MDS-FSM: Coverage-Based Frequent Subgraph Mining in Single Graphs

Xiaozhen Guo
Tianjin University
Tianjin, China
2024248119@tju.edu.cn

Xueli Liu*
Tianjin University
Tianjin, China
xueli@tju.edu.cn

Bowen Dong
Tianjin University
Tianjin, China
1020201107@tju.edu.cn

Li Wan
Tianjin University
Tianjin, China
2024248116@tju.edu.cn

Jiake Ge
Tianjin University
Tianjin, China
gejiake@tju.edu.cn

Shuai Ma
Beihang University
Beijing, China
mashuai@buaa.edu.cn

ABSTRACT

Frequent subgraph mining (FSM) in a single large graph remains challenging because pervasive embedding overlap exposes a gap between rigor and tractability: MIS-style supports enforce strict de-duplication but are NP-hard and enumeration-dependent, whereas MNI-style supports are polynomial-time yet systematically inflate frequency under distributed overlap. We propose Minimum Density Support (MDS), a coverage-based measure that minimizes coverage density over vertex subsets, uniformly penalizes redundant overlap, and preserves anti-monotonicity. MDS is theoretically bounded between MIS and MNI and can be computed in polynomial time via submodular minimization. We further develop MDS-FSM with orbit compression, separability, and progressive bound tightening to avoid exhaustive embedding enumeration. Experiments on six real graphs show that MDS reduces overestimation and cross-topology estimation bias while scaling to million-node graphs.

PVLDB Reference Format:
Xiaozhen Guo, Xueli Liu, Bowen Dong, Li Wan, Jiake Ge, and Shuai Ma. MDS-FSM: Coverage-Based Frequent Subgraph Mining in Single Graphs. PVLDB, 19(9): 2276 - 2288, 2026.
doi:10.14778/3819518.3819550

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/Wenzezhi/MDS-FSM.git.

1 INTRODUCTION

Graphs naturally model complex relational systems such as social networks [21, 37], biological systems [31], and knowledge graphs [4, 32]. Frequent subgraph mining (FSM) [15, 30] identifies recurring patterns in a single large graph. These patterns serve as reusable structural evidence, making FSM a core graph data management workload with applications in biological interaction analysis [25], fraud and risk monitoring [33], cyber-attack forensics [40], and knowledge graph exploration [18].

Defining what it means for a pattern to be "frequent" in a single large graph is nontrivial. A pattern often has many overlapping embeddings [17]. Simply counting embeddings is therefore unsuitable: it does not satisfy anti-monotonicity [2], ruling out principled pruning and scalable mining. Semantically, treating all embeddings as equally contributive is problematic: embeddings that largely overlap with existing ones provide less new structural evidence than those that cover previously unexplained regions.

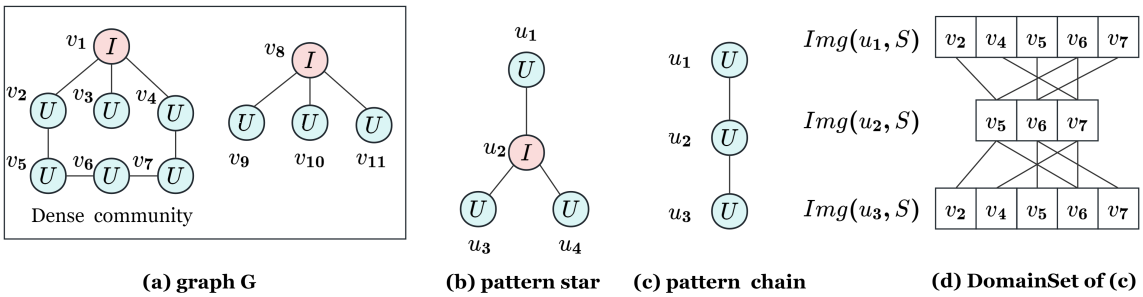
A classical notion MIS [35] counts only mutually non-overlapping occurrences, so each realization contributes distinct structural evidence. This matches the intuition that frequency should reflect how much structure a pattern accounts for. However, computing MIS is NP-hard [9] and tied to embedding enumeration. Subsequent relaxations and approximations [8, 36] reduce the cost, but largely retain MIS's binary treatment of overlap and reliance on embeddings.

To achieve scalability, modern single-graph FSM systems [1, 6, 43] turn to image-based or locality-based supports, most notably Minimum Node Image (MNI) [5]. MNI avoids explicit embedding enumeration by measuring frequency through candidate image-set sizes of pattern vertices, making it efficient and anti-monotone in practice. However, this design induces structural bias. Since MNI measures frequency purely through local image multiplicity, it rewards regions where many embeddings repeatedly reuse nearly the same graph part. In such cases, support increases even though these embeddings contribute little new structural coverage. Consequently, when overlap is common, MNI systematically inflates frequency, reflecting local combinational richness rather than how extensively the pattern is realized across the graph.

Example 1: Figure 1 illustrates this on a user–issue social network. The chain pattern (c) has MNI support 3 because $\{v_5, v_6, v_7\}$ are repeatedly reused as images of u_2 within one dense region, whereas the star pattern (b), whose embeddings span different regions without such reuse, has support only 2. Thus, although multi-region patterns are often more representative than those repeated within one dense cluster, MNI favors the latter by counting overlapping embeddings from the same region. Under heavy overlap, MNI may therefore reflect repeated generation from one dense region rather than broad realization of the pattern. □

*Corresponding author: Xueli Liu.

Proceedings of the VLDB Endowment, Vol. 19, No. 9 ISSN 2150-8097.
doi:10.14778/3819518.3819550



(a) graph G **(b) pattern star** **(c) pattern chain** **(d) DomainSet of (c)**

Figure 1: MNI Support Bias on a Social Network Example.

These observations show that the challenge in single-graph FSM goes beyond handling overlaps efficiently. The core difficulty is conceptual: existing support definitions implicitly assume that more embeddings always imply stronger frequency, while in reality different embeddings contribute unequally to structural evidence. Embeddings that repeatedly rely on the same dense region reflect local redundancy, whereas embeddings that appear across different network regions represent broader and more meaningful realization of the pattern. This reveals a tension between MIS-style supports, which emphasize independent realizations but are computationally expensive, and MNI-style supports, which enable efficient mining but may inflate frequency under heavy overlap.

To tackle this problem, we revisit frequency from a global coverage perspective rather than counting occurrences. Rather than asking how often a pattern can be instantiated, we ask how much of the graph it covers when all embeddings are considered together. Under this perspective, different embeddings no longer contribute equally: embeddings that extend the covered region make a meaningful contribution, whereas embeddings that mostly fall within already covered regions add little. Redundant overlaps are therefore discounted, without requiring the strict disjointness enforced by MIS, while benign overlaps remain harmless.

Building on this view, we propose a coverage-based support measure that evaluates frequency by how extensively a pattern is realized in the graph, rather than by how many embeddings can be formed. Using vertex image-set coverage as the basis for frequency, the measure consistently accounts for both concentrated and distributed overlap, avoiding inflation from repeatedly reusing the same graph regions. It preserves anti-monotonicity. Although its definition appears combinatorial, we show that it is polynomial-time computable via a submodular formulation; we introduce orbit-based compression, overlap separability, and progressive bound tightening to make computation practical in real mining workloads.

Contributions & Organizations. From a data management perspective, this work contributes new semantics, theory, and algorithms for scalable single-graph frequent subgraph mining:

(1) *A coverage-based support measure* (Section 3). We formalize *Minimum Density Support* (MDS), a support notion that penalizes redundant overlap and reflects how extensively a pattern is realized in the graph. MDS is theoretically bounded between MIS and MNI, preserves anti-monotonicity, and is shown to be polynomial-time computable via a submodular formulation.

(2) *A Top-k mining framework* (Section 4). We develop *MDS-FSM*, a Top-k single-graph FSM framework that computes MDS without explicit embedding enumeration. The framework incorporates orbit-based compression, overlap separability, and progressive bound tightening to achieve efficient pruning and scalable computation.

(3) *Extensive empirical validation* (Section 5). Experiments on six real-world graphs show that MDS improves support quality: it reduces frequency overestimation by 37% on average, decreases cross-topology estimation bias by 39%, and improves downstream graph-index query performance with 43.2% higher average pruning ratio and 12.2% lower average query time than the strongest baseline. MDS-FSM also maintains competitive runtime, scales to graphs with up to five million nodes, and remains competitive even where MNI-based mining already performs well, while providing more reliable, less biased support estimation.

Related Work. We categorize related work as follows.

Support Measures for Single-Graph FSM. Existing support notions fall into two families: (1) embedding-enumeration-based supports. MIS [35] defines frequency as the maximum number of pairwise non-overlapping embeddings, but computing it requires solving a maximum independent set problem, which is NP-complete [9] and impractical on large graphs. Refinements such as the HO-support relaxation [8] and edge-disjoint variants [36] do not change this: both MIS and HO-support remain NP-complete, and edge-disjoint formulations are NP-hard. Therefore, MIS-style supports still rely on (near-)explicit embedding reasoning and a binary notion of overlap, counting only strictly disjoint embeddings. (2) Image-based supports. MNI [5] counts the minimum vertex image-set size and is the de facto standard in practical graph mining because it is polynomial-time computable. Subsequent refinements relieve but do not eliminate it: MI merges symmetric pattern nodes into coarse-grained subsets [24] yet fails under cross-orbit overlap; a later MNI-based hypergraph variant further accounts for harmful overlap [23] while remaining MNI-based, and FS uses one-hop neighborhood expansion [43] but remains local.

Our work instead defines *Minimum Density Support*, a coverage-based notion that uniformly penalizes both centralized and distributed overlap. Unlike MIS, it remains polynomial-time computable via a submodular formulation; unlike MNI and its variants, it reflects how extensively a pattern is realized in the graph rather than how often it can be instantiated.

Single-Graph FSM Systems. Early systems such as SUBDUE [14] optimize an MDL objective rather than define frequency semantics. Although many support notions have been proposed, MIS-style definitions are NP-hard and impractical at scale, while image-based variants mostly refine MNI within the same paradigm. Accordingly, scalable single-graph FSM systems largely rely on MNI [1, 6, 19, 29, 41, 43], differing mainly in pruning, search strategy, or parallel execution. GraMi [6] formulates FSM as a CSP and employs arc-consistency pruning. Minting [19] develops tight bounds for best-first Top-k exploration. ScaleMine [1], T-FSM [43], FSMBUS [41], and SSiGraM [29] focus on parallel and distributed execution via approximate-then-exact processing, work-stealing, and distributed or shared-memory architectures. Approximate approaches such as MaNIACS [28] (sampling-based) and learning-based estimation [42] trade accuracy for scalability, but still evaluate frequency under existing support definitions.

Unlike most prior systems, which speed up single-graph FSM through stronger pruning, exploration strategies, or parallelization, our work departs at the mining-technique level. We introduce a coverage-driven paradigm that reasons over embeddings collectively, yielding different pruning signals, search dynamics, and patterns with stronger global structural support.

2 PRELIMINARIES

We first review the basic notation used in this paper. Let Σ be a finite vertex label set. Unless otherwise stated, this paper considers simple, undirected, vertex-labeled graphs.

Data Graph. A data graph is an undirected *vertex-labeled* graph $G = (V, E, L)$, where (1) V is a finite vertex set; (2) $E \subseteq \{\{u, v\} \mid u, v \in V,\ u \neq v\}$ is a set of undirected edges represented as unordered pairs of distinct vertices (hence no self-loops or multi-edges); and (3) $L : V \to \Sigma$ assigns each vertex $u \in V$ a label $L(u) \in \Sigma$.

Pattern Graph. A pattern graph is an undirected *vertex-labeled* graph $Q = (V_q, E_q, f_v)$ defined over the label alphabet Σ, where (1) V_q is the vertex set; (2) $E_q \subseteq \{\{u, v\} \mid u, v \in V_q,\ u \neq v\}$ is the edge set; and (3) $f_v : V_q \to \Sigma$ assigns each vertex $u \in V_q$ a label $f_v(u) \in \Sigma$.

Subgraph Isomorphism. A subgraph isomorphism match (embedding) of pattern S into data graph G is an injective mapping $f : V_S \to V$ satisfying: (1) for all $u \in V_S$, $L_S(u) = L(f(u))$ (label-preserving); (2) for all $(u, v) \in E_S$, $(f(u), f(v)) \in E$ (adjacency-preserving) [34]. The set of embeddings is denoted $\mathrm{Emb}(S, G)$. We use *realization* interchangeably with *embedding* to emphasize that each embedding is one occurrence of the pattern in the data graph.

Embedding Overlap. Two embeddings $f_1, f_2 \in \mathrm{Emb}(S, G)$ are *overlapping* if $f_1(V_S) \cap f_2(V_S) \neq \emptyset$; otherwise they are *non-overlapping*.

Image Set. For pattern vertex $u \in V_S$, its image set is defined as $\mathrm{Img}(u; S) = \{f(u) \mid f \in \mathrm{Emb}(S, G)\}$ [5, 6]. When the pattern S is clear from context, we write $\mathrm{Img}(u)$ for brevity. Intuitively, $\mathrm{Img}(u)$ is the set of all data vertices that can play the role of u in at least one valid embedding. For example, in the chain pattern of Figure 1(c), $\mathrm{Img}(u_2) = \{v_5, v_6, v_7\}$, as each of these vertices appears as the image of u_2 in some valid embedding.

DomainSet. The collection of all image sets is called the DomainSet, denoted $\mathrm{Dom}(S) = \{\mathrm{Img}(u; S) : u \in V_S\}$.

Automorphism Group. The automorphism group $\mathrm{Aut}(S)$ of S consists of all vertex permutations $\pi : V_S \to V_S$ that preserve both adjacency and labels [22]: (1) for all $(u, v) \in E_S$, $(\pi(u), \pi(v)) \in E_S$; (2) for all $u \in V_S$, $L_S(\pi(u)) = L_S(u)$.

Orbits. The action of $\mathrm{Aut}(S)$ on V_S partitions the vertices into a set of orbits $\{O_1, \ldots, O_m\}$ [10]. Vertices in the same orbit are structurally equivalent under automorphisms and thus play symmetric roles in all embeddings.

For example, in Figure 1(c), there exists an automorphism that swaps u_1 and u_3. Hence, u_1 and u_3 belong to the same orbit $O_1 = \{u_1, u_3\}$, and u_2 forms a separate orbit $O_2 = \{u_2\}$.

MIS Support. The Maximum Independent Set support [35] is defined as $\mathrm{MIS}(S, G) = \max\{|I| : I \subseteq \mathrm{Emb}(S, G),\ \forall f \neq f' \in I,\ f(V_S) \cap f'(V_S) = \emptyset\}$. Computing MIS is NP-hard [9].

MNI Support. The Minimum Node Image support [5] is defined as $\mathrm{MNI}(S, G) = \min_{u \in V_S} |\mathrm{Img}(u; S)|$. MNI can be computed in polynomial time [6].

Submodular Function. A set function $h : 2^X \to \mathbb{R}$ defined on a finite ground set X is *submodular* if for any $A \subseteq B \subseteq X$ and any $x \in X \setminus B$, $h(A \cup \{x\}) - h(A) \geq h(B \cup \{x\}) - h(B)$. This property reflects a *diminishing-returns* behavior: the marginal gain of adding an element decreases as the context grows.

3 MINIMUM DENSITY SUPPORT

This section presents a single-graph support measure, Minimum Density Support (MDS), which evaluates frequency from a global coverage perspective and penalizes redundant embedding overlap.

3.1 MDS Definition.

To precisely characterize the degree of overlap among image sets of vertex subsets, we first introduce the concept of coverage size.

Definition 3.1:[Coverage Size] Given a pattern S and a non-empty subset $U \subseteq V_S$ of its vertices, the coverage size of U is defined as

$$F_S(U) = \left| \bigcup_{u \in U} \mathrm{Img}(u; S) \right|, \tag{1}$$

i.e., the cardinality of the union of image sets of all vertices in U. Intuitively, $F_S(U)$ counts the distinct data nodes covered by U. □

Definition 3.2:[Minimum Density Support] Let S be a pattern graph and G a data graph. Define

$$\rho^*(S, G) = \min_{\emptyset \neq U \subseteq V_S} \frac{F_S(U)}{|U|}.$$

The *Minimum Density Support* of S in G is defined as $\mathrm{MDS}(S, G)$ where $\mathrm{MDS}(S, G) = \lfloor \rho^*(S, G) \rfloor$.

□

Intuitively, coverage density measures how broadly embeddings of a pattern are distributed across the graph. A high density indicates support from distinct regions, whereas a low density suggests embeddings are concentrated in a limited portion of the graph.

Discussion. The minimum-density formulation adopts a worst-case view of coverage, where frequency is determined by the least well-distributed subset of vertices, enforcing consistent structural coverage across the pattern. This choice is not arbitrary: it both controls overlap-induced inflation and preserves anti-monotonicity, enabling efficient mining. While this formulation is well suited for tasks that favor non-redundant coverage, in applications such as motif analysis or community detection, overlap may itself carry structural significance, and alternative variants (e.g., average-based or orbit-weighted measures) may be more appropriate.

Example 2: Recall the chain pattern S_{chain} in Figure 1(c), where MNI = 3 and MIS = 1. We compute MDS: here $\text{Img}(u_1) = \text{Img}(u_3) = \{v_2, v_4, v_5, v_6, v_7\}$ and $\text{Img}(u_2) = \{v_5, v_6, v_7\}$. For the full vertex set $V_S = \{u_1, u_2, u_3\}$, the union coverage is still $\{v_2, v_4, v_5, v_6, v_7\}$, so the coverage density is 5/3, yielding $\text{MDS}(S_{chain}, G) = \lfloor 5/3 \rfloor = 1$. This matches MIS, showing that MDS correctly identifies the complete overlap that MNI misses. □

Anti-Monotonicity Properties. Anti-monotonicity is the theoretical base of Apriori-style search strategies: if $\text{MDS}(S) < \theta$, then all super-patterns of S are infrequent, and the entire search branch can be safely pruned. We show that MDS satisfies anti-monotonicity.

Theorem 1:*[Anti-Monotonicity] If T is a super-pattern of S (*i.e., *$T \supseteq S$), then* $\text{MDS}(T) \leq \text{MDS}(S)$. □

Proof: Let $T \supseteq S$, *i.e., T* is obtained from S by adding vertices or edges. Since the extension operation strengthens matching constraints, we have $\text{Emb}(T, G) \subseteq \text{Emb}(S, G)$.

For any $u \in V_S$, by the relation of embedding sets, we have

$$\begin{aligned} \text{Img}(u; T) &= \{f(u) : f \in \text{Emb}(T, G)\} \\ &\subseteq \{f(u) : f \in \text{Emb}(S, G)\} = \text{Img}(u; S). \end{aligned} \tag{2}$$

Furthermore, for any non-empty subset $U \subseteq V_S$,

$$F_T(U) = \left| \bigcup_{u \in U} \text{Img}(u; T) \right| \leq \left| \bigcup_{u \in U} \text{Img}(u; S) \right| = F_S(U).$$

Therefore $F_T(U)/|U| \leq F_S(U)/|U|$. Taking the minimum over all non-empty $U \subseteq V_S$, we get $\text{MDS}(T) \leq \text{MDS}(S)$. □

Computational Complexity. The MDS definition involves enumeration over $2^{|V_S|} - 1$ non-empty subsets, which appears to have exponential complexity. However, we reveal that the MDS computation problem can be reduced to submodular function minimization, thereby achieving polynomial-time solvability. We first establish that the coverage function is submodular.

Lemma 2:*[Submodularity of the Coverage Function] For a pattern S, define*

$$F_S(U) = \left| \bigcup_{u \in U} \text{Img}(u; S) \right|, \qquad U \subseteq V_S.$$

Then F_S is a monotone submodular set function over V_S. □

Proof: Let $A \subseteq B \subseteq V_S$ and $x \in V_S \setminus B$. The marginal gain of adding x to A is $F_S(A \cup \{x\}) - F_S(A) = \left| \text{Img}(x; S) \setminus \bigcup_{u \in A} \text{Img}(u; S) \right|$. Since $A \subseteq B$, we have $\bigcup_{u \in A} \text{Img}(u; S) \subseteq \bigcup_{u \in B} \text{Img}(u; S)$, hence $F_S(A \cup \{x\}) - F_S(A) \geq F_S(B \cup \{x\}) - F_S(B)$. So F_S satisfies diminishing marginal returns and is submodular. □

Intuitively, the function $F_S(U)$ measures how many distinct vertices are covered by the image sets of nodes in U. When adding a new node u, its marginal contribution corresponds to the number of *new* vertices introduced by $\text{Img}(u; S)$. As U grows, more of these vertices are already covered, and thus fewer new ones can be contributed. This diminishing overlap directly leads to submodularity.

Theorem 3:*[Polynomial-Time Computability of MDS] Given the image sets $\{\text{Img}(u; S)\}_{u \in V_S}$, $\text{MDS}(S)$ can be computed exactly in polynomial time.* □

Proof: We compute $\rho^*(S)$ via a reduction from optimization to decision. For a given integer $k \geq 0$, consider the decision problem of whether $\rho^*(S) < k$. Observe that $\rho^*(S) < k$ holds if and only if there exists a non-empty set $U \subseteq V_S$ such that $\frac{F_S(U)}{|U|} < k$, which is equivalent to $F_S(U) - k|U| < 0$.

To test this condition, define $h_k(U) = F_S(U) - k|U|$ for $U \subseteq V_S$, with $h_k(\emptyset) = 0$. Then $\rho^*(S) < k$ if and only if $\min_{U \subseteq V_S} h_k(U) < 0$.

Since F_S is submodular (Lemma 2) and $|U|$ is modular, h_k is submodular. Thus, each such decision can be solved by minimizing a submodular function in polynomial time [26].

Since $0 \leq \rho^*(S) \leq |V(G)|$, we recover $\rho^*(S)$ via binary search over $k \in [0, |V(G)|]$, which requires only $O(\log |V(G)|)$ decision calls. Therefore, $\text{MDS}(S)$ is computable in polynomial time. □

Support Ordering Property. Let $\text{MIS}(S, G)$ denote the classical MIS support, i.e., the maximum number of pairwise vertex-disjoint embeddings of S in G. MDS has the following ordering property.

Theorem 4:*[Theoretical Bounding Property of MDS] For any pattern S and data graph G,* $\text{MIS}(S, G) \leq \text{MDS}(S, G) \leq \text{MNI}(S, G)$. □

Proof: For the lower bound, let $\text{MIS}(S, G) = k$ and let $f_1, \ldots, f_k$ be pairwise vertex-disjoint embeddings. For any non-empty $U \subseteq V_S$, define $A_i = \{f_i(u) \mid u \in U\}$. Then $|A_i| = |U|$, the sets A_i are pairwise disjoint, and $A_i \subseteq \bigcup_{u \in U} \text{Img}(u; S)$. Hence $F_S(U) \geq \left| \bigcup_{i=1}^{k} A_i \right| = k|U|$, so $F_S(U)/|U| \geq k$ for all $U \neq \emptyset$. Thus $\rho^*(S, G) \geq k$, implying $\text{MDS}(S, G) \geq \text{MIS}(S, G)$. For the upper bound, for any $u \in V_S$, $\rho^*(S, G) \leq F_S(\{u\})/1 = |\text{Img}(u; S)|$. Minimizing over u gives $\rho^*(S, G) \leq \text{MNI}(S, G)$, and hence $\text{MDS}(S, G) \leq \text{MNI}(S, G)$. □

This establishes that MDS never underestimates the number of structurally independent realizations of a pattern (in the spirit of MIS), while never exceeding the widely adopted MNI measure. Here independent realizations correspond to occurrences of the pattern that rely on largely distinct sets of graph vertices, rather than repeatedly reusing the same local neighborhood.

3.2 MDS Computation Optimization.

The previous subsection proves that MDS can be computed exactly in polynomial time via submodular minimization. This establishes a strong theoretical guarantee: unlike MIS, MDS is not intrinsically NP-hard. However, polynomial-time solvability in theory does not automatically imply practical efficiency. Known

strongly polynomial algorithms for submodular function minimization exhibit high-degree complexity and substantial constants (e.g., $O(n^5 \cdot EO + n^6)$ [26]), and in our fractional setting additional parametric iterations are required. Thus, while submodular minimization provides important theoretical tractability, it is far from lightweight and unsuitable as a practical evaluation mechanism.

Therefore, rather than invoking submodular optimization, we exploit MDS-specific structure. Our goal is not merely "polynomial solvability", but scalable and efficient computation in real mining workloads. We develop two complementary techniques, **orbit compression** and **subset separability**, together with tight upper–lower bounds to shrink the search space and avoid exact evaluation whenever possible.

Orbit Properties. Vertices in a pattern graph often exhibit strong structural symmetry. Under graph automorphisms, symmetric vertices always play equivalent structural roles and therefore induce identical behavior in candidate image sets. Such symmetry naturally partitions the vertex set into equivalence classes, called *orbits*, where all vertices in the same orbit are structurally identical from the perspective of pattern matching. We formalize it as follows.

Theorem 5:*[Orbit Image Set Invariance] Let* $\mathrm{Aut}(S)$ *be the automorphism group of pattern* S*, which induces an orbit partition* $\{O_1, \ldots, O_m\}$ *over* V_S*. If two vertices* u *and* v *belong to the same orbit* O_i*, then* $\mathrm{Img}(u; S) = \mathrm{Img}(v; S)$. □

Proof: Let $u, v \in O_i$. Then there exists $\pi \in \mathrm{Aut}(S)$ such that $\pi(u) = v$. For any $w \in \mathrm{Img}(u; S)$ there exists $f \in \mathrm{Emb}(S, G)$ with $f(u) = w$. Define $f' = f \circ \pi^{-1}$. Since π preserves labels and adjacency, f' is also an embedding. Moreover $f'(v) = f(\pi^{-1}(v)) = f(u) = w$, hence $w \in \mathrm{Img}(v; S)$. Thus $\mathrm{Img}(u; S) \subseteq \mathrm{Img}(v; S)$. The reverse inclusion follows symmetrically. □

Thus, all vertices in an orbit contribute the same image set. We next show that including only part of an orbit is never beneficial.

Theorem 6:*[Orbit Block Optimality] Let* $\{O_1, \ldots, O_m\}$ *be the orbits of pattern* S*. For any non-empty subset* $U \subseteq V_S$*, define its* orbit closure

$$U' = \bigcup_{i: U \cap O_i \neq \varnothing} O_i,$$

i.e., the union of all orbits that intersect U*. Then the coverage density cannot increase after completing the subset to full orbits:*

$$\frac{F_S(U')}{|U'|} \leq \frac{F_S(U)}{|U|}.$$

Consequently, the minimum density of the MDS objective is always attained by a subset that contains entire orbits. □

Proof: For any orbit O_i with $U \cap O_i \neq \varnothing$, pick $u_0 \in U \cap O_i$. By Theorem 5, all vertices in O_i have identical image sets, so

$$\bigcup_{u \in U \cap O_i} \mathrm{Img}(u; S) = \bigcup_{u \in O_i} \mathrm{Img}(u; S).$$

Thus $\bigcup_{u \in U} \mathrm{Img}(u; S) = \bigcup_{u \in U'} \mathrm{Img}(u; S)$ and $F_S(U) = F_S(U')$. Since $|U'| \geq |U|$, it follows that

$$\frac{F_S(U')}{|U'|} \leq \frac{F_S(U)}{|U|}.$$

Thus, the minimum is attained by a subset of complete orbits. □

From Theorem 6, the minimum in the MDS definition is always attained by a subset consisting of complete orbits. Hence the enumeration space reduces from $2^{|V_S|} - 1$ to $2^m - 1$, where m is the number of orbits. For highly symmetric patterns the reduction is dramatic: e.g., all vertices of an n-cycle form a single orbit ($m = 1$), and an n-star decomposes into two orbits (center vs. leaves, $m = 2$), yielding constant enumeration complexity.

Subset Separability. Even after orbit compression, embeddings associated with different orbits may cover largely disjoint regions of the data graph. In this case the contributions of these orbits become independent in the MDS objective: combining them cannot produce a density smaller than that of either subset individually.

Theorem 7:*[Separable Computation Theorem] Let* $U_1, U_2 \subseteq V_S$ *satisfy the disjoint image condition:*

$$\Big(\bigcup_{u \in U_1} \mathrm{Img}(u; S) \Big) \cap \Big(\bigcup_{v \in U_2} \mathrm{Img}(v; S) \Big) = \varnothing.$$

Then

$$\frac{F_S(U_1 \cup U_2)}{|U_1 \cup U_2|} \geq \min \left\{ \frac{F_S(U_1)}{|U_1|}, \frac{F_S(U_2)}{|U_2|} \right\}.$$

□

Proof: By disjointness, $F_S(U_1 \cup U_2) = F_S(U_1) + F_S(U_2)$. Let $\rho_1 = F_S(U_1)/|U_1|$ and $\rho_2 = F_S(U_2)/|U_2|$, and assume $\rho_1 \leq \rho_2$. Then

$$\frac{F_S(U_1 \cup U_2)}{|U_1 \cup U_2|} = \frac{|U_1|\rho_1 + |U_2|\rho_2}{|U_1| + |U_2|},$$

which is a weighted average between ρ_1 and ρ_2, hence at least ρ_1. □

Corollary 8: *Construct an overlap graph on the set of orbits, where two orbits are connected if their image sets intersect. Let* $\{C_1, \ldots, C_\ell\}$ *denote the connected components of this graph. Then the MDS objective decomposes across these components:*

$$\mathrm{MDS}(S) = \min_{1 \leq j \leq \ell} \mathrm{MDS}_{C_j}(S).$$

Here $\mathrm{MDS}_{C_j}(S)$ *denotes the minimum density obtained by restricting the subset* U *to vertices whose orbits belong to component* C_j*. If the largest component contains* m' *orbits, the resulting enumeration cost becomes* $O(\ell \cdot 2^{m'})$. □

Together, orbit compression and subset separability bridge the gap between theoretical polynomial-time computability and practical mining efficiency, eliminating most redundant subsets and enabling lightweight evaluation.

4 ALGORITHMS

We consider the Top-k single-graph frequent subgraph mining problem under the MDS support.

Problem Formulation. Given a data graph G and an integer $k > 0$, let $\sigma(\cdot)$ denote the MDS support of a pattern in G, and $\mathcal{P}$ denote the set of all connected, non-isomorphic patterns in G. The Top-k MDS-based frequent pattern mining problem is to compute

$$\text{Top-}k(G) = \arg\operatorname{top-k}_{S \in \mathcal{P}} \sigma(S),$$

i.e., the subset of $\mathcal{P}$ consisting of the k patterns with the largest MDS support in G.

4.1 MDS Evaluation

Evaluating $\mathsf{MDS}(S)$ requires computing possible embeddings of the pattern S in the data graph G. As explicitly enumerating all embeddings is computationally expensive, we instead work with a compact approximation based on candidate vertices, which over-approximates the set of valid embeddings. We adopt the candidate space representation $CS(S)$ [16] for this purpose.

We first introduce the candidate space representation and the candidate verification primitive, then present the progressive evaluation framework, followed by orbit-level reductions and optimization, and finally describe the complete evaluation algorithm.

Candidate Space Representation. We adopt the candidate space representation $CS(S)$ [16], which stores candidate data vertices for each pattern vertex together with the structural relations induced by the pattern edges. The candidate space over-approximates the set of valid embeddings of S in G. For each pattern vertex u, the candidate space maintains two auxiliary state sets: the confirmed candidates C_u and the remaining candidates R_u. Initially $C_u = \emptyset$ and $R_u = D_u$, where D_u denotes the candidate domain of u in $CS(S)$. If S consists of a single edge, all candidates are trivially verified, i.e., $C_u = D_u$ and $R_u = \emptyset$.

Let $\mathrm{Img}(u; S)$ denote the true image set of u in embeddings of S in G. The candidate state maintains the invariant

$$C_u \subseteq \mathrm{Img}(u; S) \subseteq C_u \cup R_u.$$

Candidate Verification. Candidate verification progressively refines this approximation. To verify a candidate $v \in R_u$, we perform a DAF-based backtracking search with failing sets [13] to find an embedding of S in G that maps u to v. The search stops after the first embedding and prefers embeddings that cover unverified candidates so that multiple nodes can be validated simultaneously. If such an embedding exists, the verification succeeds. The discovered embedding is then used to update candidate states of related pattern vertices, and the updates are propagated to vertices in the same automorphism orbit. Otherwise, v is removed from R_u.

Progressive Evaluation Framework. Computing $\mathsf{MDS}(S)$ directly would require examining all subsets of pattern vertices and their image-set overlaps. Instead we evaluate MDS progressively using bounds. The framework maintains

$$LB(S) \le \mathsf{MDS}(S) \le UB(S)$$

where

$$LB(S) = \min_{U \neq \emptyset} \frac{|\cup_{u \in U} C_u|}{|U|}$$

$$UB(S) = \min_{U \neq \emptyset} \frac{|\cup_{u \in U} (C_u \cup R_u)|}{|U|}.$$

The lower bound assumes that only confirmed candidates participate in embeddings, while the upper bound assumes that remaining candidates may still participate. Candidate verification refines these sets incrementally: successful verification moves a candidate from R_u to C_u, increasing $LB(S)$, while unsuccessful verification removes it from R_u, decreasing $UB(S)$. Evaluation terminates once either (1) $LB(S) = UB(S)$, yielding the exact MDS value, or (2) $UB(S) \le \tau$, where τ is the current Top-k threshold.

Orbit-Level Reduction. Although the progressive framework avoids explicit embedding enumeration, evaluating bounds at the vertex level can still be expensive. We therefore lift the evaluation domain from vertices to automorphism orbits of the pattern.

Let $O_1, \ldots, O_m$ denote the automorphism orbits of S. By Theorem 5 and Theorem 6, density minimization can be restricted to subsets of entire orbits, reducing the search space from vertices to a smaller orbit space.

Bounds are then computed over orbit subsets. For each orbit O_i, its candidate domain $D(O_i)$ is the intersection of the candidate domains of its vertices, $C(O_i)$ contains candidates confirmed for its vertices, and $R(O_i) = D(O_i) \setminus C(O_i)$.

$$LB(S) = \min_{U \neq \emptyset} \frac{|\cup_{O_i \in U} C(O_i)|}{\sum_{O_i \in U} |O_i|}$$

$$UB(S) = \min_{U \neq \emptyset} \frac{|\cup_{O_i \in U} (C(O_i) \cup R(O_i))|}{\sum_{O_i \in U} |O_i|}.$$

As a result, the enumeration cost decreases from $O(2^{|V_S|})$ at the vertex level to $O(2^m)$ at the orbit level.

Orbit-Aware Optimization. A remaining bottleneck in MDS evaluation is the computation of coverage densities over orbit subsets. Although orbit-level reduction significantly decreases the search space, enumerating all orbit subsets may still be expensive when the pattern contains multiple orbits.

To reduce the cost of bound computation, we leverage the separability property established in Theorem 7. Specifically, we construct an *orbit intersection graph*, where two orbits are connected if their candidate image sets overlap. Let $P_1, \ldots, P_\ell$ denote the connected components of this graph. Corollary 8 implies that the density minimization can be performed independently within each component P_j, rather than over the entire orbit set. The evaluation algorithm in Algorithm 1 implements this strategy. For each component P_j, the algorithm enumerates only the orbit subsets U contained in that component and evaluates their coverage densities

$$LB(P_j) = \min_{U \neq \emptyset \wedge U \subseteq P_j} \frac{|\cup_{O_i \in U} C(O_i)|}{\sum_{O_i \in U} |O_i|}$$

$$UB(P_j) = \min_{U \neq \emptyset \wedge U \subseteq P_j} \frac{|\cup_{O_i \in U} (C(O_i) \cup R(O_i))|}{\sum_{O_i \in U} |O_i|}.$$

Moreover, once the density drops below the current Top-k threshold τ, the enumeration terminates early, since no further subset can affect the result. As a result, the enumeration cost is reduced to $O(\ell \cdot 2^{m'})$, where m' is the number of orbits in the largest component.

Evaluation Algorithm. The evaluation of MDS is performed in two stages. Algorithm 1 first invokes the auxiliary procedure InitializeBounds to compute initial bounds based on the current candidate states, and then invokes RefineBounds to progressively tighten these bounds through candidate verification.

Initialization. The procedure InitializeBounds first computes the automorphism orbits $\{O_1, \ldots, O_m\}$ of the pattern S and derives the corresponding orbit-level candidate domains $D(O_i), C(O_i), R(O_i)$. The orbit intersection graph $\mathcal{G}$ is then constructed and decomposed

into connected groups $P_1, \ldots, P_\ell$. Based on these structures, the initial bounds $LB(O_i), UB(O_i)$ for each orbit and $LB(P_j), UB(P_j)$ for each group are computed. The pattern-level bounds LB and UB are finally derived from the group bounds.

Iterative Verification. The procedure RefineBounds progressively refines the MDS value through candidate verification. At each iteration, the algorithm selects a group P_s and an orbit $O_b \in P_s$ whose bounds are most critical to the current pattern bounds. The selection is guided by the verification success rate ρ, defined as the ratio of successful verifications to the total number of verifications. When ρ is low (e.g., below 50%), a *Lower-UB strategy* is adopted, selecting the group with the smallest upper bound and the orbit with the smallest upper bound within that group. Otherwise, an *Upper-LB strategy* is used, selecting the group with the smallest lower bound and the orbit with the smallest lower bound.

A candidate vertex is then selected from $R(O_b)$, preferring vertices that lie in intersections of candidate domains of multiple orbits. The selected candidate is verified using a subgraph matching procedure. If verification fails, the vertex is removed from $R(O_b)$; otherwise, the discovered embedding is used to update candidate states of related pattern vertices. Consequently, the sets $C(O_i)$ and $R(O_i)$ are updated for affected orbits, and the corresponding orbit and group bounds are recomputed. If candidate-domain intersections change, the orbit intersection graph $\mathcal{G}$ is updated accordingly.

The process terminates when the pattern becomes prunable ($UB \le \tau$) or the exact MDS value is determined ($LB = UB$), corresponding to the stopping condition of the loop. The algorithm then returns the exact MDS value or reports pruning.

Selection Strategy Discussion. The adaptive selection strategy aims to quickly narrow the gap between the bounds. Failed verifications tend to reduce the upper bound, whereas successful verifications increase the lower bound. When the success rate is low, the algorithm prioritizes candidates that may reduce the upper bound; when it is high, it prioritizes candidates that may increase the lower bound.

Example 3: Figure 2 illustrates the MDS evaluation process. The pattern S contains four vertices with labels A-A-B-C. Due to the automorphism $u_1 \leftrightarrow u_2$, the pattern vertices are partitioned into three automorphism orbits: $O_1 = \{u_1, u_2\}$, $O_2 = \{u_3\}$, and $O_3 = \{u_4\}$. Thus the enumeration space is reduced from $2^4 - 1 = 15$ vertex subsets to $2^3 - 1 = 7$ orbit subsets. Moreover, since the three orbits have distinct labels, their candidate domains are disjoint. Consequently, the orbit intersection graph contains three singleton groups, each consisting of a single orbit. The initial bounds are $LB = 0$ and $UB = \min\{5/2,\ 2/1,\ 3/1\} = 2$.

The algorithm then iteratively verifies candidates. In the first round, the success rate $\rho = 0$, so the *Lower-UB strategy* is used. The group with the smallest upper bound is selected, and orbit O_1 is chosen. The candidate $v_1 \in R(O_1)$ is verified (Figure 2(d)). Since no embedding maps O_1 to v_1, the verification fails and v_1 is removed from $R(O_1)$. In the second round, the success rate remains $\rho = 0$, so the Lower-UB strategy is again applied. Orbit O_2 is selected and candidate $v_5 \in R(O_2)$ is verified (Figure 2(e)). This verification succeeds and the discovered embeddings update the candidate states accordingly. In the third round, the success rate becomes $\rho = 1/2$, and the algorithm switches to the *Upper-LB strategy*. Orbit O_1 is selected and candidate $v_6 \in R(O_1)$ is verified (Figure 2(f)). This verification succeeds, causing the lower and upper bounds to converge to 2. The algorithm terminates and returns $\mathrm{MDS}(S) = 2$. □

Algorithm 1: EvaluateMDS

Input: Pattern S, data graph G, candidate space $CS(S)$, and threshold τ.
Output: $\mathrm{MDS}(S)$ or -1 if S is pruned.

1. InitializeBounds $(S, CS(S))$;
2. RefineBounds (S, G, τ);

Procedure InitializeBounds(S, CS(S))

1. $\{O_1, \ldots, O_m\} \leftarrow$ automorphism orbits of S;
2. $D(O_i) \leftarrow \bigcap_{v \in O_i} D_v$, for all i;
3. $C(O_i) \leftarrow \emptyset$; $R(O_i) \leftarrow D(O_i)$, for all i;
4. construct orbit intersection graph $\mathcal{G}$ over $\{O_i\}$;
5. $\{P_1, \ldots, P_\ell\} \leftarrow$ connected components of $\mathcal{G}$;
6. **for each** orbit O_i **do**
7. $\quad LB(O_i) \leftarrow |C(O_i)|/|O_i|$; $UB(O_i) \leftarrow |D(O_i)|/|O_i|$;
8. **for each** group P_j **do**
9. $\quad$ update bounds $LB(P_j)$ and $UB(P_j)$ from all orbit subsets;
10. $LB \leftarrow 0$; $UB \leftarrow \min_{P_j} UB(P_j)$;
11. **return** LB, UB;

Procedure RefineBounds(S, G, τ)

1. $\rho \leftarrow 0$; $succ \leftarrow 0$; $tot \leftarrow 0$;
2. **repeat**
3. $\quad$ **if** $\rho < 0.5$ **then**
4. $\quad\quad P_s \leftarrow \arg\min_{P_j} UB(P_j)$; $O_b \leftarrow \arg\min_{O_i \in P_s} UB(O_i)$;
5. $\quad$ **else**
6. $\quad\quad P_s \leftarrow \arg\min_{P_j} LB(P_j)$; $O_b \leftarrow \arg\min_{O_i \in P_s} LB(O_i)$;
7. $\quad$ pick $v \in R(O_b)$ (prefer intersection candidates);
8. $\quad$ **if** VerifyCandidate (S, O_b, v, G) **then**
9. $\quad\quad succ \leftarrow succ + 1$;
10. $\quad\quad$ update candidate states using discovered embedding;
11. $\quad$ **else** $R(O_b) \leftarrow R(O_b) \setminus \{v\}$;
12. $\quad tot \leftarrow tot + 1$; $\rho \leftarrow succ/tot$;
13. $\quad$ update affected $C(O), R(O), LB, UB$ and intersection graph $\mathcal{G}$;
14. **until** $UB \le \tau$ **or** $LB = UB$;
15. **if** $UB \le \tau$ **then return** -1;
16. **else return** LB;

4.2 Mining Algorithm

Instead of enumerating all patterns and filtering afterwards, our algorithm adopts a best-first exploration paradigm driven by MDS support. At every step, the search focuses on the pattern that has the highest potential to yield a large MDS value, so high-quality patterns are discovered early and immediately tighten the Top-k threshold, enabling aggressive pruning of the remaining space.

Best-First Top-k Search. To realize this strategy, we maintain two priority structures. An *answer heap* H_{ans} stores the current Top-k patterns ranked by their *exact* MDS scores. A *candidate heap* H_{cand} stores unexplored patterns ordered by their MDS *upper bounds*, representing the maximum support they can still achieve. A dynamic threshold τ is defined as the k-th largest MDS value in H_{ans} and increases monotonically as better patterns are discovered.

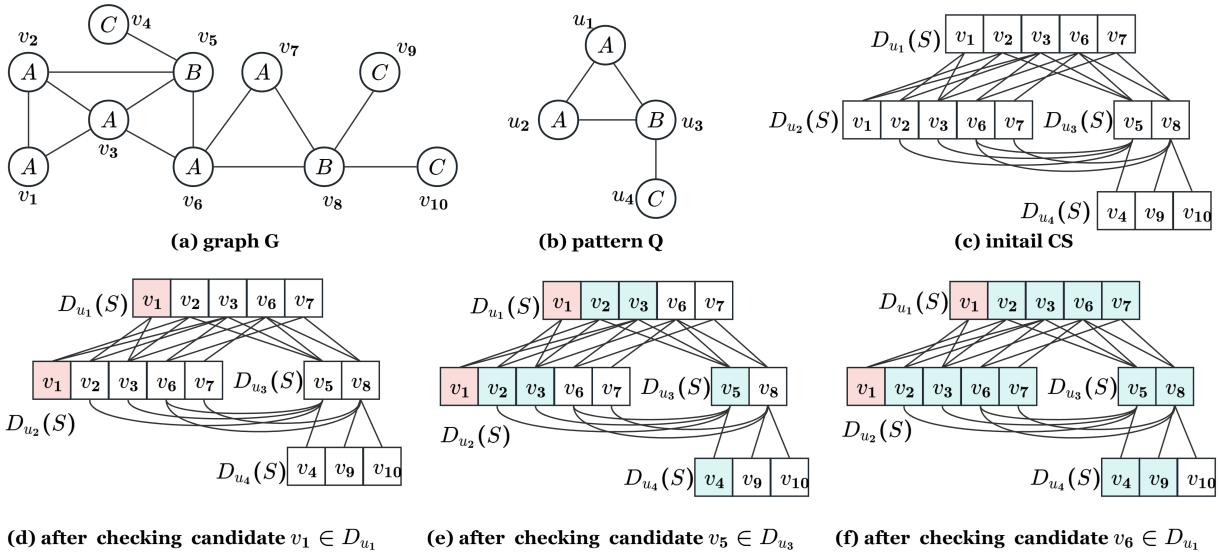


Figure 2: MDS Evaluation Process Illustration.

The search is initialized using all distinct edge patterns in G. For each edge pattern e, its exact MDS score is computed and the Top-k edge patterns are inserted into H_{ans}. The initial threshold τ is then set to the k-th largest MDS value in H_{ans}. All canonical one-edge extensions of these edge patterns whose upper bounds exceed τ are inserted into H_{cand}.

During the search, the algorithm repeatedly extracts from H_{cand} the pattern with the highest upper bound and evaluates its exact MDS score via RefineBounds. If the score exceeds τ, the pattern is inserted into H_{ans}, τ is updated, and its canonical one-edge extensions whose upper bounds exceed τ are inserted into H_{cand}. Otherwise the candidate is discarded.

To avoid duplicate generation, we follow the canonical DFS-code ordering used in frequent subgraph mining (e.g., gSpan), where only extensions that preserve the canonical DFS code of a pattern are explored, ensuring that each pattern is generated exactly once.

Since MDS support satisfies anti-monotonicity and the upper bound applies to all descendants of a pattern, pruning remains sound regardless of exploration order. Therefore, best-first exploration preserves completeness: any pattern whose MDS could enter the Top-k result will eventually be evaluated unless safely pruned by the bound. The algorithm terminates when H_{cand} becomes empty or its largest upper bound is no greater than τ, since no remaining pattern can enter the Top-k result.

The complete procedure is summarized in Algorithm 2, with the auxiliary procedure ExpandPattern shown below the main routine.

Algorithm 2: MDS − FSM

Input: Data graph G and an integer k.
Output: Top-k patterns H_{ans} with the highest MDS scores.

1. $\mathcal{E} \leftarrow$ all edge patterns in G together with their candidate spaces;
2. **for each** $e \in \mathcal{E}$ **do**
3. $\quad O \leftarrow$ automorphism orbits of e;
4. $\quad mds_e \leftarrow \left\lfloor \min_{\emptyset \neq U \subseteq O} \frac{|\bigcup_{o \in U} \bigcup_{v \in o} C_v|}{\sum_{o \in U} |o|} \right\rfloor$;
5. $H_{ans} \leftarrow$ top-k from $\mathcal{E}$ by mds; $\tau \leftarrow$ the k-th score in H_{ans};
6. $H_{cand} \leftarrow \emptyset$;
7. **for each** $e \in \mathcal{E}$ with $mds_e \geq \tau$ **do**
8. $\quad \mathcal{P} \leftarrow$ CanonicalExtensions(e); ExpandPattern $(\mathcal{P})$;
9. **while** $H_{cand} \neq \emptyset$ **and** H_{cand}.top.$UB > \tau$ **do**
10. $\quad S \leftarrow H_{cand}$.pop();
11. $\quad mds \leftarrow$ RefineBounds(S, G, τ);
12. $\quad$ **if** $mds > \tau$ **then**
13. $\quad\quad$ insert S into H_{ans}; update τ;
14. $\quad\quad \mathcal{P} \leftarrow$ CanonicalExtensions(S); ExpandPattern $(\mathcal{P})$;
15. **return** H_{ans};

Procedure ExpandPattern$(\mathcal{P})$

1. **for each** $S \in \mathcal{P}$ **do**
2. $\quad$ InitializeBounds $(S, CS(S))$;
3. $\quad$ **if** $UB(S) > \tau$ **then** insert S into H_{cand};
4. **return** H_{cand};

4.3 Correctness and Complexity

Correctness Analysis. We justify the correctness of the proposed framework from two aspects: (i) the MDS values computed by the evaluation procedure are exact (soundness), and (ii) the mining algorithm does not miss any true Top-k pattern (completeness).

Soundness. During evaluation, each candidate domain D_u is split into confirmed candidates C_u and remaining candidates R_u, maintaining $C_u \subseteq \mathrm{Img}(u; S) \subseteq C_u \cup R_u$. The lower and upper bounds are computed from C_u and $C_u \cup R_u$, respectively, thus ensuring $LB \leq \mathrm{MDS}(S) \leq UB$. Furthermore, Theorem 6 shows that the minimum density is achieved by complete orbit subsets, and Corollary 8 ensures that density minimization can be performed independently within orbit-intersection components. Hence the orbit-level and component-wise evaluation computes the exact MDS value.

Completeness. Completeness follows from the anti-monotonicity of MDS (Theorem 1). If $UB(S) \leq \tau$, then every superpattern of S has MDS $\leq \tau$, making pruning safe. Our mining algorithm performs a best-first search in decreasing order of upper bounds. Therefore any pattern with $\mathrm{MDS}(S) > \tau$ will be evaluated before the threshold drops below τ, ensuring that no Top-k pattern is missed.

Complexity Analysis. Let $n = |V_S|$ be the pattern size, m the number of automorphism orbits of S, and m' the number of orbits in the largest orbit-intersection component. Let $D = \sum_u |D_u|$ denote the total candidate size over all pattern vertices. In the initialization stage, computing automorphism orbits takes $O(n^2)$ time, and constructing candidate domains via label and neighborhood filtering costs $O(|V_G| + |E_G|)$. Building the orbit intersection graph requires checking intersections between orbit domains and costs $O(m^2 D)$ in the worst case. Computing the initial orbit and group bounds requires enumerating orbit subsets within each component and costs $O(\ell \cdot 2^{m'})$. In the iterative evaluation stage, each candidate is verified at most once, yielding at most D verifications, each performed via subgraph matching with one pattern vertex fixed and costing at most C_{match}. After each verification, candidate states, orbit intersections, and affected bounds are updated in $O(mD+2^{m'})$ time. The overall cost of evaluating a single pattern is therefore $O\big(|V_G| + |E_G| + D(C_{\text{match}} + mD + 2^{m'})\big)$.

5 EXPERIMENTAL EVALUATION

Using real-life and synthetic datasets, we evaluate (1) the effectiveness of MDS in reducing overlap, (2) the efficiency and scalability of MDS-FSM, (3) an ablation study on MDS-FSM, and (4) a case study on biological networks.

Experimental Setting. We start with the experimental setting.

Table 1: Dataset Statistics.

Dataset	Type	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert\Sigma\rvert$
Yeast	PPI	3,112	12,519	71
Human	PPI	4,674	86,282	44
WCGoals	Event	49,078	158,114	11
Mico	Social	100,000	1,080,298	29
DBLP	Coauthor	317,080	1,049,866	15
Patent	Citation	2,745,761	13,965,409	37

Datasets. We used six real-life datasets, as summarized in Table 1. Yeast [39] and Human [27] are clustered protein–protein interaction networks; WCGoals [44] and Mico [6] are social/event graphs dominated by chain and star patterns; DBLP [20] and Patent [12] are co-authorship and citation graphs, respectively.

For synthetic datasets, we generated synthetic graphs using NetworkX [11] with three topology models: Barabási-Albert (BA, $m = 5$) [3] for scale-free networks via preferential attachment, Erdős-Rényi (ER) [7] for random graphs, and Watts-Strogatz (WS, $k = 10, \beta = 0.1$) [38] for small-world networks. All synthetic graphs had average degree $\bar{d} \approx 10$, and vertex labels were sampled from $|\Sigma| = 30$ labels following a power-law distribution ($\alpha = 0.8$) to simulate realistic skewed label frequencies. Under this common setting, we additionally instantiated smaller 1K-node BA/ER/WS graphs for exact-MIS validation, and generated larger graphs with $|V|$ ranging from 10K to 5M for scalability evaluation.

Baselines. We compare MDS against representative support measures and mining systems. For support, we evaluate (1) MNI [5] as the mainstream single-graph support measure, (2) FS [43] as an image-based refinement that distributes fractional weights among overlapping embeddings to mitigate vertex reuse, and (3) MIS [35] as a representative reference based on independent embeddings. For systems, we compare MDS-FSM with three widely established FSM engines. (1) GraMi [6] serves as the representative classical single-graph FSM engine and has been widely adopted as a standard reference system in the literature. (2) T-FSM [43] represents the parallelization direction of single-graph FSM and provides a strong engineering baseline for scalable mining; we evaluate both T-FSM-MNI and T-FSM-frac to cover its two officially supported configurations. (3) Minting [19] represents the Top-k best-first mining paradigm. (4) MIS-Greedy [8] serves as a practical heuristic baseline that estimates MIS-based coverage by greedily selecting non-overlapping embeddings. Minting and MDS-FSM both natively support Top-k mining and are compared under identical k settings. GraMi and T-FSM are threshold-based systems; to ensure fair comparison, we use the smallest support threshold that yields at least k frequent patterns. Mining is then performed under this threshold to enumerate all frequent patterns, which are finally sorted by support to retain the Top-k results.

Evaluation Metrics. For MDS effectiveness, we examine how closely a support measure aligns with independent embedding coverage. Since computing exact MIS is computationally expensive on large graphs, we use AIC, computed via a greedy heuristic, as a practical approximation to MIS. Prior work [8] has shown that such heuristics often yield values close to exact MIS in practice. The metric, denoted as *OER* (under AIC), is then defined as

$$\mathrm{OER}(S) = \frac{\sigma(S) - \mathrm{AIC}(S)}{\sigma(S)}$$

where $\sigma(S)$ is the support value returned by the support measure for pattern S. OER represents the signed relative deviation of the support value from the AIC reference: positive values indicate overestimation, while negative values indicate underestimation.

For efficiency, we report the total runtime of all algorithms.

Implementation. MDS-FSM and MIS-Greedy are implemented in C++, while Minting, T-FSM, and GraMi are evaluated using their official C++ implementations (GraMi from the T-FSM artifact). All experiments run on a Linux server with dual Intel Xeon Gold 6142 CPUs (32 cores, 2.60GHz) and 512GB RAM. Each configuration is run three times and the results are averaged. The timeout for each run is set to 2000 seconds.

Experimental Results. We next report our findings.

Exp-1: Effectiveness of MDS. We evaluate MDS in terms of support accuracy, structural impartiality, and downstream utility, with Top-k set to 50 by default.

Support Accuracy. Since exact MIS computation is infeasible on large graphs, we evaluate support accuracy on small synthetic graphs (1K nodes) using three standard models, BA, ER, and WS,

Table 2: OER under AIC and Exact MIS on Synthetic.

Method	BA		ER		WS	
	AIC	MIS	AIC	MIS	AIC	MIS
MNI	56.38%	56.38%	69.00%	68.90%	51.32%	51.27%
FS	53.60%	53.45%	69.85%	69.57%	30.68%	30.61%
MDS	**21.42%**	**21.42%**	**12.48%**	**12.41%**	**15.89%**	**15.84%**

where exact MIS can be computed. For the Top-20 patterns, we report OER under both AIC and exact MIS, and summarize the results in Table 2. (1) Across all graph families, the OER values computed under AIC and under exact MIS are nearly identical for all support measures. This provides empirical evidence that AIC is a reliable practical proxy for MIS. (2) Under both references, MDS consistently yields substantially smaller OER than MNI and FS, indicating that it aligns more closely with independent-realization semantics. Under exact MIS, the average OER of MDS is about 16.6%, compared to 58.9% for MNI and 51.2% for FS.

Table 3: OER Statistics on Real Datasets.

Dataset	Support	Mean (%)	Std.	P50 (%)	P90 (%)
Yeast	MNI	96.88	4.17	98.27	99.13
Yeast	FS	88.35	13.90	92.01	94.07
Yeast	MDS	46.87	19.86	54.61	62.04
Human	MNI	96.42	1.95	96.26	98.97
Human	FS	94.31	13.55	96.32	97.31
Human	MDS	58.97	30.71	74.79	85.37
Mico	MNI	95.47	6.91	98.69	99.25
Mico	FS	85.78	12.98	93.21	96.97
Mico	MDS	42.78	18.25	50.77	55.50
WCGoals	MNI	18.55	12.19	26.32	27.75
WCGoals	FS	20.49	14.76	14.38	25.03
WCGoals	MDS	9.24	10.33	11.24	26.32
DBLP	MNI	26.19	14.92	19.36	57.95
DBLP	FS	11.01	18.72	-2.80	50.28
DBLP	MDS	19.19	0.44	19.25	19.62
Patent	MNI	14.37	1.28	14.55	15.69
Patent	FS	-5.21	5.73	-5.93	-4.67
Patent	MDS	14.37	1.28	14.55	15.69

Table 3 further reports OER statistics on real datasets. (1) On datasets where image-based supports exhibit severe overestimation (Yeast, Human, and Mico), the mean OER decreases from 96.26% for MNI to 49.54% for MDS on average, and extreme deviations are also reduced (e.g., Yeast P90: 99.13% vs. 62.04%). (2) On WCGoals, where overlap is more localized, the mean OER decreases from 18.55% to 9.24%. (3) On sparse large-scale graphs (DBLP and Patent), the difference becomes smaller: MDS produces a much lower extreme deviation on DBLP (P90 19.62% vs. 57.95% for MNI), while matching MNI on Patent (mean 14.37%). (4) Negative OER values occur only for FS on a few datasets (e.g., DBLP and Patent), as its fractional sharing heuristic may overly penalize reused vertices. Overall, MDS provides the largest benefit when embedding overlap is substantial, while remaining competitive when overlap is sparse.

Structural Impartiality. A desirable support measure should behave consistently across different structural patterns rather than favoring particular topologies in practice. To examine this property, we group mined patterns from six real datasets into five topology families (paths, trees, cycles, cliques, and stars) and compare their OER statistics. As shown in Table 4, MNI and FS exhibit strong structural bias. They severely overestimate highly overlapping structures such as cycles and cliques (OER > 90%), while producing smaller deviations on trees and stars. In contrast, MDS consistently reduces OER across all topology families, with the largest improvements observed on cycles and cliques (−31.7% and −43.4%). Even on paths and stars, where image-based supports are relatively stable, MDS still yields noticeable reductions. Moreover, MDS achieves much lower cross-topology OER variance (8.92 vs. 14.72 for MNI), indicating more uniform evaluation behavior. Overall, these results show that MDS substantially improves structural impartiality across diverse graph patterns.

Downstream Utility. We evaluate whether the mined patterns improve downstream query processing by using the Top-k patterns as indexes for query verification. On each dataset, we generate 30 query graphs (3–6 edges) and vary k from 10 to 50. All methods use the same query workload and candidate generation procedure. We report the *candidate pruning ratio*, i.e., the fraction of query candidates eliminated before verification, together with the overall average end-to-end query time.

Figure 3 reports the results. (1) MDS achieves the highest pruning ratio. Averaged over the six datasets and all k values, its average pruning ratio is 0.3604, outperforming MNI (0.2517) and FS (0.1779) by 43.2% and 102.6%, respectively. (2) The stronger pruning also reduces the end-to-end query time. MDS lowers the average end-to-end query time to 441.5 s, compared with 503.1 s for MNI and 530.3 s for FS, i.e., reductions of 12.2% and 16.7%. (3) The gains are largest on the PPI networks (Yeast, Human), where embedding overlap is substantial, while the bibliographic graphs (DBLP, Patent) show smaller but consistent improvements.

Exp-2: Efficiency of MDS-FSM. We compare the runtime of MDS-FSM with existing FSM systems by varying k from 10 to 50; the results are shown in Figure 4. MIS-Greedy is included only as an accuracy reference and becomes impractical even at small k. (1) On dense-overlap graphs (Yeast, Human, Mico), MDS-FSM is substantially faster. For example, on Yeast at $k = 50$, MDS-FSM finishes in about 1.1s while GraMi and T-FSM require around 10^3s. On Human, MDS-FSM runs in about 1s compared with 10^2–10^3s for image-based FSM systems. (2) On sparse graphs (DBLP, Patent),

Table 4: Per-Topology Average OER Comparison (%).

Topology	MNI	FS	MDS	Improvement
Path	74.4	79.7	48.4	-26.0
Tree	60.8	69.6	55.9	-4.9
Cycle	97.5	94.6	65.8	-31.7
Clique	92.0	84.1	48.6	-43.4
Star	63.9	63.6	39.0	-24.9
Cross-Topology Std.	14.72	10.89	**8.92**	-39.4

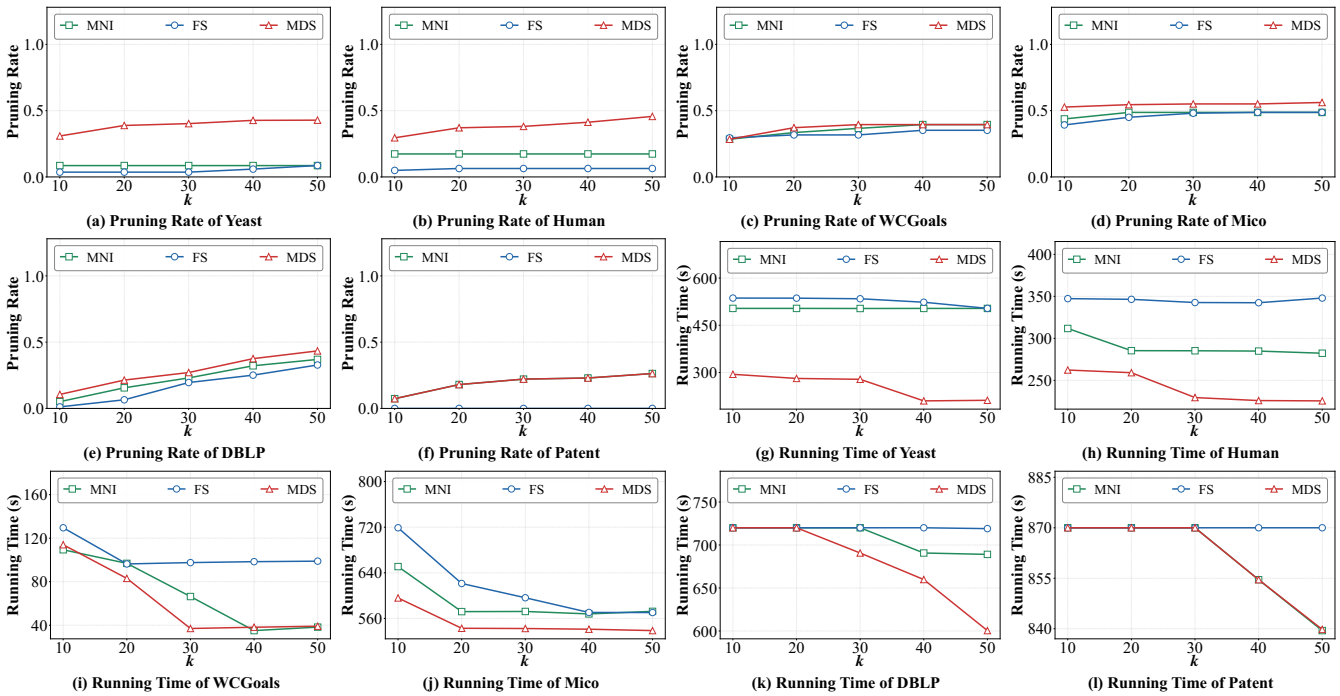


Figure 3: Downstream Performance.

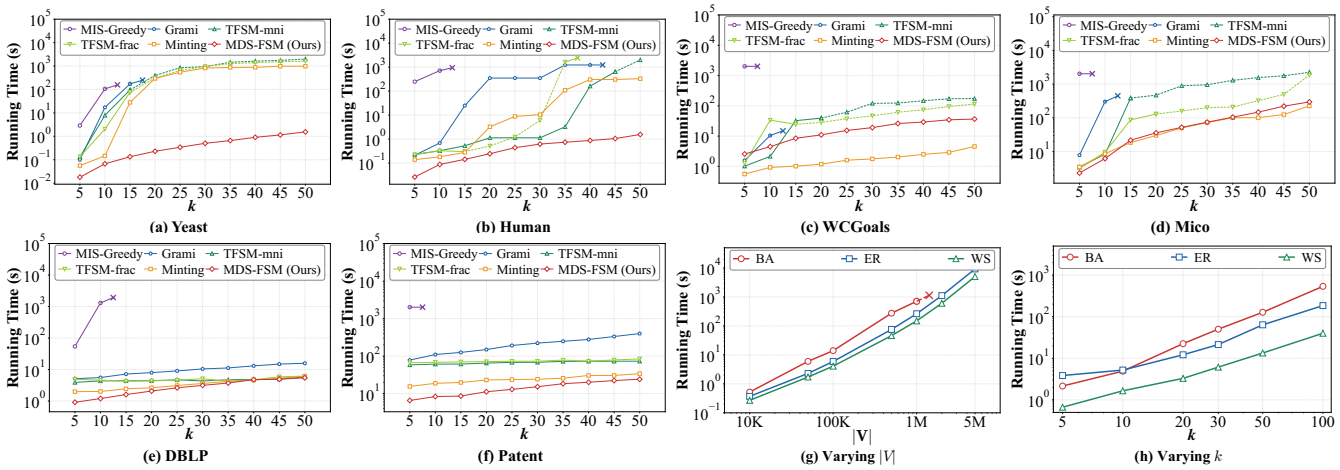


Figure 4: Efficiency Performance.

the gap becomes smaller but MDS-FSM remains consistently competitive. For instance, on DBLP at $k = 50$, MDS-FSM takes about 5s versus roughly 15s for GraMi. (3) WCGoals is the only dataset where Minting is faster (about 5s vs. 32s for MDS-FSM), which is consistent with its star-dominated structure that is already well captured by image-based supports. Overall, MDS-FSM scales smoothly with k and avoids the runtime escalation observed in image-based FSM approaches in practice.

Exp-3: Scalability of MDS-FSM. We further evaluate scalability on synthetic graphs under skewed label distributions.

Varying $|V|$. Figure 4(g) reports runtime as $|V|$ increases from 10K to 5M. Runtime increases consistently as graph size grows. ER and WS graphs complete successfully even at 5M vertices, finishing in about 10^4s. BA graphs are consistently more expensive due to their hub-dominated topology, which produces larger candidate sets.

Varying k. Figure 4(h) evaluates sensitivity to k by fixing $|V| = 100$K and varying k from 5 to 100. Runtime increases with k and reaches about 254s at $k = 100$, showing that the system remains practical even when more patterns are required.

Ablation Study. We conduct ablations to evaluate the impact of the efficiency optimizations (orbit decomposition and upper-bound pruning) and the minimum global density formulation of MDS.

Table 5: Efficiency Ablation Results.

Dataset	Time (s)				
	MDS-FSM	SFM	NoSep	Vertex	NoLB
Yeast	0.42	21.81	0.54	0.62	0.75
Human	0.74	31.00	1.01	1.65	1.91
Mico	175.67	1524.91	253.82	279.61	669.76
WCGoals	32.50	723.36	41.25	155.30	1312.23
DBLP	3.14	3.21	3.14	3.18	3.13
Patent	27.39	27.53	27.05	27.31	27.67

Evaluation Strategy Study. We evaluate the contribution of each component using five variants: MDS-FSM (full), SFM (direct solver), NoSep (without separability), Vertex (vertex-level evaluation), and NoLB (without lower-bound tightening). Table 5 reports the results. Overall, MDS-FSM achieves the best performance. Replacing bound-based evaluation with SFM increases runtime by about 9.7 times on average, suggesting that the polynomial-time formulation is computationally expensive in practice without optimization. Removing decomposition (NoSep and Vertex) incurs moderate overhead (36.3% and 95.0% on average), indicating that separability and orbit-level reduction reduce redundant evaluation. In contrast, disabling lower-bound tightening (NoLB) leads to the largest slowdown (about 8.4 times), as it weakens pruning during mining and forces many patterns to be fully evaluated.

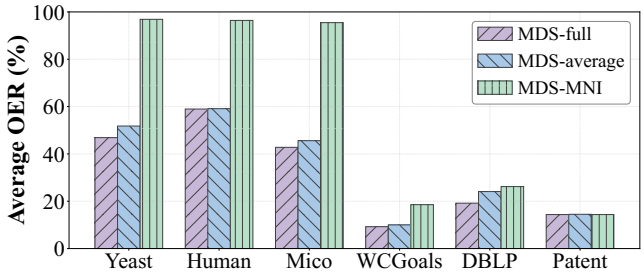


Figure 5: Global Density Minimization Ablation.

MDS Variants. We compare three variants: MDS-full (minimum density), MDS-average (average density), and MDS-MNI (equivalent to MNI). Figure 5 reports average OER. (1) Across all datasets, MDS-MNI yields the largest OER, while both MDS-full and MDS-average substantially reduce it, especially on high-overlap graphs (Yeast, Human, Mico). (2) Among the two global variants, MDS-full consistently achieves the lowest OER; for example, on DBLP it reduces OER to about 19% compared with 24% for MDS-average and 26% for MDS-MNI. Overall, global density reasoning is essential for accurate support estimation, and worst-case minimization provides the most stable results.

Exp-4: Case Study. To illustrate the qualitative effect of different support measures, we analyze representative Top-k patterns discovered on the Yeast PPI dataset. Figure 6 reports the Top-5 patterns with their support values. Under MNI and FS, the patterns are almost identical and are dominated by homogeneous backbone motifs with label "2–2", which receive very large support values due to heavy embedding overlap. Under MDS, these backbone motifs obtain much smaller supports and no longer dominate the ranking. Instead, MDS surfaces structurally different motifs (e.g., involving labels "0–2" and "2–3") that do not appear in the results under MNI and FS. Overall, by mitigating overlap inflation, MDS changes the ranking of frequent patterns and reveals additional motifs that would otherwise be suppressed by backbone structures.

Rank	MNI	FS	MDS
1	support = 529	support = 302	support = 264
2	support = 264	support = 267	support = 191
3	support = 393	support = 262	support = 183
4	support = 373	support = 261	support = 176
5	support = 368	support = 241	support = 172

Figure 6: Representative Top-k Patterns on Yeast.

Summary. MDS substantially improves both support behavior and mining outcomes. Across datasets, MDS (1) reduces OER by up to 55% (37% on average), with absolute reductions of up to 52.7% points on densely overlapping graphs; (2) lowers cross-topology OER variation from 14.7% to 8.9% (a 39.4% reduction), indicating much more stable structural evaluation; (3) significantly improves downstream graph-index query performance, achieving 43.2% higher average pruning ratio and 12.2% lower average end-to-end query time than the strongest baseline; and (4) scales smoothly to graphs with millions of nodes while maintaining competitive runtime.

6 CONCLUSION

We propose Minimum Density Support (MDS), a coverage-based frequency measure for single-graph FSM based on global subset density minimization. MDS is anti-monotone and provably bounded between MIS and MNI, offering a tractable alternative to FSM. Leveraging orbit-block optimality and subset separability, we design an efficient MDS-FSM algorithm. Experiments on six real-world graphs show that MDS reduces overestimation by 37% on average, decreases cross-topology OER variance by 39%, and improves downstream graph-index query performance, while maintaining competitive efficiency and scaling to million-node graphs. Future work includes extending MDS to dynamic or streaming graphs and integrating it with approximate frameworks for scalability.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (Grant No.62572345, 62272311 and 61902274).

REFERENCES

[1] Ehab Abdelhamid, Ibrahim Abdelaziz, Panos Kalnis, Zuhair Khayyat, and Fuad Jamour. 2016. ScaleMine: Scalable Parallel Frequent Subgraph Mining in a Single Large Graph. In *SC16: International Conference for High Performance Computing, Networking, Storage and Analysis*. 716–727. https://doi.org/10.1109/SC.2016.60

[2] Rakesh Agrawal and Ramakrishnan Srikant. 1994. Fast Algorithms for Mining Association Rules in Large Databases. In *Proc. Int. Conf. on Very Large Data Bases (VLDB)*. 487–499.

[3] Albert-László Barabási and Réka Albert. 1999. Emergence of Scaling in Random Networks. *Science* 286, 5439 (1999), 509–512.

[4] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*. 2787–2795.

[5] Björn Bringmann and Siegfried Nijssen. 2008. What Is Frequent in a Single Graph?. In *Proc. Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD)*. 858–863.

[6] Mohammed Elseidy, Ehab Abdelhamid, Spiros Skiadopoulos, and Panos Kalnis. 2014. GRAMI: Frequent Subgraph and Pattern Mining in a Single Large Graph. *Proc. VLDB Endow.* 7, 7 (2014), 517–528.

[7] Paul Erdős and Alfréd Rényi. 1959. On Random Graphs I. *Publicationes Mathematicae Debrecen* 6 (1959), 290–297.

[8] Mathias Fiedler and Christian Borgelt. 2007. Support Computation for Mining Frequent Subgraphs in a Single Graph. In *Proc. Int. Workshop on Mining and Learning with Graphs (MLG)*. https://dblp.org/rec/conf/mlg/FiedlerB07

[9] Michael R. Garey and David S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.

[10] Chris Godsil and Gordon F. Royle. 2001. *Algebraic Graph Theory*. Graduate Texts in Mathematics, Vol. 207. Springer.

[11] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. 2008. Exploring Network Structure, Dynamics, and Function using NetworkX. In *Proc. Python in Science Conference (SciPy)*. 11–15.

[12] Bronwyn H. Hall, Adam B. Jaffe, and Manuel Trajtenberg. 2001. *The NBER Patent Citation Data File: Lessons, Insights and Methodological Tools*. NBER Working Paper 8498. National Bureau of Economic Research. https://doi.org/10.3386/w8498

[13] 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 *Proc. ACM SIGMOD Int. Conf. on Management of Data*. 1429–1446.

[14] Lawrence B. Holder, Diane J. Cook, and Surnjani Djoko. 1994. Substructure Discovery in the SUBDUE System. In *Proc. AAAI Workshop on Knowledge Discovery in Databases*. 169–180.

[15] Chuntao Jiang, Frans Coenen, and Michele Zito. 2013. A Survey of Frequent Subgraph Mining Algorithms. *The Knowledge Engineering Review* 28, 1 (2013), 75–105. https://doi.org/10.1017/S0269888912000331

[16] 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 *Proc. ACM SIGMOD Int. Conf. on Management of Data*. 925–937.

[17] Michihiro Kuramochi and George Karypis. 2005. Finding Frequent Patterns in a Large Sparse Graph. *Data Mining and Knowledge Discovery* 11, 3 (2005), 243–271.

[18] Duy Le, Kris Zhao, Mengying Wang, and Yinghui Wu. 2024. GraphLingo: Domain Knowledge Exploration by Synchronizing Knowledge Graphs and Large Language Models. In *Proc. IEEE Int. Conf. on Data Engineering (ICDE)*. 5477–5480. https://doi.org/10.1109/ICDE60146.2024.00432

[19] Seonho Lee, Yeunjun Lee, and Kunsoo Park. 2024. Efficient Top-k Frequent Subgraph Mining Using Tight Upper and Lower Bounds. *Proc. VLDB Endow.* 18, 3 (2024), 557–570. https://doi.org/10.14778/3712221.3712225

[20] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. https://snap.stanford.edu/data.

[21] Jure Leskovec and Rok Sosić. 2016. SNAP: A General-Purpose Network Analysis and Graph-Mining Library. *ACM Trans. Intell. Syst. Technol.* 8, 1 (2016), 1–20.

[22] Brendan D. McKay and Adolfo Piperno. 2014. Practical Graph Isomorphism, II. *Journal of Symbolic Computation* 60 (2014), 94–112.

[23] Jinghan Meng, Napath Pitaksirianan, and Yi-Cheng Tu. 2020. Counting Frequent Patterns in Large Labeled Graphs: A Hypergraph-Based Approach. *Data Mining and Knowledge Discovery* 34, 4 (2020), 980–1021. https://doi.org/10.1007/s10618-020-00686-9

[24] Jinghan Meng and Yi-Cheng Tu. 2017. Flexible and Feasible Support Measures for Mining Frequent Patterns in Large Labeled Graphs. In *Proceedings of the 2017 ACM International Conference on Management of Data (SIGMOD '17)*. ACM, New York, NY, USA, 391–402. https://doi.org/10.1145/3035918.3035936

[25] Aida Mrzic, Pieter Meysman, Wout Bittremieux, et al. 2018. Grasping Frequent Subgraph Mining for Bioinformatics Applications. *BioData Mining* 11, 1 (2018), 20. https://doi.org/10.1186/s13040-018-0181-9

[26] James B. Orlin. 2009. A Faster Strongly Polynomial Time Algorithm for Submodular Function Minimization. *Mathematical Programming* 118, 2 (2009), 237–251. https://doi.org/10.1007/s10107-007-0189-2

[27] T. S. Keshava Prasad, Renu Goel, Kumaran Kandasamy, Shivakumar Keerthikumar, Sameer Kumar, Suresh Mathivanan, Deepthi Telikicherla, Rajesh Raju, Beena Shafreen, Abhilash Venugopal, et al. 2009. Human Protein Reference Database—2009 Update. *Nucleic Acids Research* 37 (2009), D767–D772.

[28] Giulia Preti, Gianmarco De Francisci Morales, and Matteo Riondato. 2023. MaNIACS: Approximate Mining of Frequent Subgraph Patterns through Sampling. *ACM Trans. Intell. Syst. Technol.* 14, 3 (2023), 1–29. https://doi.org/10.1145/3587254

[29] Fengcai Qiao, Xin Zhang, Pei Li, Zhaoyun Ding, Shanshan Jia, and Hui Wang. 2018. A Parallel Approach for Frequent Subgraph Mining in a Single Large Graph Using Spark. *Applied Sciences* 8, 2 (2018), 230. https://doi.org/10.3390/app8020230

[30] Saif Ur Rehman, Asmat Ullah Khan, and Simon Fong. 2012. Graph Mining: A Survey of Graph Mining Techniques. In *Proc. Int. Conf. on Digital Information Management (ICDIM)*. 88–92. https://doi.org/10.1109/ICDIM.2012.6360146

[31] Roded Sharan and Trey Ideker. 2006. Modeling Cellular Machinery through Biological Network Comparison. *Nature Biotechnology* 24, 4 (2006), 427–433.

[32] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. YAGO: A Core of Semantic Knowledge Unifying WordNet and Wikipedia. In *Proc. Int. World Wide Web Conf. (WWW)*. 697–706.

[33] Sheng Tian, Xintan Zeng, Yifei Hu, Baokun Wang, Yongchao Liu, Yue Jin, Changhua Meng, Chuntao Hong, Tianyi Zhang, and Weiqiang Wang. 2024. GraphRPM: Risk Pattern Mining on Industrial Large Attributed Graphs. In *Proc. Joint European Conf. on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD)*. Springer, 133–149.

[34] Julian R. Ullmann. 1976. An Algorithm for Subgraph Isomorphism. *J. ACM* 23, 1 (1976), 31–42.

[35] Natalia Vanetik, Ehud Gudes, and Solomon Eyal Shimony. 2002. Computing Frequent Graph Patterns from Semistructured Data. In *Proc. IEEE Int. Conf. on Data Mining (ICDM)*. 458–465. https://doi.org/10.1109/ICDM.2002.1183988

[36] Natalia Vanetik, Solomon Eyal Shimony, and Ehud Gudes. 2006. Support Measures for Graph Data. *Data Mining and Knowledge Discovery* 13, 2 (2006), 243–260.

[37] Stanley Wasserman and Katherine Faust. 1994. *Social Network Analysis: Methods and Applications*. Cambridge University Press.

[38] Duncan J. Watts and Steven H. Strogatz. 1998. Collective Dynamics of 'Small-World' Networks. *Nature* 393, 6684 (1998), 440–442.

[39] Ioannis Xenarios, Lukasz Salwínski, Xiaoqun Joyce Duan, Patrick Higney, Sul-Min Kim, and David Eisenberg. 2002. DIP, the Database of Interacting Proteins: A Research Tool for Studying Cellular Networks of Protein Interactions. *Nucleic Acids Research* 30, 1 (2002), 303–305.

[40] Bo Yan, Cheng Yang, Chuan Shi, Yong Fang, Qi Li, Yanfang Ye, and Junping Du. 2023. Graph mining for cybersecurity: A survey. *ACM Transactions on Knowledge Discovery from Data* 18, 2 (2023), 1–52.

[41] Yuliang Yan, Yihong Dong, Xianmang He, and Wei Wang. 2015. FSMBUS: A Frequent Subgraph Mining Algorithm in a Single Large-Scale Graph Using Spark. *Journal of Computer Research and Development* 52, 8 (2015), 1768–1783. https://doi.org/10.7544/issn1000-1239.2015.20150256

[42] Rex Ying, Tianyu Fu, Andrew Wang, Jiaxuan You, Yu Wang, and Jure Leskovec. 2024. Representation Learning for Frequent Subgraph Mining. *arXiv preprint arXiv:2402.14367* (2024). https://doi.org/10.48550/arXiv.2402.14367

[43] Lyuheng Yuan, Da Yan, Wenwen Qu, Saugat Adhikari, Jalal Khalil, Cheng Long, and Xiaoling Wang. 2023. T-FSM: A Task-Based System for Massively Parallel Frequent Subgraph Pattern Mining from a Big Graph. *Proc. ACM Manag. Data* 1, 1 (2023), 1–26. https://doi.org/10.1145/3588928

[44] Gensheng Zhang, Damian Jimenez, and Chengkai Li. 2018. Maverick: Discovering Exceptional Facts from Knowledge Graphs. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*. 1317–1332. https://doi.org/10.1145/3183713.3183730