Technical Report for "Effient Partition-based Approaches for Diversified Top-k Subgraph Matching"

THEORETICAL ANALYSIS

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. We assume that all subgraph matches of query Q in data graph G have already been computed, forming the candidate set $\mathcal{R}$. We reduce from the classical Max–Min k-Dispersion problem, which is known to be NP-hard on metric inputs. Max–Min k-Dispersion. Given a finite point set $P = \{p_1, \ldots, p_n\}$ and a distance function $d : P \times P \to \mathbb{R}_{\geq 0}$ that is non-negative, symmetric, and satisfies $d(p, p) = 0$ (and possibly the triangle inequality), and a parameter $k \leq n$, the goal is to find a subset

$$S^* \subseteq P, \quad |S^*| = k, \text{that maximizes} \min_{u \neq v \in S^*} d(u, v). \tag{1}$$

Reduction construction. Let (P, d, k) be an arbitrary instance of Max–Min Dispersion.

- Data graph. For each point $p_i \in P$, create a vertex v_i. Construct a complete undirected graph $G = (V, E)$ over $\{v_1, \ldots, v_n\}$, and set the weight of each edge (v_i, v_j) to $d(p_i, p_j)$. Because d satisfies the triangle inequality, the shortest-path distance between v_i and v_j in G equals $d(p_i, p_j)$.
- Query and matches. Let Q be a single isolated vertex. Every v_i is a valid match. Define the candidate set as $\mathcal{R} = \{\{v_1\}, \ldots, \{v_n\}\}$.
- Distance between matches. For any two matches $R_i = \{v_i\}$ and $R_j = \{v_j\}$, define the inter-match distance as

$$d^*(R_i, R_j) = \text{dist}_G(v_i, v_j) = d(p_i, p_j). \tag{2}$$

- Objective. The DTkSM problem asks to find a subset $S \subseteq \mathcal{R}$ of size k that maximizes the minimum inter-match distance:

$$\max_{S \subseteq \mathcal{R}, |S|=k} \min_{R_i \neq R_j \in S} d^*(R_i, R_j). \tag{3}$$

Equivalence. Any solution $S \subseteq \mathcal{R}$ corresponds to a subset $P_S \subseteq P$ of the same size, and since $d^*(R_i, R_j) = d(p_i, p_j)$, the two problems are equivalent.

Complexity. The transformation uses $O(n^2)$ edges and n candidate matches, so its time and space costs are polynomial in n. Thus Max–Min Dispersion $\leq_P$ DTkSM, proving that DTkSM is NP-hard.

□

Under the specific assumptions, the approximation ratio ρ of the distance-based diversity is as follows:

Assumptions (A1)–(A5). Throughout, we consider a graph partitioned into disjoint subgraphs (or "partitions"). We impose the following conditions on each partition G_k:

(A1) Compactness. Each partition G_k lies within a ball of fixed radius d around its center c_k, i.e.,

$$\max_{x \in G_k} \text{dist}_G(x, c_k) \;\leq\; d.$$

(A2) Uniformity. The radius d in (A1) is the same for all partitions.

(A3) Boundary thickness. When moving from one partition to the next along a path in the Partition Adjacency Graph (PAG), any path that crosses an internal boundary must travel at least $2d$.

(A4) Filtering success. After a filtering stage, every retained partition contains at least one valid match; that is, partitions selected by the algorithm are not dead ends.

(A5) Match in the partition. For analysis, we only consider matches that are fully contained within a single partition.

Definition 0.1 (Approximation ratio)**.** For distance-based diversity, the approximation ratio is defined by

$$\rho \;=\; \frac{D_{\text{alg}}}{D^*},$$

where D_{alg} is the distance-based diversity achieved by the algorithm and D^* is the optimal distance-based diversity.

Lemma 1 (Distance envelope). *Let G_i and G_j be partitions whose hop distance in the PAG is $h = d_H(G_i, G_j)$. Under (A1) and (A2), the true distance between any vertices $u \in G_i$ and $v \in G_j$ satisfies*

$$2\,(h-1)\,d \;\le\; \text{dist}_G(u,v) \;\le\; 2\,(h+1)\,d.$$

Theorem 2 (Approximation ratio). *Let G_i and G_j be the pair of partitions selected by the algorithm as the farthest in the PAG, with hop distance $h = d_H(G_i, G_j)$. Under assumptions (A1)–(A4) and the boundary thickness (A3) for the lower bound, the approximation ratio defined in Definition 0.1 satisfies*

$$\frac{h-1}{h+1} \;\le\; \rho \;\le\; 1.$$

Proof. Since the ratio is defined by $\rho = D_{\text{alg}}/D^*$, it is immediate that $\rho \le 1$, because D_{alg} cannot exceed D^*.

For the lower bound, consider the partitions G_i and G_j with hop distance h. Lemma 1 gives

$$2(h-1)d \;\le\; \text{dist}_G(u,v) \;\le\; 2(h+1)d$$

for $u \in G_i$, $v \in G_j$. The algorithm selects a pair that realizes the left-hand (worst-case) bound, so its achieved diversity is $D_{\text{alg}} \ge 2(h-1)d$. On the other hand, an optimal solution cannot exceed the right-hand bound, hence $D^* \le 2(h+1)d$. Therefore

$$\rho \;=\; \frac{D_{\text{alg}}}{D^*} \;\ge\; \frac{2(h-1)d}{2(h+1)d} \;=\; \frac{h-1}{h+1},$$

proving the stated bounds on ρ. □

The lower bound $\frac{h-1}{h+1}$ increases with h and approaches 1 as $h \to \infty$, meaning that for partitions far apart in the PAG, the approximation ratio becomes arbitrarily close to one.

Remark. Firstly, although Theorem 2 is stated for pairwise distances, it naturally extends to the top-k setting. Specifically, every pair among the k selected matches must satisfy the same lower bound. In addition, since we do not select adjacent partitions, the h-hop distance between any two chosen partitions in the PAG is always at least 2. Furthermore, when the partition size decreases, the number of partitions increases, which makes it easier to select non-adjacent partitions that are farther apart in the PAG. As a result, the minimum inter-partition hop h tends to grow, thereby improving the lower bound of distance diversity (cf. Theorem 2). However, the partition size cannot be arbitrarily small. As illustrated in Experiment 1, when the partition size falls below the range of 1000–2000 vertices, the execution time increases because boundary vertex replication grows, leading to more cross-partition matches. Hence,

there exists a clear trade-off between distance diversity (which benefits from larger h) and execution time (which deteriorates when the partition size is too small).

EXPERIMENTAL RESULTS

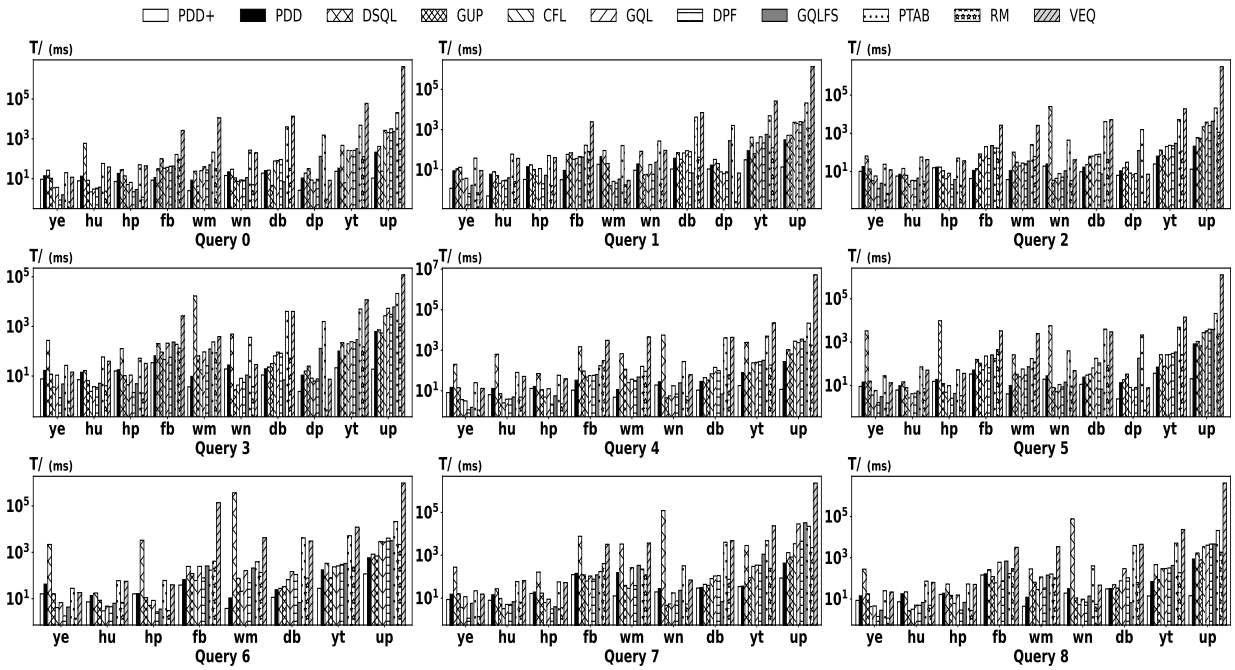

Fig. 1. Runtime of Different Query on Different Real-World Datasets (k = 30)

As shown in Figure 1, we evaluate PDD+ and baselines across all queries and read-world datasets. Some datasets are omitted for some queries when no valid match exists. For simple queries (Query 0–3), PDD and PDD+ generally achieve lower runtimes across most datasets, with a clear advantage on large graphs such as *YouTube* and *UP*. As dataset size increases, the runtime of most baselines (e.g., CFL, GUP, PTAB, VEQ) escalates rapidly, often reaching 10^5 ms, whereas PDD and PDD+ remain efficient. PDD+ also consistently outperforms DSQL, another top-k diversity subgraph matching method. Although DSQL achieves runtimes on the order of 100 ms, our approach still reduces latency by approximately 70%. On sparse graphs such as *wm* and *wn*, where few target results exist, PDD+ attains a speed-up of three orders of magnitude over DSQL, demonstrating robustness across data distributions. On smaller datasets (*ye*–*hp*), however, subgraph enumeration algorithms sometimes outperform PDD+; this is attributable to the comparable sizes of the partition graph and the original graph, therefore the processing time of classical methods is similar to our intra-partition matching time. In contrast, these classical methods suffer from extremely poor diversity, which is shown in Section 5.2. On larger datasets, PDD+ again surpasses these classical approaches. For moderately complex queries (Query 4–8), PDD+ achieves over an order-of-magnitude speed-up compared to DSQL, and up to four orders on sparse datasets such as *wn* and *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

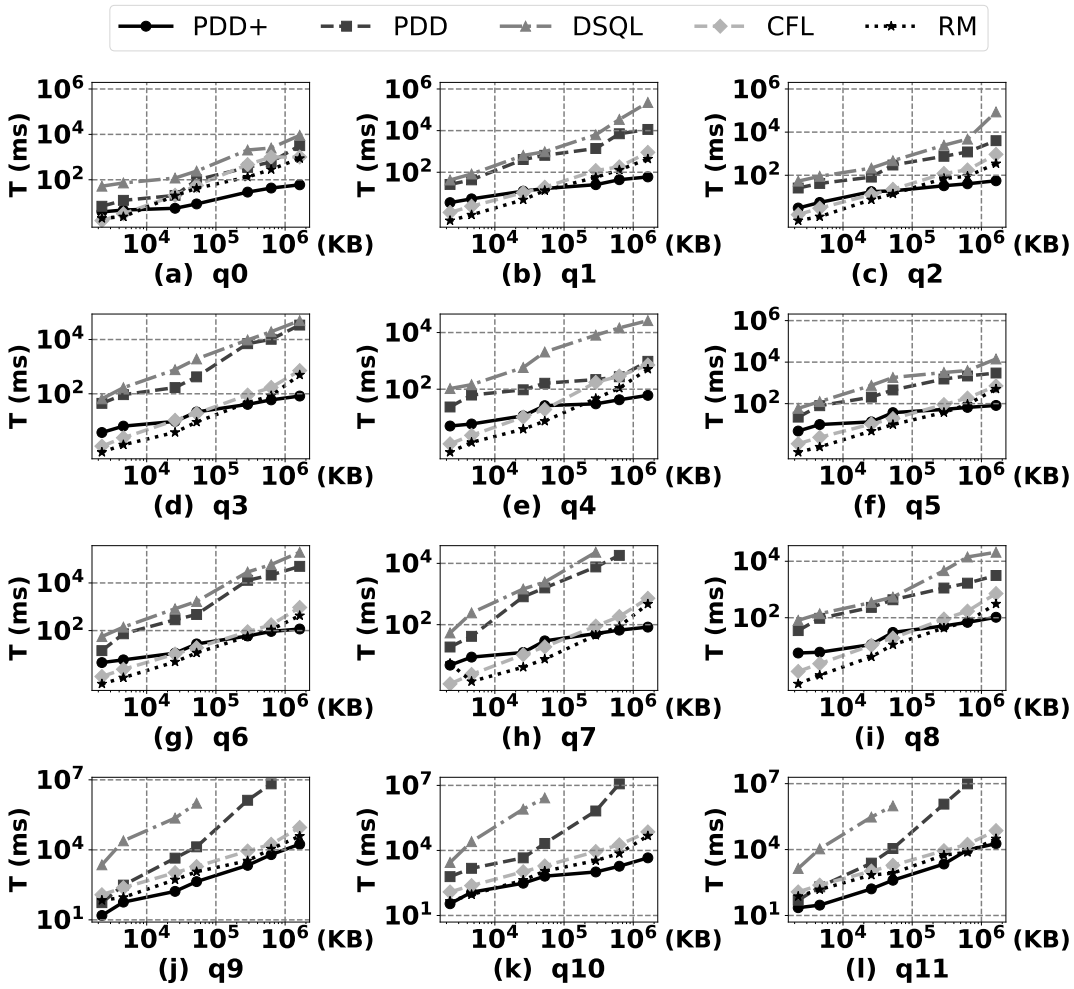

Fig. 2. Runtime Variation with Data Size (k = 30).

near-clique queries (Query 9–11), PDD+ still achieves two to three orders of magnitude speed-up over DSQL. Compared with subgraph enumeration methods, PDD+ is not always the fastest on small graphs, but it consistently shows clear advantages on medium and large datasets.

For simple queries, all methods exhibit low runtimes and smooth growth with increasing input size. PDD+ consistently performs the fastest, maintaining the lowest curve across all scales, while DSQL is noticeably slower and RM and CFL fall in between. The overall performance gap is small on smaller datasets but widens as data size increases. As query complexity increases, PDD+ shows the slowest growth in runtime, whereas the traditional top-k diversified subgraph matching method DSQL rises exponentially. Conventional subgraph matching algorithms perform relatively well at small scales but are gradually outperformed by PDD+ as the data grows. In contrast, although PDD is slightly slower than classical subgraph matching algorithms, it demonstrates greater stability than DSQL. For complex queries, the

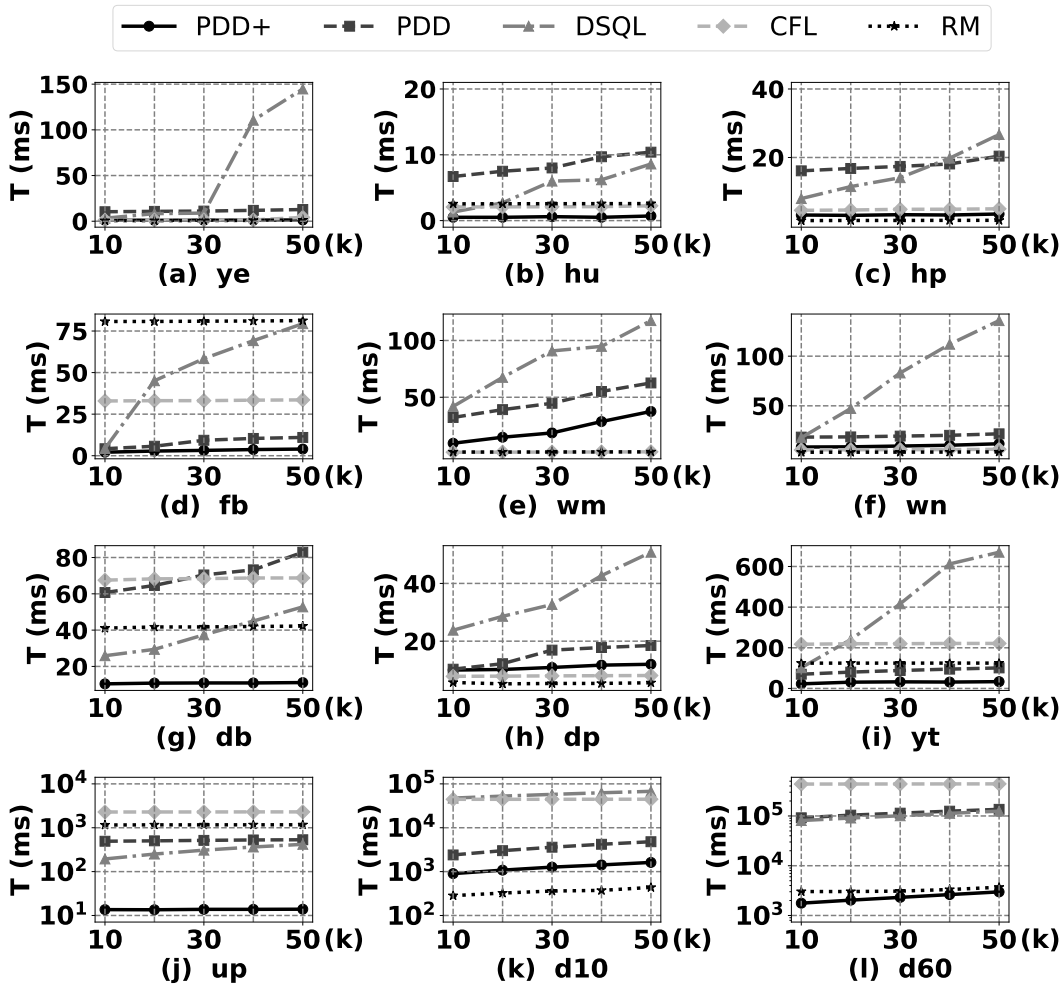

Fig. 3. Runtime Variation with k.

runtime of all methods increases by several orders of magnitude, yet PDD+ still maintains the lowest curve and the smoothest growth, indicating strong robustness and scalability.

This fig 3 presents the runtime comparison of different methods across twelve real-world datasets, where the parameter k varies from 10 to 50. Overall, PDD+ consistently achieves the lowest runtime on all datasets, demonstrating excellent scalability and stability. For smaller datasets (e.g., ye, hu, hp), all methods perform efficiently, yet PDD+ still maintains a clear advantage. As the dataset size increases (e.g., fb, wm, wn, db, dp, yt), the performance gap widens—DSQL shows a steep growth in runtime, while PDD+ and PDD exhibit smoother and much lower growth trends. On the largest datasets (up, d10, d60), this pattern becomes even more pronounced: DSQL and CFL experience exponential increases in runtime, whereas PDD+ maintains near-linear scalability, highlighting its efficiency and robustness on large-scale graph data.