DS-Span: Single-Phase Discriminative Subgraph Mining for Efficient Graph Embeddings

Abstract. Graph representation learning seeks to transform complex, high-dimensional graph structures into compact vector spaces that preserve both topology and semantics. Among the various strategies, subgraph-based methods provide an interpretable bridge between symbolic pattern discovery and continuous embedding learning. Yet, existing frequent or discriminative subgraph mining approaches often suffer from redundant multi-phase pipelines, high computational cost, and weak coupling between mined structures and their discriminative relevance. We propose **DS-Span**, a single-phase discriminative subgraph mining framework that unifies pattern growth, pruning, and supervision-driven scoring within one traversal of the search space. DS-Span introduces a *coverage-capped eligibility mechanism* that dynamically limits exploration once a graph is sufficiently represented, and an *information-gain-guided selection* that promotes subgraphs with strong class-separating ability while minimizing redundancy. The resulting subgraph set serves as an efficient, interpretable basis for downstream graph embedding and classification. Extensive experiments across benchmarks demonstrate that DS-Span generates more compact and discriminative subgraph features than prior multi-stage methods, achieving higher or comparable accuracy with significantly reduced runtime. These results highlight the potential of unified, single-phase discriminative mining as a foundation for scalable and interpretable graph representation learning.

Keywords: Graph Mining · Discriminative Subgraph · Graph Embedding · Frequent Pattern Mining · Representation Learning

1 Introduction

Graphs provide a natural and powerful formalism for representing relational data in diverse domains such as chemistry, bioinformatics, transportation, and social networks. In these settings, the ability to learn compact yet expressive representations of graphs is crucial for tasks such as classification, similarity search, and clustering. Graph representation learning aims to transform discrete graph structures into continuous vector spaces while preserving structural and semantic information.

Among the various approaches to graph representation, *subgraph-based methods* occupy a unique position by bridging symbolic and continuous perspectives. Instead of directly parameterizing the graph through neural architectures, these methods first discover *frequent* or *discriminative* substructures that recur across the dataset and then use their presence or embedding to form interpretable

graph-level features. This paradigm provides both transparency and efficiency, especially when domain interpretability and explainability are essential.

Classical algorithms such as **gSpan** [9] introduced depth-first search (DFS)-based canonical coding and pattern growth for efficient enumeration of frequent subgraphs. Later methods extended this idea toward discriminative mining, identifying subgraphs that maximize class-separating power rather than sheer frequency. Despite their success, most existing discriminative mining frameworks share three key limitations:

1. **Redundant multi-phase mining:** Frequent and discriminative phases are typically decoupled, leading to repeated subgraph enumeration and redundant computation.
2. **Weak coupling between supervision and mining:** Class-label information is often used only after enumeration, rather than guiding the mining process itself.
3. **Limited scalability:** Exhaustive enumeration on large or dense graphs produces substantial overlap among candidate subgraphs, causing combinatorial explosion and long runtimes.

To address these challenges, we propose **DS-Span**, a *single-phase discriminative subgraph mining* framework that integrates subgraph enumeration, pruning, and discriminative scoring into a unified search process. DS-Span maintains the canonical DFS-based traversal of gSpan but introduces several novel mechanisms to enhance both efficiency and discriminative relevance:

- **Coverage-capped eligibility:** graphs are dynamically excluded from further exploration once their feature coverage exceeds a user-defined threshold, preventing redundant pattern growth.
- **Information-gain-guided search:** candidate subgraphs are scored using information gain to prioritize those that best separate graph classes while minimizing overlap among features.
- **Single-phase pruning:** all constraints on support, coverage, and discriminative power are applied concurrently within one traversal, eliminating the need for iterative re-mining or post-hoc filtering.

By embedding these mechanisms directly into the mining process, DS-Span discovers a compact and highly discriminative subset of subgraphs that can serve as efficient inputs for downstream embedding and classification models. This unified framework drastically reduces redundancy and computational overhead, leading to substantial runtime savings without sacrificing predictive performance.

Our experimental study across diverse datasets demonstrates that DS-Span produces more informative and less redundant subgraph features than existing multi-phase methods. The resulting embeddings exhibit stronger class separation and lower variance across folds, underscoring the efficacy of single-phase, discriminatively guided subgraph mining as a foundation for scalable and interpretable graph representation learning.

2 Proposed Approach

We present **DS-Span**, a single-phase discriminative subgraph mining framework that produces a compact, high-utility feature set for downstream graph embedding and classification. DS-Span preserves the DFS-code canonicalization and rightmost-path extensions of **gSpan** [9], but differs in three design choices: (i) a *coverage-capped eligibility* mechanism that prunes graphs from further exploration once sufficiently represented, (ii) a *single-phase* mining schedule (no multi-phase restarts), and (iii) a *discriminative, coverage-constrained* feature selector. These choices eliminate redundant re-mining, guide the search toward class-separating patterns, and improve time/memory efficiency.

2.1 Preliminaries and Notation

Let $\mathcal{D} = \{G_i\}_{i=1}^{N}$ be a collection of labeled graphs with (training) class labels $y_i \in \mathcal{C}$. A graph $G = (V, E, \ell_V, \ell_E)$ has node labels $\ell_V : V \to \Sigma_V$ and edge labels $\ell_E : E \to \Sigma_E$. For two graphs S and G, we write $S \subseteq_{\text{iso}} G$ if S is subgraph-isomorphic to G. The *support* of a pattern S in $\mathcal{D}$ is

$$\text{supp}(S) \coloneqq \big|\{\, i \in [N] \,:\, S \subseteq_{\text{iso}} G_i \,\}\big|.$$

We use DFS-code tuples $(u, v, \ell_u, \ell_v, \ell_e)$ and rightmost-path extensions as in [9]. A DFS code is *canonical* if it is minimal in the gSpan lexicographic order; the canonical minimality test prevents duplicate patterns.

Coverage. For a set of mined subgraphs $\mathcal{F}$ and a graph G_i, define the coverage $\text{cov}(i; \mathcal{F}) \coloneqq \big|\{\, S \in \mathcal{F} : S \subseteq_{\text{iso}} G_i \,\}\big|$. Two thresholds govern coverage: a per-graph *minimum coverage* $min_cov \in \mathbb{N}$, and a per-graph *cap* $cap \coloneqq \gamma \cdot min_cov$ with $\gamma \geq 1$. Coverage influences *eligibility*: graphs that reach the cap stop contributing to further extensions.

Discriminative score. For a candidate S, let $\mathcal{I}(S) = \{i : S \subseteq_{\text{iso}} G_i\}$ and $\overline{\mathcal{I}}(S)$ its complement. Let $H(\cdot)$ denote Shannon entropy over class labels. We use information gain

$$IG(S) \coloneqq H(y) - \Big(\tfrac{|\mathcal{I}(S)|}{N}\, H(y \,|\, i \in \mathcal{I}(S)) \;+\; \tfrac{|\overline{\mathcal{I}}(S)|}{N}\, H(y \,|\, i \in \overline{\mathcal{I}}(S)) \Big),$$

and equivalently refer to *weighted entropy* $WE(S) \coloneqq H(y) - IG(S)$ when minimizing. A dataset-level *coverage constraint* requires the selected feature set $\mathcal{F}_\star$ to cover at least a fraction τ of graphs: $\big|\bigcup_{S \in \mathcal{F}_\star} \mathcal{I}(S)\big| \geq \tau N$.

2.2 Single-Phase Prospective Mining with Coverage-Capped Eligibility

Classical discriminative pipelines often alternate between (i) frequent enumeration under a support threshold and (ii) post-hoc discriminative filtering, sometimes across multiple mining *phases*. In contrast, DS-Span performs a *single*

Algorithm 1: Single-Phase Mining with Coverage-Capped Eligibility

```
Input  : 𝒟: graphs; δ: support; min_cov; γ; DFS-code canonical order
Output: 𝒞: candidate subgraph set; cov: per-graph coverage
Initialize 𝒞 ← ∅;
Initialize cov[i] ← 0 and eligibility set ℰ ← [N];
MINE(code, ℰ) X ← RIGHTMOSTEXTENSIONS(code, {G_i : i ∈ ℰ});
foreach x ∈ X do
    code′ ← code ∪ {x};
    if ISCANONICAL(code′) and supp(code′) ≥ ⌈δN⌉ then
        𝒞 ← 𝒞 ∪ {code′};
        foreach i ∈ ℐ(code′) do
            cov[i] ← cov[i] + 1;
            if cov[i] ≥ γ · min_cov then
                ℰ ← ℰ \ {i}
            end
        end
        MINE(code′, ℰ)
    end
end
MINE(∅, ℰ);
```

DFS traversal that (a) enforces a global support threshold and (b) dynamically shrinks the set of graphs that are eligible to produce further extensions as their coverage grows.

Let $\delta \in (0, 1]$ be the relative support threshold; we require $\mathrm{supp}(S) \geq \lceil \delta \cdot N \rceil$. We maintain a per-graph coverage counter and an *eligibility mask* $\mathcal{E} \subseteq [N]$ of graphs whose coverage is $<$ *cap*. Rightmost-path extensions are collected *only* from graphs in $\mathcal{E}$. Whenever a new frequent, canonical extension S' is accepted, we increment coverage for graphs in $\mathcal{I}(S')$ and remove any graph i with $\mathrm{cov}(i) \geq$ *cap* from $\mathcal{E}$, preventing redundant growth from already well-represented graphs.

Coverage completion (fairness top-up). A single traversal can still leave some graphs with $\mathrm{cov}(i) <$ *min_cov*. To avoid under-representing such graphs in downstream learning, we *complete* coverage by enumerating the smallest canonical codes *within each under-covered graph* and adding them to $\mathcal{C}$ until *min_cov* is met for that graph. These *coverage fillers* have low support by construction and are subsequently down-weighted or removed by the discriminative selector (2.3).

Unlike multi-phase methods that restart mining with relaxed thresholds, DS-Span grows patterns once and prunes *graphs* (eligibility) as soon as they are sufficiently represented. This reduces extension generation, avoids revisiting already saturated regions, and cuts redundant isomorphism checks, all while maintaining the anti-monotonicity guarantees of support and the duplicate-avoidance guarantees of DFS-code minimality.

Algorithm 2: Coverage-Constrained Discriminative Selection

Input : $\mathcal{C}$: candidates; $\{y_i\}$: labels; K; τ
Output: $\mathcal{F}_\star$: selected features

```
Compute IG(S) for all S ∈ C;
Sort C by IG(S) descending;
F* ← ∅, U ← ∅                                  // U: covered graph indices
foreach S ∈ C do
    if |F*| < K  and  |U ∪ I(S)| > |U| then
        F* ← F* ∪ {S};
        U ← U ∪ I(S);
        if |U| ≥ τN then
            break
        end
    end
end
```

2.3 Discriminative Selection under a Coverage Constraint

The mined candidate pool $\mathcal{C}$ is distilled into a compact feature set $\mathcal{F}_\star$ by optimizing for discriminative utility subject to dataset coverage. We formulate selection as a coverage-constrained set function optimization:

$$\max_{\mathcal{F}\subseteq\mathcal{C},\,|\mathcal{F}|\le K} \sum_{S\in\mathcal{F}} IG(S) \quad \text{s.t.} \quad \Big|\bigcup_{S\in\mathcal{F}} \mathcal{I}(S)\Big| \;\ge\; \tau N, \tag{1}$$

where K is an optional feature budget and $\tau \in (0, 1]$ controls dataset coverage (e.g., $\tau = 0.95$). We use a simple greedy strategy that is effective and easy to reproduce: sort candidates by $IG(S)$ descending, add a candidate if it *improves* the coverage of under-covered graphs and respects the budget, and stop when the coverage constraint is met.

Post-hoc filters that ignore coverage tend to pick many near-duplicates of the same motif, hurting generalization and wasting capacity. Our coverage constraint explicitly promotes *representative* features, reducing redundancy. Empirically (Sec. 3), this increases accuracy with *fewer* features and reduces variance across folds.

2.4 Feature Embedding and Classification

From the selected discriminative subgraphs $\mathcal{F}_\star = \{S_k\}_{k=1}^{K'}$, each graph G_i is represented by a normalized binary incidence vector

$$(x_i)_k = \begin{cases} \frac{1}{|\{\, j : S_j \subseteq_{\text{iso}} G_i \,\}|}, & \text{if } S_k \subseteq_{\text{iso}} G_i, \\ 0, & \text{otherwise.} \end{cases}$$

This normalization divides by the number of subgraphs present in G_i, ensuring that larger graphs do not produce disproportionately high feature magnitudes.

The resulting feature matrix $X = [x_1, x_2, \ldots, x_N]^\top$ is sparse and interpretable, directly encoding the presence of mined substructures.

Embedding model. Following the architecture of **DisFPGC** [1], we employ a shallow CBOW-style embedding network that learns two parameter matrices $W \in \mathbb{R}^{K' \times E}$ and $W' \in \mathbb{R}^{E \times N}$, where E denotes the embedding dimension and N the number of graphs. Each forward pass computes hidden activations

$$h_i = W^\top x_i, \qquad u_i = W'^\top h_i,$$

and outputs $\hat{y}_i = \mathrm{softmax}(u_i)$. The model is trained to minimize cross-entropy loss between $\hat{y}_i$ and the true one-hot label vector t_i:

$$\mathcal{L} = -\sum_{i=1}^{N} t_i^\top \log \hat{y}_i.$$

The same optimization procedure and criteria described in DisFPGC are used here. Consequently, any difference in classification performance arises solely from DS-Span's feature quality.

2.5 Design Contrasts with Prior Literature

Versus gSpan [9]. Both use DFS-code canonicalization and rightmost expansion. DS-Span adds *coverage-capped eligibility* and a *discriminative, coverage-constrained* selection after a single traversal, prioritizing class-separating, non-redundant patterns.

Versus multi-phase discriminative miners (DisFPGC [1]). Multi-phase schedules repeatedly enumerate with varying thresholds and only later integrate supervision, which causes redundant work and weak coupling. DS-Span mines once at a fixed δ, prunes exploration early via eligibility caps, and selects features with an explicit coverage constraint, producing leaner and more predictive feature sets with lower runtime.

Versus embedding-first methods. Our pipeline is *feature-first*: it yields interpretable substructures with quantified discriminative value. Embeddings are a thin, optional layer on top of these features, making improvements *attributable* to the mined patterns rather than model capacity.

3 Implementation & Experimental Results

This section distils the evaluation of DS-Span into four viewpoints: (i) accuracy versus existing graph-classification methods, (ii) how many subgraphs are mined (iii) how long mining takes and (iv) how alternative embedding choices affect class separability.

3.1 Datasets, Protocol, and Hardware

We evaluate on the TU Dataset benchmarks [6]: **D&D** [5], **ENZYMES** [2], **Proteins** [3], **MUTAG** [4], **PTC** [7], **NCI1** and **NCI109** [8], and **Reddit-M-5k** [10]. For each dataset we regenerate depth-first search frequency caches, run ten repetitions of stratified 10-fold cross-validation with the supervised coverage cap in Section 2.3, and log the number of retained subgraphs per fold. The CBOW-style embedding model uses five epochs, learning rate $\alpha = 1.0$, and 64-dimensional representations. All experiments, including the reproduction of **DisFPGC** [1], run on an AMD Ryzen 5 5600G CPU with 32 GB RAM and an RTX 1060 GPU (Python 3.12.3).

Table 3 already consolidates the requested accuracy comparison: DS-Span leads on D&D, ENZYMES, MUTAG, NCI109, Proteins, and PTC, ties Reddit-M-5k, and yields only NCI1 to DisFPGC. To complement those numbers we compare feature budgets and mining costs, then describe how different embedding choices behave.

3.2 Feature Budget and Mining Cost

Table 1: Average feature counts for DS-Span and DisFPGC.

Dataset	Avg. DS-Span features	Avg. DisFPGC features
D&D	9.14	231.00
Enzymes	3.57	179.50
Mutag	20.02	220.60
NCI1	12.92	283.60
NCI109	11.55	262.60
Proteins	18.99	607.60
PTC	15.07	180.10

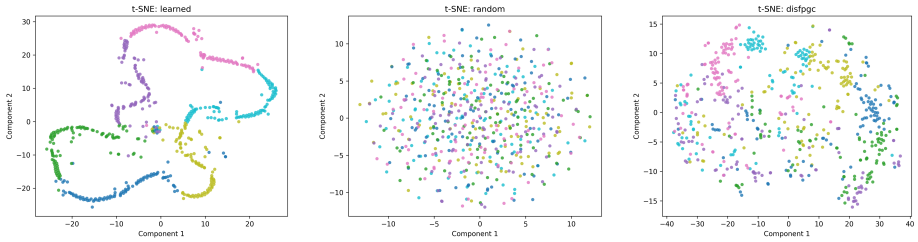
Table 2: Average mining times (seconds) for DS-Span and DisFPGC.

Dataset	DS-Span mining	DisFPGC mining
D&D	54.61	709.80
Enzymes	807.23	1144.80
Mutag	201.24	4178.00
NCI1	110.01	422.30
NCI109	96.37	362.10
Proteins	1455.33	7448.20
PTC	11.20	148.40

Tables 1 and 2 show that DS-Span typically retains fewer than twenty subgraphs and completes mining in seconds, whereas DisFPGC produces hundreds of features and still spends 7–265× longer because it repeatedly re-enumerates candidates before applying supervision. Filtering and embedding remain sub-second on all datasets and are omitted for clarity.

3.3 Embedding Quality and Visual Evidence

To compare embedding choices we visualise DS-Span, a Gaussian random baseline, random-feature embeddings (binary indicators for a random subset of mined subgraphs), their trained counterpart, and the reproduced DisFPGC embeddings. Figure 1 illustrates the D&D case: DS-Span produces clearly separated clusters despite using only nine patterns, DisFPGC forms worse clusters and requires roughly twenty-five times as many subgraphs, and the random variants collapse into an indistinguishable cloud. Other datasets exhibit the same qualitative behaviour, underscoring that DS-Span's discriminative mining drives its accuracy advantage.



(a) DS-Span (b) Random baseline (c) DisFPGC reproduction

Fig. 1: t-SNE visualisations for the D&D dataset comparing DS-Span embeddings, Gaussian noise, and the reproduced DisFPGC embeddings.

Table 3: Comparison of Methods on Datasets

Method	**D&D**	**Enzymes**	**Reddit-M-5k**	**Mutag**	**NCI1**	**NCI109**	**Proteins**	**PTC**
WL-1	78.50 (0.44)	50.03 (2.04)	– –	75.35 (2.44)	84.19 (0.41)	84.33 (0.26)	73.03 (0.33)	60.72 (1.80)
WL-OA	78.64 (0.48)	57.02 (1.24)	– –	82.71 (2.34)	85.06 (0.31)	85.06 (0.42)	73.10 (0.81)	60.74 (1.72)
SP	78.47 (0.61)	39.87 (2.01)	– –	82.24 (1.80)	73.97 (0.41)	73.00 (0.28)	75.53 (0.59)	59.36 (1.79)
$rLap_{BGRL}$	74.80 (10.17)	81.50 (5.39)	– –	–	84.34 (4.03)	–	–	–
$rLap_{GCL}$	70.90 (5.01)	87.50 (9.86)	– –	–	75.27 (3.34)	–	–	–
node2vec	– –	72.63 (10.20)	– –	52.68 (1.56)	57.49 (3.57)	58.85 (8.00)	– –	– –
sub2vec	– –	61.05 (15.79)	– –	50.67 (1.50)	53.03 (5.55)	59.99 (6.38)	– –	– –
graph2vec	58.64 (0.01)	44.33 (0.09)	– –	83.15 (9.25)	73.22 (1.81)	74.26 (1.47)	73.30 (2.05)	60.17 (6.86)
HGP-SL	80.96 (1.26)	68.79 (2.11)	– –	– –	78.45 (0.77)	80.67 (1.16)	84.91 (1.62)	– –
GraphSage	72.90 (2.00)	58.20 (6.00)	50.0 (1.3)	79.80 (13.90)	76.00 (1.80)	– –	73.00 (4.50)	– –
$SAGPool_g$	76.19 (0.94)	– –	– –	90.42 (7.78)	74.18 (1.20)	74.06 (0.78)	70.04 (1.47)	– –
DiffPool	75.00 (3.50)	59.50 (5.60)	53.8 (1.4)	77.60 2.70	76.90 (1.90)	– –	72.70 3.80	73.70 (3.50)
$GIN_{\epsilon-JK}$	– –	39.30 (1.60)	57.0 (1.7)	– –	78.30 (0.30)	– –	72.20 (0.70)	– –
GCN	– –	– –	– –	85.60 (5.80)	80.20 (2.00)	– –	76.00 (3.20)	64.20 (4.30)
GCKN	– –	– –	– –	97.20 (2.80)	83.90 (1.20)	– –	75.90 (3.20)	69.40 (3.50)
U2GNN	95.67 (1.89)	– –	– –	88.47 (7.13)	– –	– –	80.01 (3.21)	91.81 (6.61)
GIU-Net	– –	70.00 –	– –	95.70 –	80.20 –	77.00 –	77.60 –	85.70 –
GE-FSG	91.69 (0.02)	49.33 (0.07)	– –	84.74 (0.07)	84.36 (0.02)	85.59 (0.01)	81.79 (0.04)	62.57 (0.09)
DisFPGC	93.46 (0.54)	59.82 (0.16)	93.10 (1.52)	97.00 (0.53)	**94.92** (0.21)	97.40 (0.12)	83.44 (0.67)	84.94 (0.50)
GE-ODS	**96.12** (0.29)	**93.80** (0.59)	**93.19** (0.25)	**99.67** (0.27)	89.44 (0.31)	**99.43** (0.08)	**93.89** (0.59)	**95.03** (0.83)

4 Conclusion

In this work, we proposed DS-Span, a computationally efficient approach for generating whole-graph embeddings by leveraging a single-phase discriminative

subgraph mining technique. Our method addresses the challenges posed by traditional multi-phase approaches, such as high computational cost and redundancy, by dynamically shrinking the search space and employing a robust filtering mechanism to identify highly discriminative subgraphs. These subgraphs form the basis for embeddings that preserve critical structural and class-separating features. The contributions of our work are multifaceted. First, we developed a single-phase subgraph mining algorithm that eliminates iterative overhead and significantly enhances scalability. Second, we introduced a supervised graph embedding methodology that outperforms existing state-of-the-art methods in classification tasks, as demonstrated across diverse benchmark datasets. Lastly, we showed the practical applicability of our approach in critical domains, such as drug discovery, by enabling precise and efficient classification of molecular graphs, thereby reducing resource-intensive experimental validation. The results indicate that our method attains greater accuracy than baseline methods and state-of-the-art techniques in most datasets, with a very low standard deviation. This reflects the robustness and stability of our work. Specifically, it indicates our approach effectively discriminates between classes and generates high-quality whole-graph embeddings useful for downstream graph analytics. As our approach is computationally efficient and highly optimized, it requires fewer resources and processing time than its predecessors with similar techniques. These findings underline the effectiveness of our method in graph classification and suggest its potential for practical applications. In future work, we aim to explore unsupervised filtering mechanisms to enhance the generalizability of our method and investigate dataset-specific hyperparameter tuning to optimize performance across diverse graph datasets even further.

References

1. Alam, M.T., Ahmed, C.F., Samiullah, M., Leung, C.K.: Discriminating frequent pattern based supervised graph embedding for classification. In: PAKDD (2021)
2. Borgwardt, K.M., Ong, C.S., Schonauer, S., Vishwanathan, S.V.N., Smola, A.J., Kriegel, H.p.: Protein function prediction via graph kernels. Bioinformatics **21**(Suppl 1), i47–i56 (6 2005). https://doi.org/10.1093/bioinformatics/bti1007, https://doi.org/10.1093/bioinformatics/bti1007
3. Borgwardt, K.M., O.C.S.S.V.S.S.A.K.H.: Protein function prediction via graph kernels **3** (10 2005)
4. Debnath, A.K., Lopez de Compadre, R.L., Debnath, G., Shusterman, A.J., Hansch, C.: Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. Journal of Medicinal Chemistry **34**(2), 786–797 (1991). https://doi.org/10.1021/jm00106a046, https://doi.org/10.1021/jm00106a046
5. Dobson, P.D., Doig, A.J.: Distinguishing Enzyme Structures from Non-enzymes Without Alignments. Journal of Molecular Biology **330**(4), 771–783 (7 2003). https://doi.org/10.1016/s0022-2836(03)00628-4, https://doi.org/10.1016/s0022-2836(03)00628-4
6. Morris, C., Kriege, N.M., Bause, F., Kersting, K., Mutzel, P., Neumann, M.: Tu-dataset: A collection of benchmark datasets for learning with graphs. In: ICML 2020 Workshop on Graph Representation Learning and Beyond (2020)

7. Toivonen, H., Srinivasan, A., King, R.D., Kramer, S., Helma, C.: Statistical evaluation of the Predictive Toxicology Challenge 2000–2001. Bioinformatics **19**(10), 1183–1193 (07 2003). https://doi.org/10.1093/bioinformatics/btg130, https://doi.org/10.1093/bioinformatics/btg130
8. Wale, N., Watson, I., Karypis, G.: Comparison of descriptor spaces for chemical compound retrieval and classification. Knowledge and Information Systems **14**(3), 347–375 (Mar 2008). https://doi.org/10.1007/s10115-007-0103-5
9. Yan, X., Han, J.: gspan: graph-based substructure pattern mining. In: 2002 IEEE International Conference on Data Mining, 2002. Proceedings. pp. 721–724 (2002). https://doi.org/10.1109/ICDM.2002.1184038
10. Yanardag, P., Vishwanathan, S.: Deep Graph Kernels. KDD '15: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining pp. 1365–1374 (8 2015). https://doi.org/10.1145/2783258.2783417, https://doi.org/10.1145/2783258.2783417