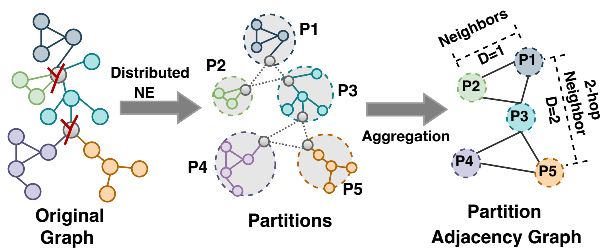


Figure 3: Illustration of Partitions Coarsen

Example 2. *As shown in Figure 3, the original graph is partitioned into five regions (P1–P5), with red crosses marking cut points. Gray vertices at cuts are replicated across relevant partitions, indicated by dashed lines. Each partition is then abstracted as a supernode to coarsen the graph. Edges connect supernodes sharing replicated vertices, forming the Partition Adjacency Graph (PAG). In the PAG, distance is measured by shortest-path: adjacent P1 and P2 have distance 1, while P1 and P5, connected via P3, have distance 2.*

3.3 Distance-aware Partition Selection

Our next objective is to select k structurally dispersed partitions from the partition set P. To promote distance diversity, we aim to select k partitions from the set P such that the selected partitions are well-separated in the graph. Specifically, we use the partition-level distance $d_H(P_i, P_j)$ defined in Definition 10 to measure the distance between two partitions. Our objective is to select a subset $S \subseteq \mathcal{P}$ with $|S| = k$, in which the minimum pairwise distance among all partition pairs is maximized. This ensures that even the closest pair in S is still far apart, thus encouraging the overall dispersion of the selected regions. Although the pairwise distances between partitions can be efficiently computed using a preconstructed distance matrix, selecting the optimal subset of k partitions remains a combinatorial optimization problem. The search space consists of all possible k-sized subsets of $\mathcal{P}$, totaling $\binom{|\mathcal{P}|}{k}$ combinations. As a result, exhaustive search becomes computationally infeasible on large-scale graphs. For the solution, we employ a greedy dispersion-aware strategy to efficiently approximate the optimal solution. The procedure begins by randomly selecting an initial partition. In each step, we add the partition that has the maximum minimum distance to all partitions currently in the selected set. This process relies on a precomputed distance matrix and continues until k partitions are selected. The algorithm has a time complexity of $O(kn)$, is simple to implement, and scales well to large graphs while producing well-dispersed partition sets.

3.4 Hybrid Subgraph Matching

After selecting the top-k partitions, we perform fine-grained subgraph matching within them. Since partitioning phase may sever inter-partition structures, matches spanning multiple partitions may be missed. For this challenge, our algorithm use a two-phase matching strategy. In the first phase, a basic subgraph matching algorithm is applied independently within each partition to identify local matches. If no satisfactory result is found at this stage, the process advances to the second phase, in which the search is expanded across partition boundaries to recover connectivity and discover potential inter-partition matches.

The subgraph fine-matching procedure is detailed in Algorithm 2 and consists of two stages designed to ensure both completeness and result diversity. In the first stage, each selected partition G_i is evaluated independently and in parallel using a hybrid subgraph matching algorithm. We initialize candidate sets for each query vertex based on labels and local constraints, then refine them using bipartite graphs with semi-perfect matching checks [38] to prune invalid mappings. A cost model [38] determines the join order, and LFTJ [84] is applied to compute synchronized intersections, enabling depth-first enumeration without large intermediates.

If no match is found within a partition, the algorithm enters the second stage to search results across partitions. In the second stage, BFS pattern trees are built, with each query vertex as root, to define the propagation order of constraints and guide inter-partition expansion. BFS cascading filtering is then applied from boundary candidate sets, retaining only inter-partition candidates that satisfy inexpensive checks (e.g., label consistency, degree bounds, adjacency summaries) while recording boundary information for stitching [5]. The filtered frontiers are refined by DFS matching, where query vertices are incrementally mapped with forward checking and edge constraints. When traversing partitions, only essential states are preserved via stored boundary contexts to reduce overhead while ensuring correctness. Backtracking is invoked upon constraint violations to guarantee completeness.

Theorem 2. *By maximizing the distance-based diversity objective function $\mathcal{F}_{dist}(\mathcal{R})$, the framework simultaneously maximizes the coverage-based diversity $\mathcal{F}_{cov}(\mathcal{R})$.*

Proof. Let $\mathcal{R} = \{c_1, c_2, \ldots, c_k\}$ be the set of selected matches, where each c_i is chosen from a distinct partition $P_i \in \mathcal{P}$. By the distance-based diversity criterion, the selected partitions are pairwise non-adjacent in the Partition Adjacency Graph (PAG). Hence, for all $i \neq j$, we have $P_i \cap P_j = \emptyset$, which implies $V(c_i) \cap V(c_j) = \emptyset$. It follows that the total coverage of the match set is equal to the sum of the sizes of the individual matches: $\left|\bigcup_{i=1}^{k} V(c_i)\right| = \sum_{i=1}^{k} |V(c_i)|$. Moreover, by construction, each c_i is selected to maximize $|V(c_i)|$ within its corresponding partition P_i. Since the vertex sets $V(c_i)$ are disjoint and locally optimal within disjoint regions, no alternative selection of k disjoint subgraphs can yield a higher total coverage. Thus, the framework also maximizes $\mathcal{F}_{\text{cov}}(\mathcal{R})$. □

Remark 3. *In our framework, the graph is partitioned with vertex replication to preserve inter-partition connectivity, so adjacent partitions share vertex cuts while non-adjacent ones do not (Lemma 3.1). Since PDD and PDD+ prioritize distance, typically non-adjacent partitions, the resulting matches contain no overlapping vertices. By Definition 2.5, coverage is defined as the union of matched vertex sets, which is maximized when matches are disjoint. Hence, ensuring distance-based diversity inherently guarantees maximum coverage, as further supported by our experimental results.*

4 Optimizations

In this section, we propose an optimized framework that improves both the effectiveness of selection and computational efficiency.

As shown in Algorithm 3, the first stage follows Algorithm 1: Partition-based Graph Preprocessing phase. Partitions and their distance information are retained for downstream processing. The second stage introduces the use of representation learning from machine learning, where feature vectors are extracted for each subgraph. By embedding these vectors in an ordered space, we can efficiently determine the relationship between the query graph and each partitioned subgraph, estimating the probability of their existence. In the third stage, a Partition Distance Graph (PDG) is constructed by integrating both existence probabilities and inter-partition distances, enabling the discovery of a densest subgraph of size k. The detailed procedure is presented in Algorithm 4. Vertexes of achieved densest subgraph are the selected partitions. Finally, in the last stage, traditional algorithms are employed to perform parallelized subgraph matching within the top-k ranked partitions. This approach optimizes the matching process by focusing computational resources on the most promising partitions.

Time Complexity Analysis: Excluding offline preprocessing steps, such as graph partitioning and distance-matrix computation, our online algorithm consists of two phases. First, feature extraction and model inference over all n partitions execute in parallel on n workers in $O(1)$ wall-clock time. Second, we launch k threads for "select next" and fine-grained subgraph matching: main thread scans m predicted candidates against the k already-chosen partitions in $O(mk)$ and the other k threads performs matching in $O(T_{\text{match}})$ at the same time. Since main thread and all k threads run concurrently, this phase completes in $O\big(\max\{m\,k,\ T_{\text{match}}\}\big)$, which therefore also bounds the overall parallel time complexity.

Algorithm 2: Hybrid Subgraph Matching (HySM)

Input: Query graph Q, One of Selected partition P_i
Output: One matched subgraph in P_i or `NoResult`

```
// Intra-partition matching
```
Select and prune candidates with neighborhood information [38];
Reduce search space using semi-perfect matching [38];
Optimize search order using a cost model [38];
if *LFTJ [84] finds a match* **then**
    return *matched subgraph*
```
// Inter-partition matching
```
Generate BFS tree patterns from each vertex in Q;
Identify cut-off candidates in P_i for each vertex in Q;
foreach *query vertex* ***in parallel*** **do**
    Perform BFS filtering to prune candidates[5] ;
    Use DFS to find valid matchings from filtered candidates;
    if *valid match found* **then**
        return *matched subgraph*
return `NoResult`

Algorithm 3: The PDD Plus Framework

Input: Data graph G, query graph Q, target size k
Output: Set $\mathcal{M}$ of k diverse subgraph matches
```
// Offline
```
Execute *Algorithm 1: Partition-based Graph Preprocessing*;
Achieve $\{G_1, \ldots, G_n\}$& Distance matrix $D \in \mathbb{R}^{n \times n}$;
```
// Optimized Partition Selection
```
forall *partition* G_i ***in parallel*** **do**
    $\mathbf{x}_i \leftarrow$ ExtractFeatures(Q, G_i);
    $\hat{y}_i \leftarrow$ Model.predict$(\mathbf{x}_i)$;
$\mathcal{P}_1 \leftarrow \{\, G_i \mid \hat{y}_i = 1 \}$;
Construct graph $G_H = (\mathcal{P}_1, E_H, w_H)$ with
    $E_H = \{(G_i, G_j) \mid G_i, G_j \in \mathcal{P}_1\}$, $w_H(i,j) = D_{i,j}$;
    $\mathcal{M} \leftarrow$ DekPS(H, k) (Algorithm 4);
return $\mathcal{M}$

Definition 11 (Approximation Ratio). *For distance-based diversity, we define the approximation ratio as $\rho = \frac{D_{alg}}{D^*}$, where D_{alg} is the distance-based diversity achieved by our algorithm, and D^* is the optimal distance-based diversity.*

Theorem 3. *Let G_i and G_j be two partitions in the Partition Adjacency Graph (PAG) such that the shortest path between them is $h = d_H(G_i, G_j)$. Under the specific assumptions, the approximation ratio ρ is bounded by: $\rho \geq \frac{h-1}{h+1}$*

Proof. Due to page limitations, the detailed proof is provided in the Appendix A [14]. □

4.1 Embedding-driven Partition Filtering

In this phase, we leverage the learned embeddings of both the query and the partitions, which encode structural and semantic features in a shared vector space. By comparing the query embedding with each partition embedding, we estimate the likelihood that a given partition contains potential matches. This similarity-based estimation allows for efficient pre-filtering of the search space, significantly reducing the number of partitions that need to be explored during the fine-grained matching phase.

Initial Embedding. A key challenge in training arises from the scale mismatch between small query graphs and large partitions, which limits embedding comparability. Reducing partition size alleviates this but causes fragmentation and high overhead. To address this, we sample query-sized subgraphs from the data graph and pair them with queries as training instances, using higher sampling frequency on large graphs for coverage. We then employ GNNs to embed both queries and subgraphs into a shared space, capturing semantic and structural features for efficient comparison and alignment without costly structural matching.

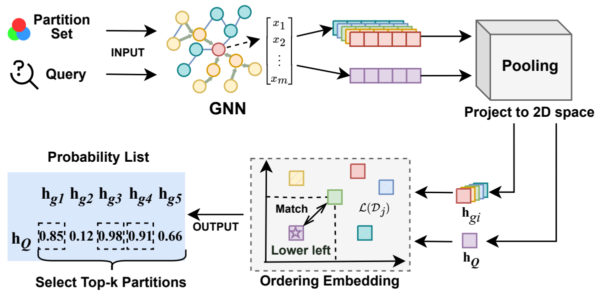


Figure 4: Neural Partition Filtering: Select k likelihood Partitions(k=2).

Ordering Embedding. Graph data is inherently complex, and direct operations on the raw structure are often computationally expensive. Traditional Graph Neural Networks (GNNs) focus on messaging local neighborhoods information, but less effective at capturing global relationships—particularly subgraph inclusion. Against this backdrop, Order Embedding, proposed by Vendrov et al.[85], emerges as a suitable representation learning method which encodes partial order relations by enforcing component-wise inequalities between embedding vectors. **A prerequisite for applying order embedding is the explicit definition of partial order relations among the elements.**

Property 1 (Partial Order). *Let S be a set. A binary relation $\leq$ on S is called a* partial order *if it satisfies the following three properties for all $a, b, c \in S$: Reflexivity: $a \leq a$; Antisymmetry: if $a \leq b$ and $b \leq a$, then $a = b$; Transitivity: if $a \leq b$ and $b \leq c$, then $a \leq c$.*

Theorem 4. *The subgraph matching problem naturally forms a partial order.*

Proof. We verify the three conditions of partial order: Reflexivity: Every graph is trivially a subgraph of itself; Antisymmetry: If $G_1 \subseteq G_2$ and $G_2 \subseteq G_1$, then $G_1 = G_2$ (i.e., the graphs are isomorphic). Transitivity: If $G_1 \subseteq G_2$ and $G_2 \subseteq G_3$, then $G_1 \subseteq G_3$. Therefore, the subgraph relation satisfies reflexivity, antisymmetry, and transitivity. □

Using this embedding, each pair (a subgraph and a query graph) is embedded into a continuous vector space, where the geometric relationships between the embeddings represent their partial order. Specifically, if the query graph is isomorphic to G_2, the embedding of the query graph is positioned geometrically "below and to the left" of the embedding of G_2 in the vector space. This spatial configuration is essential for capturing the properties of partial orders in the embedding space.

Training Process. To learn embeddings that preserve partial order, we design a loss function that enforces component-wise inequalities between embedding vectors. For any graph pair (G_1, G_2) where $G_1 \leq G_2$ (i.e., G_1 is a subgraph of G_2), their embeddings should satisfy: $\forall i, E_{G_1}[i] \leq E_{G_2}[i]$. Violations of this inequality are penalized through a loss function.

Positive Pairs. For subgraph-supergraph pairs (G_1, G_2), we apply a hinge-style loss: $L_{\text{pos}} = \sum_i \max(0,\ E_{G_1}[i] - E_{G_2}[i])$.

Negative Pairs. For unrelated pairs (G_1, G_3) where $G_1 \nleq G_3$, we enforce a margin to prevent false ordering:$L_{\text{neg}} = \sum_i \max(0,\ \text{margin} + E_{G_3}[i] - E_{G_1}[i])$.

The overall training loss is defined as: $L = \lambda_{\text{pos}} L_{\text{pos}} + \lambda_{\text{neg}} L_{\text{neg}}$, where λ_{pos} and λ_{neg} balance the two components. Minimizing this loss across labeled graph pairs encourages the embedding space to reflect true subgraph relations.

Inference and Filtering. After training, the model is used to predict whether a query graph Q is a subgraph of a candidate graph G. Given their embeddings E_Q and E_G, the model determines a positive subgraph relation (i.e., $Q \leq G$) if $\forall i, E_Q[i] \leq E_G[i]$.

At the beginning, the data graph is partitioned to identify regions likely to contain query matches. Rather than evaluating each partition sequentially, we perform predictions in parallel across all partitions. This parallel inference strategy significantly reduces latency and accelerates candidate region identification during the online phase, with an accuracy of around 75%. Based on the prediction results, we treat partitions prediction labelled as 1 as those more likely to contain matching subgraphs. In the subsequent step, we select k relatively dispersed partitions from this high-likelihood subset to ensure high-quality matching. This pre-selection strategy increases the likelihood that the final partitions contain valid results, thereby reducing the need for backtracking and improving overall efficiency.

4.2 Densest-based Partition Selection

Based on the previously defined notion of diversity, our objective is to select k partitions that are maximally distant from each other. Greedy-based approach to this problem often suffers from a lack of global optimality and strong sensitivity to the initial selection, resulting in suboptimal and unstable outcomes.

Definition 12 (Densest Subgraph). *A subgraph G_D of graph G is called a densest subgraph if maximizes the density $\frac{|E_D|}{|V_D|}$.*

Definition 13 (Partition Distance Graph). *Let G be partitioned into n disjoint blocks $\mathcal{P} = \{P_1, \ldots, P_n\}$. For each partition P_i is a vertex v_i in the* partition distance graph *(PDG). The PDG is then a weighted, undirected graph $G_{\text{P}} = (V, E, w)$, where $V = \{v_1, v_2, \ldots, v_n\}$. Two vertices v_i and v_j are connected by an edge in E if and only if their corresponding partitions P_i and P_j are non-adjacent in the original graph G. For such a pair, the edge weight is $w(v_i, v_j) = \text{dist}_{\text{PAG}}(P_i, P_j) - 1$, where $\text{dist}_{\text{PAG}}(P_i, P_j)$ denotes the shortest path between P_i and P_j in the partition adjacency graph (PAG).*

Definition 14 (Degree in PDG). *The degree of a vertex v_i in the PDG is defined as the number of vertices to which it is connected, i.e., the number of partitions P_j that are non-adjacent to its corresponding partition P_i in the original graph.*

To overcome these limitations, we reformulate the k dispersed partition selection process as a densest subgraph discovery problem over PAG.

Lemma 2. *Selecting a subset of k partitions that maximizes the minimum pairwise distance is equivalent to finding a k-size densest subgraph in the Partition Distance Graph (PDG) whose minimum edge weight is maximized.*

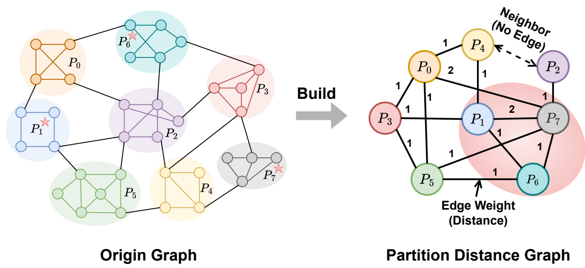


Figure 5: Dispersed k Partitions Selection

Proof. We show that the original problem and the max–min edge-weight subgraph problem on the PDG share the same feasible space and objective.
Feasibility. In the original problem, a valid solution requires selecting k partitions such that no two are adjacent in the data graph. In the PDG, vertices represent partitions, and edges exist only between non-adjacent pairs. Thus, any feasible subset of k partitions corresponds to a k-vertex subgraph in the PDG.
Objective. Let $S \subseteq \mathcal{P}$ be a feasible selection. The original objective is to maximize $\mathcal{D}(S) = \min_{P_i, P_j \in S} d(P_i, P_j)$. In the PDG, edge weights are defined as $w(P_i, P_j) = d(P_i, P_j)$, so the objective becomes $\mathcal{D}(S) = \min_{(P_i, P_j) \in E[S]} w(P_i, P_j)$. Hence, the original problem reduces to selecting a k-vertex subgraph of the PDG that maximizes the minimum edge weight. □

Example 3. *Figure 5 illustrates the process. First, the original graph is partitioned, and both adjacency and pairwise partition distances are recorded. In the filtering stage, partitions predicted likely to contain query matches are retained as candidates. To enforce structural constraints and support efficient selection, we construct the Partition Distance Graph (PDG) as defined in Definition 13. We then select a k-size densest subgraph in the PDG that maximizes minimum edge weight. In Figure 5, the highlighted red region shows the chosen partitions p_1, p_7, and p_6, representing the $k = 3$ most dispersed partitions.*

Compared to greedy methods, our approach captures global structure via a densest subgraph formulation over the PDG, enforces disjointness and dispersion through encoded constraints, and yields an interpretable k size high-weight densest subgraph that reflects both coverage and distance diversity.
Customized Densest subgraph discovery. While classical densest subgraph algorithms aim to identify dense regions by exploring a variety of subgraphs with varying sizes, they are not directly applicable to our setting. We seek a single densest subgraph of fixed size k that optimizes a specific objective—maximizing the each edge weight among selected vertices. Moreover, most existing algorithms are not designed to accommodate such task-specific goals. To address these challenges, we design a task-specific heuristic that draws on established principles from densest subgraph discovery while accommodating our unique constraints.

As shown in Algorithm 4, DEkPS heuristically selects k partitions from the PDG to form a high-weight, structurally disjoint subgraph. The strategy is inspired by densest subgraph discovery, aiming to incrementally construct a size-k subgraph with maximal edge weight under non-adjacency constraints. The algorithm starts from the partition with the highest degree and adds it to the candidate set L. At each iteration, it identifies candidate partitions that are non-adjacent to all members of L and selects the one that contributes the highest edge weight to L, thus approximating a dense, dispersed subgraph. Ties are resolved by degree. Each selected partition undergoes subgraph matching via the hybrid matcher HySM (Algorithm 2), which is executed in parallel across partitions. The process continues until k valid matches are found or no further candidates remain. Through a peeling-style expansion guided by edge weights, DEkPS balances the goals of structural density and global dispersion in a unified selection framework.

Algorithm 4: Densest-based k Partition Selection (DEkPS)

Input: PDG $G_P = (V, E, w)$, query graph Q, target size k, the candidate partition set L
Output: Set $\mathcal{M}$ of up to k valid matches

Initialization:
$L \leftarrow \emptyset$; $\mathcal{M} \leftarrow \emptyset$;
$p^* \leftarrow \arg\max_{v \in V} \deg(v)$;
$L \leftarrow L \cup \{p^*\}$;
$\mathcal{M}_{p^*} \leftarrow \mathsf{HySM}(Q, p^*)$ (Algorithm 2);
if $\mathcal{M}_{p^*} \neq \emptyset$ **then**
  $\mathcal{M} \leftarrow \mathcal{M} \cup \mathcal{M}_{p^*}$;
$N \leftarrow \bigcap_{v \in L} Neighbor(v) \setminus L$;
while $|\mathcal{M}| < k$ ***and*** $N \neq \emptyset$ **do**
  $p^* \leftarrow \arg\max_{c \in N} (\max_{l \in L} w(c, l))$;
  if *multiple candidates* **then**
    choose p^* with highest degree among candidates;
  $L \leftarrow L \cup \{p^*\}$;
  $\mathcal{M}_{p^*} \leftarrow \mathsf{HySM}(Q, p^*)$ **in parallel** (Algorithm 2);
  if $\mathcal{M}_{p^*} \neq \emptyset$ **then**
    $\mathcal{M} \leftarrow \mathcal{M} \cup \mathcal{M}_{p^*}$;
  $N \leftarrow \bigcap_{v \in L} Neighbor(v) \setminus L$;
return $\mathcal{M}$

5 Experiments

In this section, we present the preprocessing cost and the performance of our algorithm in terms of efficiency, effectiveness, and scalability.
Environment. Experiments are conducted on a machine equipped with an Intel(R) Xeon(R) Gold 5218R CPU @ 2.10GHz, 256GB main memory, and a NVIDIA Tesla T4 GPU. We use PyTorch and C++ for the algorithm implementation.
Datasets. We evaluate our approach using real-world and synthetic graph datasets. Detailed information on all datasets is provided in Table 1. *Real-World Data Graph.* We use 10 real-world graph datasets from prior studies [104]. *Synthetic Data Graphs.* Existing real-world datasets are relatively small to evaluate scalability, we follow prior work [35, 41, 49, 61, 89, 105] and use LDBC [2] to generate synthetic datasets. The synthetic dataset, denoted as DGx (x is the scale factor), includes integer node labels from 0 to 10.

Table 1: Statistics of Datasets (Dataset Generated Labels).

Datasets	Name	$\|V\|$	$\|E\|$	$\|\Sigma\|$	Degree	Type
Yeast	*ye*	3,112	12,519	71	8.0	Protein
Human	*hu*	4,674	86,282	44	36.9	Protein
HPRD	*hp*	9,460	34,998	307	7.4	Protein
FreeBase15k	*fb*	14,951	260,184	1	42.7	Lexical
Wordnet18	*wm*	40,943	75,770	4	3.7	Lexical
WordNet	*wn*	76,853	120,399	5	3.1	Lexical
DBLP	*db*	317,080	1,049,866	15	4.9	Social
DBpedia	*dp*	343,794	1,371,562	1233	8.0	Wiki
Youtube	*yt*	1,134,890	2,987,624	24	5.2	Social
US Patents	*up*	3,774,768	16,518,947	25	8.8	Citation
DG 10	*d10*	29,987,834	176,481,399	11	11.6	Synthetic
DG 60	*d60*	187,108,072	1,246,659,788	11	13.2	Synthetic

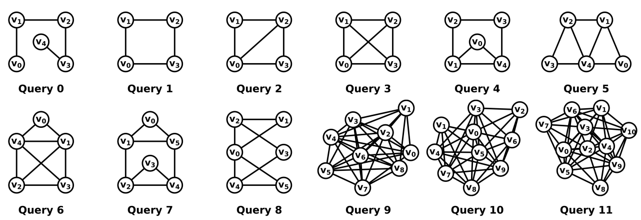


Figure 6: Examples of different kinds of queries.

Query. Traditional random-walk methods often yield overly simple query graphs. To ensure controlled and diverse evaluation, we categorize queries by topological complexity: *simple* (paths, cycles), *common* (combinations with shared vertices/edges), and *complex* (dense patterns such as quasi-cliques). Queries are generated using random walks, frequent subgraph mining, and densest subgraph discovery. Each set contains 1,000 queries with fewer than 20 vertices. Representative patterns are shown in Figure 6.

Baselines. We evaluate 11 algorithms: our proposed methods *PDD* and *PDD+*, along with 9 baselines. DSQL [95] is a diversified top-k subgraph matching algorithm, while PTAB [94] is a top-k subgraph matching algorithm. GuP [4], CFL [6], GQL [38], DAF [35], GQLfs [78], RapidMatch (RM) [79], and VEQ [44] are subgraph matching algorithms, among which GuP and DAF support parallel execution.

5.1 Initial Partition Evaluation

To give a comprehensive view of the offline phase, we present detailed partitioning information such as partition costs, and conduct a robustness analysis to examine the impact of different partitioning schemes on efficiency and result quality.

Exp1 - Preprocessing Cost. We adopt a default partition size in the range of 1000–2000 vertices, under which we report the corresponding preprocessing time and memory costs across all datasets in the Table 2. On most datasets, the partitioning time is only a few seconds with memory costs in the MB scale. For million-scale graphs, the overhead remains modest. Although the partitioning time becomes relatively high on billion-scale datasets, it is a one-time offline process and therefore acceptable. The choice of partition sizes in the range of 1000–2000 vertices is motivated by the favourable trade-off it offers between efficiency and result quality, and the experimental results are shown in Figure 7. Smaller partitions lead to excessive fragmentation and costly inter-partition searches, whereas larger ones approximate the entire graph and thereby reduce parallelism.

Exp 2 - Robustness under Partitioning Schemes. We evaluated different partitioning methods to analyze the robustness of our algorithm in terms of execution time and result quality. Specifically, the original graph was partitioned using two edge partitioning strategies (SHEEP [58] and Distributed-NE [37]) as well as a community detection method (Leiden) [80], and the resulting partitions were then used as inputs to our framework. As shown in Table 3, although the edge-based methods yield different replication ratios, our framework produces almost identical outcomes in both efficiency and effectiveness. The maximum runtime difference between them is only 0.17 ms, while the distance diversity of the results differs by at most 0.12, indicating that our algorithm is robust and largely unaffected by the choice of edge-based partitioning scheme. In contrast, the community detection method performs worse in both matching time and distance diversity compared with edge-based approaches. For example, on the *DBLP* dataset, it is about 20 times slower. This is because, unlike edge-based partitioning, community detection does not replicate endpoints: each vertex belongs to exactly one community (partition). As a result, matches that involve boundary vertices may be missed, leading to excessive inter-partition searches and degraded efficiency.

5.2 Efficiency Evaluation

Using the default k=30, we compared the execution times of our algorithm with baselines on real and synthetic datasets, evaluating both parallel and single-thread efficiency.

Exp3 - Overall Efficiency Comparison (Real-World Data). As shown in Figure 10, we evaluate PDD+ and baselines across all queries and real-world datasets. For clarity, we present results on three representative queries: one simple query, one medium-complexity query, and one near-clique query. The remaining experimental results are provided in the Appendix B[14]. For simple queries (Query 2), PDD and PDD+ generally achieve lower runtimes across most datasets, with clear advantages on large graphs such as *yt* and *up*. As dataset size grows, baselines (e.g., CFL, GuP, PTAB, VEQ) often escalate to 10^5ms, while PDD and PDD+ remain efficient. PDD+ also consistently outperforms DSQL(another diversified top-k subgraph matching algorithm), reducing latency by about 70% and achieving up to three orders of magnitude speed-up on sparse graphs (e.g., *wm*, *wn*), demonstrating robustness across data distributions.

On smaller datasets (*ye*–*hp*), some classical algorithms (e.g., CFL, GQL, DAF, RM) may run faster since the partition graph is similar in size to the original graph, making their processing time comparable to our intra-partition matching. However, these methods provide very poor diversity (see Section 5.2), whereas PDD+ again surpasses them on larger datasets. For moderately complex queries (Query 6), PDD+ achieves over an order-of-magnitude speed-up compared to DSQL, and up to four orders on sparse datasets such as *wm*, highlighting its robustness as query complexity increases. While classical methods may occasionally run faster on small graphs, PDD+ consistently outperforms them on larger datasets. For near-clique queries (Query 9), PDD+ still achieves two to three orders

Table 2: Preprocessing cost on different datasets

Dataset	Yeast	HPRD	Human	FB-15K	WordNet18	WordNet	DBLP	DBpedia	YouTube	US-Patents	YAGO	DG10	DG60
Graph Size(MB)	0.11	0.31	0.74	2.47	0.83	1.32	13.29	17.89	36.93	249.54	2590.72	3001.92	23490.56
Partition Time(s)	0.06	0.08	0.10	0.19	0.12	0.18	0.59	1.05	2.52	19.32	167.43	148.86	1078.82
Partition Memory(MB)	8.11	8.52	8.76	14.25	9.57	12.59	62.95	77.25	174.57	883.00	11622.00	9044.00	63693.00

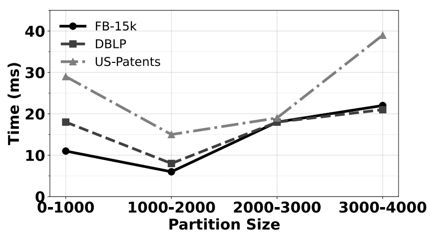


Figure 7: Partition Size Evaluation.

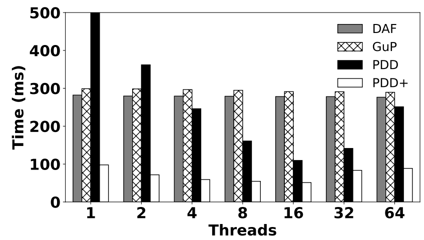


Figure 8: Parallel Comparison (YouTube).

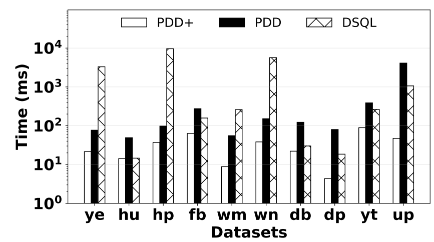


Figure 9: Sequential Comparison.

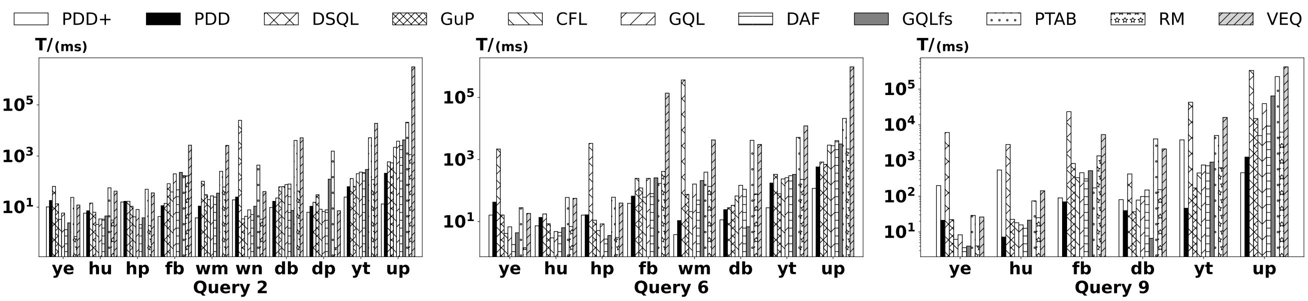


Figure 10: Runtime of Different Query on Different Real-World Datasets (k = 30)

Table 3: Comparison of different partition schemes.

Dataset	Replication Ratio			Execution Time(ms)			Result Distance		
	NE	SHEEP	Leiden	NE	SHEEP	Leiden	NE	SHEEP	Leiden
HPRD	2.57	2.12	1.00	1.65	1.48	5.80	2.46	2.56	2.06
DBLP	1.43	1.88	1.00	1.64	1.81	21.01	5.95	5.93	5.61
YouTube	1.77	2.21	1.00	25.77	25.63	31.25	4.58	4.60	3.71

of magnitude speed-up over DSQL. Compared with subgraph enumeration, PDD+ is not always the fastest on small graphs, but consistently shows clear advantages on medium and large datasets.

Exp4 - Overall Efficiency Comparison (Synthetic Data).
We also evaluate all queries on two synthetic datasets, *DG10* and *DG60*. Due to the extreme scale of these datasets, DSQL sometimes doesn't run properly due to its extensive record of temporary data for pruning during the process. As shown in Figure 11, PDD+ and PDD are generally among the fastest methods. On the large-scale dataset DG60, PDD+ is typically one to two orders of magnitude faster than most baselines, and in clique queries (q9–q11) it can maintain advantages exceeding three orders of magnitude. DSQL is clearly slower than PDD+. On DG10 the gap is about one order of magnitude, while on DG60 with more complex queries (q6–q11),

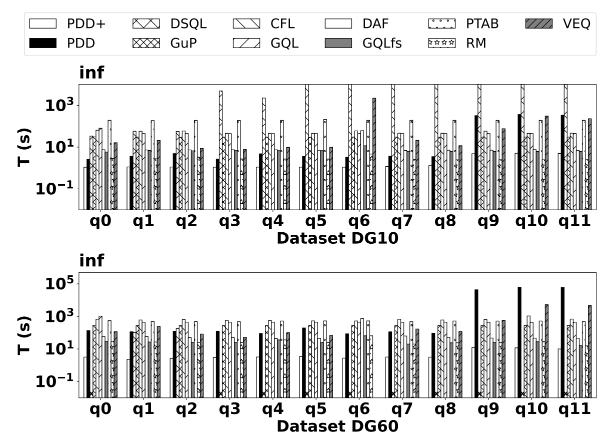


Figure 11: Runtime of All Queries on Syn-Datasets (k = 30)

the difference further enlarges to two or even three orders of magnitude. Subgragh enumeration methods (CFL, GuP, GQL, DAF, etc.) are occasionally comparable to, or slightly faster than, PDD+ on small datasets and simple queries (q0–q2). However, as query complexity grows, their runtime increases sharply, and in clique queries (q9–q11) they are often two to three orders of magnitude slower than PDD+. DSQL, PTAB, VEQ remain the slowest across all baselines, being consistently two to three orders of magnitude slower than PDD+ on DG10, with the gap widening further (up to three to four orders) on DG60.

Exp5 - Parallel Comparison. The experiment evaluates the parallel performance of our method against other parallelizable subgraph matching algorithms under varying thread counts using Query 1. Since existing diversified top-k subgraph matching algorithms do not support parallel execution, we compare our method against parallelizable subgraph matching algorithms including DAF and GuP. In our framework, the partition-based design enables parallelism throughout the entire process, whereas DAF and GuP apply parallelism only during the backtracking stage. As shown in Figure 8, PDD+ consistently outperforms the baselines, being about three times faster under fewer threads and reaching up to 6× speed-up as the thread count increases. For more details, our optimized algorithm achieves sub-100 ms runtime across all thread settings, whereas the baselines remain around 300 ms. PDD is slower than the baselines in the fewer-thread case, since it explores many partitions without results and incurs additional costs for inter-partition tree reconstruction. However, with more threads, these costs are effectively amortized through parallelism, allowing PDD to gradually surpass the baselines.

Exp6 - Single-Thread Comparison. The experiment compares the single-thread performance of our algorithm with DSQL, a diversified top-k subgraph matching algorithm with the same problem definition, across various real-world datasets using Query 1 to evaluate efficiency. As shown in Figure 9, the sequential version of PDD+ consistently achieves the lowest runtime, often 1–3 orders of magnitude faster than DSQL. On datasets with fewer results (e.g., *hp*, *wn*, *up*), sequential PDD incurs higher costs by frequently exploring partitions without feasible matches, whereas PDD+ incorporates a prediction stage that avoids such overhead by filtering out infeasible partitions in advance.

5.3 Effectiveness Evaluation

Exp7 - Results Quality Evaluation. We analyze the *coverage* – *distance* distribution patterns for PDD+ and baselines in Figure 12. For *coverage* (Definition 5), it is normalized by its exact optimal value—the highest *coverage* over all combinations of matches. For *distance* (Definition 7), we conduct a greedy algorithm with backtracking on all matches to calculate the approximate best distance to normalize *distance* metric. As depicted in Figure 12, about 95% of the PDD+ matches fall in the upper right region, indicating simultaneously high normalized *coverage* and high normalized *distance*, and thus superior diversity. In contrast, PDD achieves a comparable quality in only 51% of cases, with the rest exhibiting low normalized *distance*. DSQL performs even worse: despite generally high normalized *coverage*, its matches have low *distance* because it considers only vertex coverage. Other subgraph enumeration algorithms also lack diversity guarantees. In about 35% of the cases—the largest proportion—their top-k matches predominantly occupy the lower-left region, indicating both low coverage and distance diversity.

Exp8 - Distance-Time Trade-Off Evaluation. In Figure 13, we report the trade-off between diversity (distance) and efficiency (time), comparing PDD+ with PDD, DSQL, and the optimal distance value. For a fair baseline, we adapt existing subgraph matching methods to account for topological dispersion and treat them as the "Optimal" algorithm: first enumerating all matches with a subgraph isomorphism algorithm, then computing pairwise distances and selecting the most dispersed k results. Both distance and time are normalized for comparability. To ensure fair comparison, both distance and time are normalized. Specifically, the normalized distance is defined as $ND = dis/dis_E$, where dis_E is the best achievable distance. For time, we use $NT = \log_{10}(time)/\log_{10}(time_E)$, where $time_E$ is the elapsed time to compute the best distance. This normalization highlights the trade-off between solution diversity and efficiency across different algorithms. In Figure 13, we observe that PDD+ always uses less elapsed time to obtain a higher distance compared to PDD, DSQL, and the best distance computation. Overall, PDD+ achieves a favourable trade-off, reaching 65–80% of the optimal distance while being up to six orders of magnitude faster than exact computation. Compared to PDD, PDD+ is about 7× faster on average in runtime and achieves a 14.7% improvement on average in distance, demonstrating the effectiveness of our optimizations. Compared to DSQL, PDD+ is approximately 1725× faster in runtime and achieves at least a 76.8% improvement in distance, highlighting its superiority in both efficiency and diversity. For small-result cases (e.g., Query 9 on *ye* and Query 10 on *hu*), where fewer than 30 matches exist, both PDD+ and DSQL are able to enumerate all matches, thus reaching the maximum distance.

5.4 Scalability Evaluation

Since our problem involves diversified top-k matches, we assess scalability by varying k and by scaling the data at k=30. With space constraints, we report results from three datasets of different sizes and three representative query types. We compare our method to DSQL and representative classical baselines (CFL, RapidMatch), omitting other baselines with similar trends; additional results are detailed in the Appendix B [14].

Exp9 - Time-k Evaluation. As shown in Figure 14, PDD+ consistently achieves the fastest runtime across different real-world datasets and synthetic datasets, remaining nearly unaffected as k increases. PDD also shows stable runtime trends, but it is consistently slower than PDD+. RM demonstrates similarly fast performance and its runtime is almost insensitive to the value of k. In contrast, CFL exhibits relatively poor performance on several datasets, yet its runtime grows smoothly with k. DSQL, however, shows the most significant increase, with runtime escalating rapidly on most datasets as k becomes larger.

Exp10 - Datasize-k Evaluation. To analyze the scalability of PDD+ while reducing structural bias, we extract seven sub-datasets of varying sizes from DG10 using a simple query, a medium-complexity query, and a near-clique query to demonstrate trends of runtime. Overall, as shown in Figure 15, PDD+ achieves the best performance in most cases, with the lowest runtime and smooth growth. PDD also grows relatively steadily, whereas DSQL increases sharply with k, becoming orders of magnitude slower on large datasets. Subgraph matching baselines remain relatively stable. CFL and RM achieve low runtimes on small datasets. As the graph size increases, performance gaps between algorithms become more pronounced, with the superiority of PDD+ most evident under logarithmic scales. Overall, the results show that PDD+ maintains stable across differ-

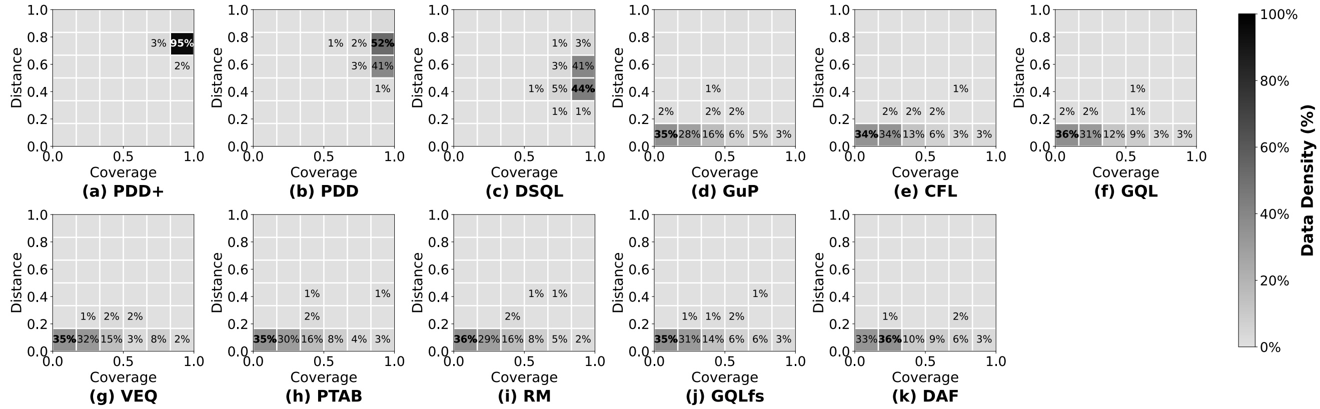


Figure 12: Coverage vs. Distance Metrics Across Valid Datasets and Queries

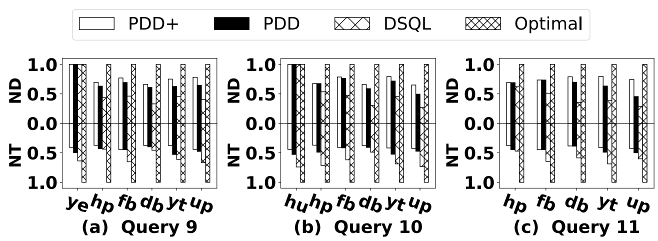


Figure 13: Distance–Time Trade-Off on Complex Queries.

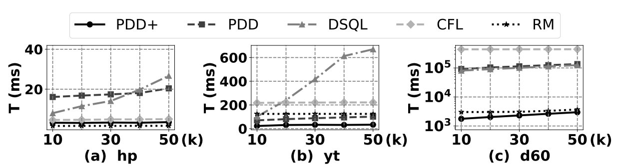


Figure 14: Runtime Variation with k for Query q1.

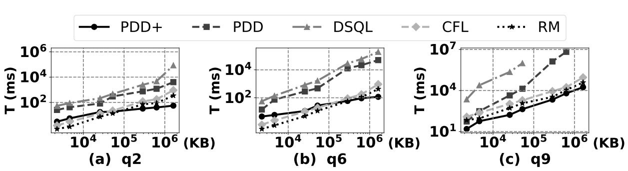


Figure 15: Runtime Variation with Data Size (k = 30).

ent k values and data scales, and handles large datasets without notable bottlenecks.

6 Related Work

Graph Partitioning. Graph partitioning approaches fall into three categories: vertex partitioning [43, 74, 75, 81], which minimizes edge cuts but struggles with high-degree nodes; edge partitioning [37, 58, 59, 101], which replicates vertices to improve load balance; and hybrid partitioning [16, 26, 32], which combines both to balance computation and communication.

Densest Subgraph. The densest subgraph problem is widely studied due to its applications in biology, summarization, and anomaly detection. Exact algorithms [31, 98] ensure optimality, while approximate methods [8, 12, 98] use greedy strategies to scale to large graphs. Variants extend to constraint-aware, objective-tuned, streaming, and k-clique settings [98].

Subgraph Matching: Classical Subgraph matching methods adopt filtering-ordering-enumeration paradigm [50, 104]. Algorithms [4–6, 11, 35, 38, 42, 78] enumerate subgraphs using backtracking strategies, and some of these can be parallelized, such as DAF [35] and GuP [4]. Other algorithms [45, 47, 48, 65, 88, 94, 96] enumerate results using joining strategies. RapidMatch [79] leverages nucleus decomposition to improve the efficiency of the join phase. Learning-based methods have been proposed for subgraph matching, including both approximate [52, 68] and exact approaches [54, 93, 99].

Result Diversification: Diversified result selection balances relevance and diversity to avoid redundant output, and it is widely applied in information retrieval [10, 22, 27, 39, 100], recommendation systems [29, 53, 64, 87, 92], database systems [19, 24] and subgraph matching algorithms [25, 95]. Typically, α-nDCG, P-Coverage, Jaccard Distance, M-IA, etc. [90] are used to evaluate the diversity of results in information retrieval and recommendation systems. For database systems, there are also result diversity rules [19, 24], such as tuple coverage and max-min distance. Some works take the diversity of subgraph matching results into consideration [25, 95].

7 Conclusion

We study the problem of top k diversified subgraph matching and identify limitations in existing methods in diversity quality. To address this, we propose a distance-based diversity metric and a partition-based framework with two optimizations that improve scalability and result dispersion.

ACKNOWLEDGEMENT

This work is supported by the Creative Research Groups Program of the National Natural Science Foundation of China (Grant No. 62321003), the National Natural Science Foundation of China (Grant Nos. U23A20317, 62572182, 62402481, 62576051), and the Natural Science Foundation of Hunan Province (Grant No. 2023JJ30083).

References

[1] Monica Agrawal, Marinka Zitnik, and Jure Leskovec. 2017. Large-scale analysis of disease pathways in the human interactome. arXiv:1712.00843 [q-bio.MN] https://arxiv.org/abs/1712.00843

[2] Renzo Angles, János Benjamin Antal, Alex Averbuch, Altan Birler, Peter Boncz, Márton Búr, Orri Erling, Andrey Gubichev, Vlad Haprian, Moritz Kaufmann, Josep Lluís Larriba Pey, Norbert Martínez, József Marton, Marcus Paradies, Minh-Duc Pham, Arnau Prat-Pérez, David Püroja, Mirko Spasić, Benjamin A. Steer, Dávid Szakállas, Gábor Szárnyas, Jack Waudby, Mingxi Wu, and Yuchen Zhang. 2024. The LDBC Social Network Benchmark. arXiv:2001.02299 [cs.DB] https://arxiv.org/abs/2001.02299

[3] Aissam Aouar, Saïd Yahiaoui, Lamia Sadeg, and Kadda Beghdad Bey. 2024. Scalable Diversified Top-k Pattern Matching in Big Graphs. *Big Data Res.* 36 (2024), 100464. https://doi.org/10.1016/J.BDR.2024.100464

[4] Junya Arai, Yasuhiro Fujiwara, and Makoto Onizuka. 2023. GuP: Fast Subgraph Matching by Guard-based Pruning. *Proc. ACM Manag. Data* 1, 2, Article 167 (June 2023), 26 pages. https://doi.org/10.1145/3589312

[5] Bibek Bhattarai, Hang Liu, and H. Howie Huang. 2019. CECI: Compact Embedding Cluster Index for Scalable Subgraph Matching. In *Proceedings of the 2019 International Conference on Management of Data* (Amsterdam, Netherlands) *(SIGMOD '19)*. Association for Computing Machinery, New York, NY, USA, 1447–1462. https://doi.org/10.1145/3299869.3300086

[6] Fei Bi, Lijun Chang, Xuemin Lin, Lu Qin, and Wenjie Zhang. 2016. Efficient Subgraph Matching by Postponing Cartesian Products. In *Proceedings of the 2016 International Conference on Management of Data* (San Francisco, California, USA) *(SIGMOD '16)*. Association for Computing Machinery, New York, NY, USA, 1199–1214. https://doi.org/10.1145/2882903.2915236

[7] Paul Boniol and Themis Palpanas. 2020. Series2Graph: graph-based subsequence anomaly detection for time series. *Proc. VLDB Endow.* 13, 12 (July 2020), 1821–1834. https://doi.org/10.14778/3407790.3407792

[8] Digvijay Boob, Yu Gao, Richard Peng, Saurabh Sawlani, Charalampos Tsourakakis, Di Wang, and Junxing Wang. 2020. Flowless: Extracting Densest Subgraphs Without Flow Computations. In *Proceedings of The Web Conference 2020* (Taipei, Taiwan) *(WWW '20)*. Association for Computing Machinery, New York, NY, USA, 573–583. https://doi.org/10.1145/3366423.3380140

[9] Florian Bourse, Marc Lelarge, and Milan Vojnovic. 2014. Balanced graph edge partition. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, New York, USA) *(KDD '14)*. Association for Computing Machinery, New York, NY, USA, 1456–1465. https://doi.org/10.1145/2623330.2623660

[10] Sebastian Bruch, Claudio Lucchese, and Franco Maria Nardini. 2022. ReNeuIR: Reaching Efficiency in Neural Information Retrieval. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Madrid, Spain) *(SIGIR '22)*. Association for Computing Machinery, New York, NY, USA, 3462–3465. https://doi.org/10.1145/3477495.3531704

[11] Vincenzo Carletti, Pasquale Foggia, Alessia Saggese, and Mario Vento. 2018. Challenging the Time Complexity of Exact Subgraph Isomorphism for Huge and Dense Graphs with VF3. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40, 4 (2018), 804–818. https://doi.org/10.1109/TPAMI.2017.2696940

[12] Moses Charikar. 2000. Greedy approximation algorithms for finding dense components in a graph. In *International workshop on approximation algorithms for combinatorial optimization*. Springer, 84–95.

[13] Liuyi Chen, Yi Ding, Xushuo Tang, Fangyue Chen, Siyuan Gong, Xu Zhou, and Zhengyi Yang. 2025. Accelerating Streaming Subgraph Matching via Vector Databases. *Intelligent Computing* (2025).

[14] Liuyi Chen, Yuchen Hu, Zhengyi Yang, Xu Zhou, Wenjie Zhang, and Kenli Li. 2025. Efficient Partition-based Approaches for Diversified Top-k Subgraph Matching. arXiv:2511.19008 [cs.DB] https://arxiv.org/abs/2511.19008

[15] L L Chen, N Blumm, N A Christakis, A-L Barabási, and T S Deisboeck. 2009. Cancer metastasis networks and the prediction of progression patterns. *British Journal of Cancer* 101, 5 (Aug. 2009), 749–758. https://doi.org/10.1038/sj.bjc.6605214

[16] Rong Chen, Jiaxin Shi, Yanzhe Chen, Binyu Zang, Haibing Guan, and Haibo Chen. 2018. PowerLyra: Differentiated Graph Computation and Partitioning on Skewed Graphs. *ACM Trans. Parallel Comput.* 5, 3 (2018), 13:1–13:39. https://doi.org/10.1145/3298989

[17] Wei Chen, Jia Liu, Ziyang Chen, Xian Tang, and Kaiyu Li. 2018. PBSM: An Efficient Top-K Subgraph Matching Algorithm. *International Journal of Pattern Recognition and Artificial Intelligence* 32, 06 (2018), 1850020. https://doi.org/10.1142/S0218001418500209 arXiv:https://doi.org/10.1142/S0218001418500209

[18] Y. Chen and J. Liu. 2012. Clustering Social Networks Using Distance-Preserving Subgraphs. In *The Influence of Technology on Social Network Analysis and Mining*. Springer, 331–349.

[19] Ting Deng and Wenfei Fan. 2014. On the Complexity of Query Result Diversification. 39, 2, Article 15 (May 2014), 46 pages. https://doi.org/10.1145/2602136

[20] Nicolas Guenon des Mesnards and Tauhid Zaman. 2018. Detecting Influence Campaigns in Social Networks Using the Ising Model. arXiv:1805.10244 [cs.SI] https://arxiv.org/abs/1805.10244

[21] Yi Ding, Zhengyi Yang, Shunyang Li, Liuyi Chen, Haoran Ning, Kongzhang Hao, and Yongfei Liu. 2024. FGAQ: Accelerating Graph Analytical Queries Using FPGA. In *Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference on Web and Big Data*. Springer, 357–361.

[22] Guglielmo Faggioli, Nicola Ferro, Cristina Ioana Muntean, Raffaele Perego, and Nicola Tonellotto. 2023. A Geometric Framework for Query Performance Prediction in Conversational Search. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Taipei, Taiwan) *(SIGIR '23)*. Association for Computing Machinery, New York, NY, USA, 1355–1365. https://doi.org/10.1145/3539618.3591625

[23] Wenfei Fan, Ziyan Han, Yaoshu Wang, and Min Xie. 2023. Discovering Top-k Rules using Subjective and Objective Criteria. *Proc. ACM Manag. Data* 1, 1, Article 70 (May 2023), 29 pages. https://doi.org/10.1145/3588924

[24] Wenfei Fan, Ziyan Han, Min Xie, and Guangyi Zhang. 2024. Discovering Top-k Relevant and Diversified Rules. *Proc. ACM Manag. Data* 2, 4 (2024), 195:1–195:28. https://doi.org/10.1145/3677131

[25] Wenfei Fan, Xin Wang, and Yinghui Wu. 2013. Diversified top-k graph pattern matching. *Proc. VLDB Endow.* 6, 13 (Aug. 2013), 1510–1521. https://doi.org/10.14778/2536258.2536263

[26] Wenfei Fan, Ruiqi Xu, Qiang Yin, Wenyuan Yu, and Jingren Zhou. 2023. Application-driven graph partitioning. *VLDB J.* 32, 1 (2023), 149–172. https://doi.org/10.1007/S00778-022-00736-2

[27] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE v2: Sparse Lexical and Expansion Model for Information Retrieval. arXiv:2109.10086 [cs.IR] https://arxiv.org/abs/2109.10086

[28] Chuchu Gao, Youhuan Li, Zhibang Yang, and Xu Zhou. 2024. CSM-TopK: Continuous Subgraph Matching with TopK Density Constraints. In *40th IEEE International Conference on Data Engineering, ICDE 2024, Utrecht, The Netherlands, May 13-16, 2024*. IEEE, 3084–3097. https://doi.org/10.1109/ICDE60146.2024.00239

[29] Yunfan Gao, Tao Sheng, Youlin Xiang, Yun Xiong, Haofen Wang, and Jiawei Zhang. 2023. Chat-REC: Towards Interactive and Explainable LLMs-Augmented Recommender System. arXiv:2303.14524 [cs.IR] https://arxiv.org/abs/2303.14524

[30] Yurun Ge, Dominic Yang, and Andrea L. Bertozzi. 2025. Iterative active learning strategies for subgraph matching. *Pattern Recognition* 158 (2025), 110797. https://doi.org/10.1016/j.patcog.2024.110797

[31] A. V. Goldberg. 1984. *Finding a Maximum Density Subgraph.* Technical Report. USA.

[32] Joseph E. Gonzalez, Reynold S. Xin, Ankur Dave, Daniel Crankshaw, Michael J. Franklin, and Ion Stoica. 2014. GraphX: Graph Processing in a Distributed Dataflow Framework. In *11th USENIX Symposium on Operating Systems Design and Implementation, OSDI '14, Broomfield, CO, USA, October 6-8, 2014*, Jason Flinn and Hank Levy (Eds.). USENIX Association, 599–613. https://www.usenix.org/conference/osdi14/technical-sessions/presentation/gonzalez

[33] Wentian Guo, Yuchen Li, Mo Sha, Bingsheng He, Xiaokui Xiao, and Kian-Lee Tan. 2020. GPU-Accelerated Subgraph Enumeration on Partitioned Graphs. In *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*. ACM, 1067–1082.

[34] Manish Gupta, Jing Gao, Xifeng Yan, Hasan Cam, and Jiawei Han. 2014. Top-K interesting subgraph discovery in information networks. In *2014 IEEE 30th International Conference on Data Engineering*. 820–831. https://doi.org/10.1109/ICDE.2014.6816703

[35] Myoungji Han, Hyunjoon Kim, Geonmo Gu, Kunsoo Park, and Wook-Shin Han. 2019. Efficient Subgraph Matching: Harmonizing Dynamic Programming, Adaptive Matching Order, and Failing Set Together. In *Proceedings of the 2019 International Conference on Management of Data* (Amsterdam, Netherlands) *(SIGMOD '19)*. Association for Computing Machinery, New York, NY, USA, 1429–1446. https://doi.org/10.1145/3299869.3319880

[36] Wook-Shin Han, Jinsoo Lee, and Jeong-Hoon Lee. 2013. Turboiso: towards ultrafast and robust subgraph isomorphism search in large graph databases. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data* (New York, New York, USA) *(SIGMOD '13)*. Association for Computing Machinery, New York, NY, USA, 337–348. https://doi.org/10.1145/2463676.2465300

[37] Masatoshi Hanai, Toyotaro Suzumura, Wen Jun Tan, Elvis S. Liu, Georgios Theodoropoulos, and Wentong Cai. 2019. Distributed Edge Partitioning for Trillion-edge Graphs. *Proc. VLDB Endow.* 12, 13 (2019), 2379–2392. https://doi.org/10.14778/3358701.3358706

[38] Huahai He and Ambuj K. Singh. 2008. Graphs-at-a-time: query language and access methods for graph databases. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data* (Vancouver, Canada) *(SIGMOD '08)*. Association for Computing Machinery, New York, NY, USA, 405–418. https://doi.org/10.1145/1376616.1376660

[39] Andrés Hoyos-Idrobo. 2023. Learning to Re-rank with Constrained Meta-Optimal Transport. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '23)*. ACM, 48–57.

https://doi.org/10.1145/3539618.3591714
[40] Liang Jiang, Lu Liu, Jingjing Yao, and Leilei Shi. 2020. A user interest community evolution model based on subgraph matching for social networking in mobile edge computing environments. *Journal of Cloud Computing* 9, 1 (2020), 1–17.
[41] Xin Jin, Zhengyi Yang, Xuemin Lin, Shiyu Yang, Lu Qin, and You Peng. 2021. FAST: FPGA-based Subgraph Matching on Massive Graphs. arXiv:2102.10768 [cs.DB] https://arxiv.org/abs/2102.10768
[42] Alpár Jüttner and Péter Madarasi. 2018. VF2++—An improved subgraph isomorphism algorithm. *Discrete Applied Mathematics* 242 (2018), 69–81. https://doi.org/10.1016/j.dam.2018.02.018 Computational Advances in Combinatorial Optimization.
[43] George Karypis and Vipin Kumar. 1998. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM J. Sci. Comput.* 20, 1 (1998), 359–392. https://doi.org/10.1137/S1064827595287997
[44] Hyunjoon Kim, Yunyoung Choi, Kunsoo Park, Xuemin Lin, Seok-Hee Hong, and Wook-Shin Han. 2021. Versatile Equivalences: Speeding up Subgraph Query Processing and Subgraph Matching. In *Proceedings of the 2021 International Conference on Management of Data* (Virtual Event, China) *(SIGMOD '21)*. Association for Computing Machinery, New York, NY, USA, 925–937. https://doi.org/10.1145/3448016.3457265
[45] Hyeonji Kim, Juneyoung Lee, Sourav S. Bhowmick, Wook-Shin Han, JeongHoon Lee, Seongyun Ko, and Moath H.A. Jarrah. 2016. DUALSIM: Parallel Subgraph Enumeration in a Massive Graph on a Single Machine. In *Proceedings of the 2016 International Conference on Management of Data* (San Francisco, California, USA) *(SIGMOD '16)*. Association for Computing Machinery, New York, NY, USA, 1231–1245. https://doi.org/10.1145/2882903.2915209
[46] Johanna Klughammer, Kristian Koedijk, Theo Raine, et al. 2024. Mapping metastatic breast cancer complexity through single-cell and spatial profiling. *Nature Medicine* 30, 4 (2024), 891–902. https://doi.org/10.1038/s41591-024-03308-9
[47] Longbin Lai, Lu Qin, Xuemin Lin, and Lijun Chang. 2015. Scalable subgraph enumeration in MapReduce. *Proc. VLDB Endow.* 8, 10 (June 2015), 974–985. https://doi.org/10.14778/2794367.2794368
[48] Longbin Lai, Lu Qin, Xuemin Lin, Ying Zhang, Lijun Chang, and Shiyu Yang. 2016. Scalable distributed subgraph enumeration. *Proc. VLDB Endow.* 10, 3 (Nov. 2016), 217–228. https://doi.org/10.14778/3021924.3021937
[49] Longbin Lai, Zhu Qing, Zhengyi Yang, Xin Jin, Zhengmin Lai, Ran Wang, Kongzhang Hao, Xuemin Lin, Lu Qin, Wenjie Zhang, et al. 2019. Distributed subgraph matching on timely dataflow. *Proceedings of the VLDB Endowment* 12, 10 (2019), 1099–1112.
[50] Longbin Lai, Zhu Qing, Zhengyi Yang, Xin Jin, Zhengmin Lai, Ran Wang, Kongzhang Hao, Xuemin Lin, Lu Qin, Wenjie Zhang, Ying Zhang, Zhengping Qian, and Jingren Zhou. 2019. A Survey and Experimental Analysis of Distributed Subgraph Matching. *CoRR* abs/1906.11518 (2019). arXiv:1906.11518 http://arxiv.org/abs/1906.11518
[51] Zhengmin Lai, Zhengyi Yang, and Longbin Lai. 2019. Improving Distributed Subgraph Matching Algorithm on Timely Dataflow. In *2019 IEEE 35th International Conference on Data Engineering Workshops (ICDEW)*. IEEE, 266–273.
[52] Zixun Lan, Limin Yu, Linglong Yuan, Zili Wu, Qiang Niu, and Fei Ma. 2023. Sub-GMN: The Neural Subgraph Matching Network Model. In *16th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics, CISP-BMEI 2023, Taizhou, China, October 28-30, 2023*, XiaoMing Zhao, Qingli Li, and Lipo Wang (Eds.). IEEE, 1–7. https://doi.org/10.1109/CISP-BMEI60920.2023.10373342
[53] Youhua Li, Hanwen Du, Yongxin Ni, Pengpeng Zhao, Qi Guo, Fajie Yuan, and Xiaofang Zhou. 2023. Multi-Modality is All You Need for Transferable Recommender Systems. arXiv:2312.09602 [cs.IR] https://arxiv.org/abs/2312.09602
[54] Xin Liu and Yangqiu Song. 2021. Graph Convolutional Networks with Dual Message Passing for Subgraph Isomorphism Counting and Matching. arXiv:2112.08764 [cs.LG] https://arxiv.org/abs/2112.08764
[55] Qiuyi Lyu, Mo Sha, Bin Gong, and Kuangda Lyu. 2021. Accelerating Depth-First Traversal by Graph Ordering. In *SSDBM 2021: 33rd International Conference on Scientific and Statistical Database Management, Tampa, FL, USA, July 6-7, 2021*. ACM, 13–24.
[56] Hanchao Ma, Sheng Guan, Christopher Toomey, and Yinghui Wu. 2022. Diversified Subgraph Query Generation with Group Fairness. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining* (Virtual Event, AZ, USA) *(WSDM '22)*. Association for Computing Machinery, New York, NY, USA, 686–694. https://doi.org/10.1145/3488560.3498525
[57] Houari Mahfoud. 2023. Diversified Top-k Answering of Cypher Queries over Large Data Graphs. In *20th ACS/IEEE International Conference on Computer Systems and Applications, AICCSA 2023, Giza, Egypt, December 4-7, 2023*. IEEE, 1–8. https://doi.org/10.1109/AICCSA59173.2023.10479283
[58] Daniel W. Margo and Margo I. Seltzer. 2015. A Scalable Distributed Graph Partitioner. *Proc. VLDB Endow.* 8, 12 (2015), 1478–1489. https://doi.org/10.14778/2824032.2824046
[59] Christian Mayer, Ruben Mayer, Muhammad Adnan Tariq, Heiko Geppert, Larissa Laich, Lukas Rieger, and Kurt Rothermel. 2018. ADWISE: Adaptive Window-Based Streaming Edge Partitioning for High-Speed Graph Processing. In *38th IEEE International Conference on Distributed Computing Systems, ICDCS 2018, Vienna, Austria, July 2-6, 2018*. IEEE Computer Society, 685–695. https://doi.org/10.1109/ICDCS.2018.00072
[60] Ruben Mayer, Kamil Orujzade, and Hans-Arno Jacobsen. 2020. 2PS: High-Quality Edge Partitioning with Two-Phase Streaming. arXiv:2001.07086 [cs.DC] https://arxiv.org/abs/2001.07086
[61] Giovanni Micale, Antonio Di Maria, Roberto Grasso, Vincenzo Bonnici, Alfredo Ferro, Dennis Shasha, Rosalba Giugno, and Alfredo Pulvirenti. 2025. MultiGraphMatch: A Subgraph Matching Algorithm for Multigraphs. *ACM Trans. Knowl. Discov. Data* 19, 5, Article 97 (May 2025), 36 pages. https://doi.org/10.1145/3728361
[62] Muhammad Anis Uddin Nasir, Aristides Gionis, Gianmarco De Francisci Morales, and Sarunas Girdzijauskas. 2017. Fully Dynamic Algorithm for Top-k Densest Subgraphs. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management* (Singapore, Singapore) *(CIKM '17)*. Association for Computing Machinery, New York, NY, USA, 1817–1826. https://doi.org/10.1145/3132847.3132966
[63] Diogo Pacheco, Pik-Mai Hui, Christopher Torres-Lugo, Bao Tran Truong, Alessandro Flammini, and Filippo Menczer. 2021. Uncovering Coordinated Networks on Social Media: Methods and Case Studies. arXiv:2001.05658 [cs.SI] https://arxiv.org/abs/2001.05658
[64] Aleksandr Vladimirovich Petrov and Craig Macdonald. 2023. gSASRec: Reducing Overconfidence in Sequential Recommendation Trained with Negative Sampling. In *Proceedings of the 17th ACM Conference on Recommender Systems (RecSys '23)*. ACM, 116–128. https://doi.org/10.1145/3604915.3608783
[65] Miao Qiao, Hao Zhang, and Hong Cheng. 2017. Subgraph matching: on compression and computation. *Proc. VLDB Endow.* 11, 2 (Oct. 2017), 176–188. https://doi.org/10.14778/3149193.3149198
[66] Lu Qin, Jeffrey Xu Yu, and Lijun Chang. 2012. Diversifying Top-K Results. arXiv:1208.0076 [cs.DB] https://arxiv.org/abs/1208.0076
[67] Xuguang Ren and Junhu Wang. 2016. Multi-query optimization for subgraph isomorphism search. *Proc. VLDB Endow.* 10, 3 (Nov. 2016), 121–132. https://doi.org/10.14778/3021924.3021929
[68] Rex, Ying, Zhaoyu Lou, Jiaxuan You, Chengtao Wen, Arquimedes Canedo, and Jure Leskovec. 2020. Neural Subgraph Matching. arXiv:2007.03092 [cs.LG] https://arxiv.org/abs/2007.03092
[69] Mo Sha, Yuchen Li, Bingsheng He, and Kian-Lee Tan. 2017. Accelerating Dynamic Graph Analytics on GPUs. *Proc. VLDB Endow.* 11, 1 (2017), 107–120.
[70] Mo Sha, Yuchen Li, and Kian-Lee Tan. 2019. GPU-based Graph Traversal on Compressed Graphs. In *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*. ACM, 775–792.
[71] Mo Sha, Yuchen Li, and Kian-Lee Tan. 2021. Self-adaptive Graph Traversal on GPUs. In *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*. ACM, 1558–1570.
[72] Karishma Sharma, Yizhou Zhang, Emilio Ferrara, and Yan Liu. 2021. Identifying Coordinated Accounts on Social Media through Hidden Influence and Group Behaviours. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining* (Virtual Event, Singapore) *(KDD '21)*. Association for Computing Machinery, New York, NY, USA, 1441–1451. https://doi.org/10.1145/3447548.3467391
[73] Yongjun Shen, Edward A. Fox, and Yaohang Li. 2014. Applied Graph-Mining Algorithms to Study Biomolecular Interaction Networks. *BioMed Research International* 2014 (2014), 1–11. https://doi.org/10.1155/2014/439476
[74] Min Shi, Peng Peng, Xu Zhou, Jiayu Liu, Guoqing Xiao, and Kenli Li. 2024. Connectivity-Oriented Property Graph Partitioning for Distributed Graph Pattern Query Processing. *Proc. ACM Manag. Data* 2, 6, Article 229 (Dec. 2024), 26 pages. https://doi.org/10.1145/3698804
[75] Isabelle Stanton and Gabriel Kliot. 2012. Streaming graph partitioning for large distributed graphs. In *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012*, Qiang Yang, Deepak Agarwal, and Jian Pei (Eds.). ACM, 1222–1230. https://doi.org/10.1145/2339530.2339722
[76] Cheng Su, Mingxin Gan, Xiao Sun, Juan Liu, Zili Zhang, and Qi Liu. 2016. Incorporating topological information for predicting robust cancer subnetwork markers in human protein-protein interaction network. *BMC Bioinformatics* 17, Suppl 13 (2016), 372. https://doi.org/10.1186/s12859-016-1224-1
[77] Shang Su and Xiaohong Li. 2021. Dive into Single, Seek Out Multiple: Probing Cancer Metastases via Single-Cell Sequencing and Imaging Techniques. *Cancers* 13, 5 (March 2021), 1067. https://doi.org/10.3390/cancers13051067
[78] Shixuan Sun and Qiong Luo. 2020. In-Memory Subgraph Matching: An In-depth Study. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data* (Portland, OR, USA) *(SIGMOD '20)*. Association for Computing Machinery, New York, NY, USA, 1083–1098. https://doi.org/10.1145/3318464.3380581
[79] Shixuan Sun, Xibo Sun, Yulin Che, Qiong Luo, and Bingsheng He. 2020. RapidMatch: a holistic approach to subgraph query processing. *Proc. VLDB Endow.*

14, 2 (Oct. 2020), 176–188. https://doi.org/10.14778/3425879.3425888
[80] V. A. Traag, L. Waltman, and N. J. van Eck. 2019. From Louvain to Leiden: guaranteeing well-connected communities. *Scientific Reports* 9, 1 (March 2019). https://doi.org/10.1038/s41598-019-41695-z
[81] Charalampos E. Tsourakakis, Christos Gkantsidis, Bozidar Radunovic, and Milan Vojnovic. 2014. FENNEL: streaming graph partitioning for massive scale graphs. In *Seventh ACM International Conference on Web Search and Data Mining, WSDM 2014, New York, NY, USA, February 24-28, 2014*, Ben Carterette, Fernando Diaz, Carlos Castillo, and Donald Metzler (Eds.). ACM, 333–342. https://doi.org/10.1145/2556195.2556213
[82] J. R. Ullmann. 1976. An Algorithm for Subgraph Isomorphism. *J. ACM* 23, 1 (Jan. 1976), 31–42. https://doi.org/10.1145/321921.321925
[83] Elena Valari, Maria Kontaki, and Apostolos N. Papadopoulos. 2012. Discovery of top-k dense subgraphs in dynamic graph collections. In *Proceedings of the 24th International Conference on Scientific and Statistical Database Management* (Chania, Crete, Greece) *(SSDBM'12)*. Springer-Verlag, Berlin, Heidelberg, 213–230. https://doi.org/10.1007/978-3-642-31235-9_14
[84] Todd L. Veldhuizen. 2013. Leapfrog Triejoin: a worst-case optimal join algorithm. arXiv:1210.0481 [cs.DB] https://arxiv.org/abs/1210.0481
[85] Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. 2016. Order-Embeddings of Images and Language. arXiv:1511.06361 [cs.LG] https://arxiv.org/abs/1511.06361
[86] Haiyong Wang, Chenyue Zhang, Jingze Zhang, Li Kong, Hui Zhu, and Jinming Yu. 2016. The prognosis analysis of different metastasis pattern in patients with different breast cancer subtypes: a SEER based study. *Oncotarget* 8, 16 (Dec. 2016), 26368–26379. https://doi.org/10.18632/oncotarget.14300
[87] Yuyan Wang, Pan Li, and Minmin Chen. 2025. The Blessing of Reasoning: LLM-Based Contrastive Explanations in Black-Box Recommender Systems. arXiv:2502.16759 [cs.IR] https://arxiv.org/abs/2502.16759
[88] Zhaokang Wang, Weiwei Hu, Guowang Chen, Chunfeng Yuan, Rong Gu, and Yihua Huang. 2021. Towards Efficient Distributed Subgraph Enumeration Via Backtracking-Based Framework. *IEEE Transactions on Parallel and Distributed Systems* 32, 12 (2021), 2953–2969. https://doi.org/10.1109/TPDS.2021.3076246
[89] Zhengdong Wang, Qiang Yin, and Longbin Lai. 2025. Path-Centric Cardinality Estimation for Subgraph Matching. *Proc. VLDB Endow.* 18, 9 (Sept. 2025), 3063–3076. https://doi.org/10.14778/3746405.3746428
[90] Haolun Wu, Yansen Zhang, Chen Ma, Fuyuan Lyu, Bowei He, Bhaskar Mitra, and Xue Liu. 2024. Result Diversification in Search and Recommendation: A Survey. arXiv:2212.14464 [cs.IR] https://arxiv.org/abs/2212.14464
[91] Shengrui XU, Tianchi Lu, Zikun Wang, and Jixiu Zhai. 2025. SCMPPI: Supervised Contrastive Multimodal Framework for Predicting Protein-Protein Interactions. arXiv:2504.02698 [cs.LG] https://arxiv.org/abs/2504.02698
[92] Boming Yang, Dairui Liu, Toyotaro Suzumura, Ruihai Dong, and Irene Li. 2023. Going Beyond Local: Global Graph-Enhanced Personalized News Recommendations. In *Proceedings of the 17th ACM Conference on Recommender Systems (RecSys '23)*. ACM, 24–34. https://doi.org/10.1145/3604915.3608801
[93] Bin Yang, Zhaonian Zou, and Jianxiong Ye. 2025. GNN-based Anchor Embedding for Exact Subgraph Matching. *CoRR* abs/2502.00031 (2025). https://doi.org/10.48550/ARXIV.2502.00031 arXiv:2502.00031
[94] Linglin Yang, Yuqi Zhou, Yue Pang, and Lei Zou. 2024. Efficient Pruned Top-K Subgraph Matching with Topology-Aware Bounds. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management* (Boise, ID, USA) *(CIKM '24)*. Association for Computing Machinery, New York, NY, USA, 2848–2857. https://doi.org/10.1145/3627673.3679790
[95] Zhengwei Yang, Ada Wai-Chee Fu, and Ruifeng Liu. 2016. Diversified Top-k Subgraph Querying in a Large Graph. In *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, Fatma Özcan, Georgia Koutrika, and Sam Madden (Eds.). ACM, 1167–1182. https://doi.org/10.1145/2882903.2915216
[96] Zhengyi Yang, Longbin Lai, Xuemin Lin, Kongzhang Hao, and Wenjie Zhang. 2021. HUGE: An Efficient and Scalable Subgraph Enumeration System. In *Proceedings of the 2021 International Conference on Management of Data* (Virtual Event, China) *(SIGMOD '21)*. Association for Computing Machinery, New York, NY, USA, 2049–2062. https://doi.org/10.1145/3448016.3457237
[97] Zhengyi Yang, Wenjie Zhang, Xuemin Lin, Ying Zhang, and Shunyang Li. 2023. Hgmatch: A match-by-hyperedge approach for subgraph matching on hypergraphs. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 2063–2076.
[98] Xiaowei Ye, Miao Qiao, Rong-Hua Li, Qi Zhang, and Guoren Wang. 2024. Scalable k-clique Densest Subgraph Search. arXiv:2403.05775 [cs.DS] https://arxiv.org/abs/2403.05775
[99] Yutong Ye, Xiang Lian, and Mingsong Chen. 2024. Efficient Exact Subgraph Matching via GNN-based Path Dominance Embedding. *Proc. VLDB Endow.* 17, 7 (2024), 1628–1641. https://doi.org/10.14778/3654621.3654630
[100] Hamed Zamani and Michael Bendersky. 2023. Multivariate Representation Learning for Information Retrieval. arXiv:2304.14522 [cs.IR] https://arxiv.org/abs/2304.14522
[101] Chenzi Zhang, Fan Wei, Qin Liu, Zhihao Gavin Tang, and Zhenguo Li. 2017. Graph Edge Partitioning via Neighborhood Heuristic. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*. ACM, 605–614. https://doi.org/10.1145/3097983.3098033
[102] Haiwei Zhang, Xiaofang Xie, Yanlong Wen, and Ying Zhang. 2020. A Twig-Based Algorithm for Top-k Subgraph Matching in Large-Scale Graph Data. In *Web Information Systems and Applications - 17th International Conference, WISA 2020, Guangzhou, China, September 23-25, 2020, Proceedings (Lecture Notes in Computer Science)*, Guojun Wang, Xuemin Lin, James A. Hendler, Wei Song, Zhuoming Xu, and Genggeng Liu (Eds.), Vol. 12432. Springer, 475–487. https://doi.org/10.1007/978-3-030-60029-7_43
[103] Jiawei Zhang and Philip S. Yu. 2015. Mutual Community Detection across Multiple Partially Aligned Social Networks. arXiv:1506.05529 [cs.SI] https://arxiv.org/abs/1506.05529
[104] Zhijie Zhang, Yujie Lu, Weiguo Zheng, and Xuemin Lin. 2024. A Comprehensive Survey and Experimental Study of Subgraph Matching: Trends, Unbiasedness, and Interaction. *Proc. ACM Manag. Data* 2, 1 (2024), 60:1–60:29. https://doi.org/10.1145/3639315
[105] Yating Zhao, Wenhan Wang, Shuxin Chen, Bocheng Zhang, Jiajun Wang, and Zhaokang Wang. 2024. Speeding Up Subgraph Matching Queries with Schema Guided Index. In *Proceedings of the 2024 3rd International Conference on Networks, Communications and Information Technology* (Xi'an, China) *(CNCIT '24)*. Association for Computing Machinery, New York, NY, USA, 34–38. https://doi.org/10.1145/3672121.3672129
[106] Vincent W. Zheng, Mo Sha, Yuchen Li, Hongxia Yang, Yuan Fang, Zhenjie Zhang, Kian-Lee Tan, and Kevin Chen-Chuan Chang. 2018. Heterogeneous Embedding Propagation for Large-Scale E-Commerce User Alignment. In *IEEE International Conference on Data Mining, ICDM 2018, Singapore, November 17-20, 2018*. IEEE Computer Society, 1434–1439.
[107] Lei Zou, Lei Chen, and Yansheng Lu. 2007. Top-k subgraph matching query in a large graph. In *Proceedings of the First Ph.D. Workshop in CIKM, PIKM 2007, Sixteenth ACM Conference on Information and Knowledge Management, CIKM 2007, Lisbon, Portugal, November 9, 2007*, Aparna S. Varde and Jian Pei (Eds.). ACM, 139–146. https://doi.org/10.1145/1316874.1316897