ISSN 2007-9737

Closed and Maximal Patterns on Dissimilarity-Based Graph Embedding for Supervised Classification

Daybelis Jaramillo-Olivares*, J. Ariel Carrasco-Ochoa,
José Fco. Martínez-Trinidad

Instituto Nacional de Astrofísica Óptica y Electrónica,
Mexico

{daybelis, ariel, fmartine}@inaoep.mx

Abstract. This paper presents a comparative analysis of the performance of Dissimilarity-based Graph Embedding, built from all frequent, closed, and maximal graph patterns for supervised classification. Dissimilarity-based Graph Embedding employs a dissimilarity metric to transform a graph into a vector representation by computing each entry of the vector as the dissimilarity between a graph and each graph pattern mined from a graph collection. Given that the number of graph patterns mined can be too large, subsets such as closed and maximal patterns are used to reduce the number of graph patterns, leading to more compact and non-redundant representations. Our study employed four benchmark graph collections and six different supervised classifiers. The experimental results enabled us to identify which type of graph pattern (all frequent, closed, or maximal) allows a Dissimilarity-based Graph Embedding that yields the highest performance in supervised classification.

Keywords. Dissimilarity-based graph embedding, supervised classification, graph patterns, closed and maximal graph patterns.

1 Introduction

Graph data structures are widely used because they represent relationships between objects [3] in a flexible and versatile way. Graphs are employed to represent a variety of data across many domains, like chemistry [6], computer vision [33], biology networking [18], and social networking [17], among others.

Graph pattern mining identifies frequent graph patterns (frequent subgraphs) in a single graph or across a graph collection, based on a user-specified frequency threshold [14, 28, 22, 13]. There are two types of graph patterns: exact and approximate. Exact graph patterns match subgraphs in an exact form, while approximate patterns allow some changes.

Graph embeddings have become essential in machine learning applications involving graph-structured data, as they provide an easy way to represent complex data. This paper focuses on these embeddings, which are vector representations of graphs. In contrast to other embeddings, graph embeddings are designed to capture the structural and relational information of the graph-based data. To capture this information, we followed the Dissimilarity-based Graph Embedding approach used in [12], where a set of mined patterns serve to build features by identifying the dissimilarity between them and the graphs within a graph collection.

To reduce the number of patterns mined and eliminate redundancy, pattern subsets, such as closed and maximal graph patterns, have been employed. Closed patterns retain the complete support information while discarding subpatterns that share the same support [24]. On the other hand, maximal patterns offer a more reduced number of patterns, as they only consider the largest patterns [23].

The objective of this paper is to conduct a comparative analysis of the performance of Dissimilarity-based Graph Embedding built from all frequent, closed, and maximal graph patterns for supervised classification. Specifically, we

aim to evaluate how closed and maximal pattern graph embeddings compare to using the entire (non-pruned) set of frequent graph patterns in terms of classification performance and how much these subsets reduce the size of the embedding space.

To achieve this, we employ an experimental methodology that includes mining all the graph patterns from a graph collection, followed by a post-processing stage to obtain the closed and maximal pattern subsets from the whole set of frequent patterns. The graph patterns are then used to generate dissimilarity graph-based embeddings. Subsequently, we evaluate these embeddings using supervised classifiers.

The remainder of this paper is organized as follows: Section 2 provides some basic concepts, and Section 3 provides some background. Section 4 describes the methodology employed to evaluate the use of Dissimilarity-based Graph Embedding for supervised classification using all frequent, closed, and maximal graph patterns. Section 5 presents the experiments conducted on benchmark graph collections and analyzes the results. Finally, Section 6 summarizes the conclusions and outlines directions for future work.

2 Basic Concepts

Since this work focuses on graph-structured data, we begin by defining the concept of a graph and some terminology related to frequent graph pattern mining.

A graph G is a structure that represents relationships among objects, consisting of a vertex set V and an edge set E that connect pairs of vertices. In this paper, we are focus on undirected labeled graphs, where labels are assigned to both vertices and edges.

An undirected labeled graph is defined as $G = (V, E, L_V, L_E, \ell_V, \ell_E)$. In this type of graph, labels are assigned to both vertices and edges, and the edges have bidirectional relationships. The sets L_V and L_E are the possible vertex and edge labels, respectively. The functions $\ell_V : V \to L_V$ and $\ell_E : E \to L_E$ assign the labels to the vertices and edges.

A graph collection consists of a set of graphs denoted by $D = \{G_1, G_2, \cdots, G_n\}$.

A subgraph g is a graph contained in another graph G, denoted by $g \subseteq G$, meaning that all vertices and edges in g are also in G. Moreover, a supergraph G' is a graph that contains another graph G as a subgraph; that is, G is a subgraph of G'.

Frequent graph patterns can be mined using exact or approximate approaches. A frequent exact graph pattern follows a strict approach, while frequent approximate graph mining follows a flexible approach where a pattern can be dissimilar to another.

A frequent exact graph pattern is a subgraph p that appears as an exact or isomorphic match in at least σ graphs within a graph collection D, where σ denotes a predefined minimum support threshold.

An exact or isomorphic match exists if a subgraph $g_i \subseteq G_i$ has an identical match to p.

A frequent approximate graph pattern is a subgraph p that appears as a similar subgraph in at least σ graphs within a graph collection D, where σ is a predefined minimum support threshold. Subgraphs p and $g_i \subseteq G_i \in D$ are considered similar if $d(p, gi) \leq \delta$, where δ is a predefined dissimilarity threshold, and d is a graph dissimilarity function (often based on the graph edit distance (GED)). p is a frequent exact pattern if $\delta = 0$.

Closed and maximal patterns are subsets of the frequent patterns that address the redundancy and obtain a more compact subset of patterns. Their definitions apply to both exact and approximate patterns.

A closed graph pattern p is a frequent pattern for which there is no supergraph P' of p that has the same support. This type of pattern preserves the support information.

A maximal graph pattern p is a frequent pattern that has no frequent supergraph P'. This type of pattern retains only the largest subgraphs that satisfy the support threshold.

Because this work will explore the use of closed and maximal patterns in dissimilarity-based graph embedding for supervised classification, it is necessary to evaluate the dissimilarity of these patterns. In this paper, we will use the dissimilarity function presented in [12], which is based on the Graph Edit Distance (GED).

This function is computed following a strategy that avoids redundant computations and provides an efficient approximation of the overall transformation costs previously obtained and adding those of the newly introduced nodes and edges.

Given two graphs G_1 and G_2, the dissimilarity function is defined as:

$$d(G_1, G_2) = \sum_{i=1}^{k} c(V_i) + \sum_{j=1}^{m} c(E_j). \quad (1)$$

where $c(V_i)$ and $c(E_j)$ represent the costs associated with the modifications, including label substitution, insertion, and deletion; k and m indicate the modifications made to the vertices and edges needed to transform G_1 into G_2.

The defined dissimilarity function forms the basis for representing graph patterns as vectors, in which each graph pattern is transformed while preserving its dissimilarity transformation cost.

Dissimilarity-based Graph Embedding transforms graphs into a numerical representation as an entry in a vector space [5]. This vector is built following the approach used in [12]; specifically, the vector is built by calculating the dissimilarities between a graph in a graph collection and each graph pattern mined.

Given a graph G and the set of mined patterns P, the vector $z = [d_{min}(G, p_1), ..., d_{min}(G, p_{|P|})]$ can be built. For which, we calculate the dissimilarity $d(g_i, p_j)$ for all occurrences g_i of p_j within G and identify the minimum dissimilarity value $d_{min}(g_i, p_j)$ for each pattern $p_j \in P$. Consequently, given a graph collection D, each graph G in the graph collection D is embedded using frequent graph patterns. This process transforms the graph collection $D = \{G_1, G_2, \cdots, G_n\}$, into a vector set $Z = \{z_1, z_2, \cdots, z_n\}$.

3 Background and Related Work

Pattern mining has long been used in classification tasks, both as a way to extract informative features and as a basis for interpretable models.

Approaches such as subgroup discovery [26] and pattern-based classification [19, 12] involve the identification of patterns.

In the literature, there are some algorithms designed to mine exact graph patterns (e.g., FSG [16], DewgSpan [11]), whereas others focus on mining approximate graph patterns (e.g., SUBDUE [10], VEAM [2], AGrAP [9], MaNIACS [25]). However, there are few algorithms in the literature that consider patterns allowing structural and label variations in both vertices and edges within a graph collection. (e.g., REAFUM [20]).

Graph embeddings have become a powerful tool in machine learning because they can effectively preserve relevant information while providing a way to represent graph-structured data. This is achieved by transforming graphs into vectors [34]. In the context of graph mining, graph embeddings have been applied in machine learning tasks [21, 15] and by embedding patterns into vector representations [12, 4], particularly for use with standard classifiers such as support vector machines or neural networks.

The use of graph embeddings allowed patterns to be analyzed and compared using standard machine learning techniques. The use of exact versus approximate graph patterns in embeddings and their impact on classification performance has been studied [12]. Exact patterns capture precise structures but are too strict. In contrast, approximate patterns allow for controlled variations, which enhances robustness in noisy data.

Redundancy is a major challenge in pattern mining; to address this, several procedures based on structural properties have been proposed to choose a subset of patterns. Closed and maximal patterns are one type of these subsets of patterns.

They aim to reduce the number of patterns, leading to more compact and not redundant patterns [27]. Closed patterns retain the frequency information while eliminating redundant patterns that have different support [24, 1, 7, 32]. Maximal patterns, on the other hand, only keep the longest ones. This approach reduces the number of patterns, but it may also discard patterns that are useful [23, 8, 30, 31].

Therefore, it is important to evaluate these subsets of patterns in terms of the number of patterns and their impact on the quality of classification. Considering closed and maximal patterns reduce the size of the embedding space.

ISSN 2007-9737

This has implications not only for classification performance but also for training efficiency and memory usage.

4 Experimental Methodology

The proposed methodology consists of several stages for assessing Dissimilarity-Based Graph Embedding for Supervised Classification employing all frequent, closed, and maximal graph patterns.

First, the graph collections are divided into training and testing sets, with 80% of the data used for training and the remainder 20% for testing. To ensure robustness and reduce overfitting, a 5-fold cross-validation is applied. Each graph collection was randomly divided into five folds, with one fold used for testing and the remaining four for training in each iteration. Since the dataset includes labeled classes, the training set is divided into separate sets according to the different class labels. For each class, the frequent graph patterns are mined with a graph miner.

Then, a post-processing step is performed to extract closed and maximal graph patterns from the set of initially mined graph patterns. Each graph pattern is examined in relation to others that contain it as a subgraph. The process of extracting closed and maximal graph patterns focuses on identifying whether a graph pattern has supergraphs. If the graph pattern has a supergraph with the same support, the current pattern is not closed. On the other hand, only those graph patterns that don't have a supergraph (regardless of support) are classified as maximal graph patterns.

Once the frequent graph patterns for every class have been identified in the training set, the entire dataset (both training and testing sets) is embedded.

The graphs are embedded using the procedure presented in [12] and described in Section 2, and the dissimilarity function described in equation 1, resulting in an n-length vector representation for each graph in the collection.

On the vector representation previously constructed, conventional supervised classifiers designed for vectors can now be applied, as we have a Dissimilarity-based Graph Embedding.

Table 1. Dataset characteristics of the PTC graphs collections

PTC collection	$\|D\|$	$\|L_V\|$	$\|L_E\|$	$\|C_1\|$	$\|C_2\|$
MM	336	18	4	207	129
MR	344	20	4	192	152
FM	349	18	4	206	143
FR	351	19	4	230	121

The proposed experimental methodology involves the use of various classifiers to represent the different strategies of supervised classification. Each classifier is trained using the Dissimilarity-based Graph Embedding generated from the training data. The performance of the trained models is then evaluated on the 20% test set. The classification performance is measured using accuracy and F1-score. The quality of the classifiers' performance is computed as the average over the five folds of the cross-validation.

5 Experiments and Results

This section presents the results of experiments conducted to assess the performance of supervised classification using dissimilarity-based graph embedding derived from all frequent, closed, and maximal (exact and approximate) graph patterns.

The experimental setup consisted of a Linux Ubuntu 22 system with dual Intel Xeon E5-2620 processors (2.40 GHz) and 256 GB RAM.

We selected the PTC graph collections for our experiments because they are datasets with labeled classes. These datasets were retrieved from the Network Repository [29]. They include compounds categorized into two classes (positive or negative) based on carcinogenicity in male mice (MM), male rats (MR), female mice (FM), and female rats (FR).

The properties of the PTC graph collections are shown in Table 1, which reports the size of the graph collection $|D|$, the number of vertex labels $|L_V|$ and edge labels $|L_E|$, as well as the number of graphs in each class $|C_1|$ and $|C_2|$ for each PTC collection.

As a graph miner, we use REAFUM [20] because this algorithm allows us to mine approximate graph patterns, and also it can mine exact graph patterns when the dissimilarity threshold $\delta = 0$.

Table 2. Mean number of graph patterns mined obtained from the five-fold cross-validation for the datasets of Table 1

Pattern	Dissimilarity δ | Support σ					
	$\delta = 0$		$\delta = 1$		$\delta = 2$	
	$\sigma =$ 40%	$\sigma =$ 50%	$\sigma =$ 80%	$\sigma =$ 90%	$\sigma =$ 80%	$\sigma =$ 90%
PTC-MM						
All	13.8	8.8	15	10	56.6	32.6
Closed	12.6	8.8	14	9	55.4	31.6
Maximal	7.4	4.8	11	8.6	41.6	22.6
PTC-MR						
All	16.6	9.8	16.2	10.6	61	37.2
Closed	16	9.8	15.2	9.6	59.4	36
Maximal	8.2	6	13	9.6	44.6	26.8
PTC-FM						
All	12.4	9.8	18	10	66.2	41.8
Closed	12	9.8	17	9	64.6	40.4
Maximal	6. 2	6	14.6	9	48.2	30.8
PTC-FR						
All	13.4	9.6	14.4	10.6	55.2	35.6
Closed	13.4	9.6	13.4	9.6	53.6	34.2
Maximal	5. 6	5.6	9.8	9.6	39.2	23.6

Table 3. Default scikit-learn parameters settings used for the classifiers

Classifier	**Default parameters**
KNN	`n_neighbors=5`
Linear SVM	`C=1.0`, `penalty=l2`, `max_iter = 1000`
Decision Tree	`criterion='gini'`, `splitter='best'`
Random Forest	`n_estimators=100`, `criterion='gini'`
Neural Networks	`activation='relu'`, `solver='adam'`,
	`hidden_layer_sizes=(100,)`
GBM	`loss='log_loss'`, `learning_rate=0.1`,
	`n_estimators=100`

REAFUM was executed, varying the dissimilarity and support thresholds. The dissimilarity threshold (δ) specifies how much pattern occurrences can differ. It was varied from zero, which allows for the mining of exact graph patterns, to two, in increments of one. On the other hand, the support threshold (σ) specifies how often the graph pattern must appear in the graphs of the collection. For exact graph patterns, the support thresholds were set to 40% and 50%, whereas for approximate patterns, they were set to 80% and 90%. These support thresholds allowed the mining of exact graph patterns since high support values often result in the mining of very few or even no patterns, preventing the building of the dissimilarity-based graph embeddings. The flexibility of the approximate patterns allowed us to use high support values.

For obtaining closed and maximal graph patterns, we perform a post-processing step to extract closed and maximal graph patterns to ensure that the resulting set of patterns preserves the same properties and to obtain a fair performance analysis.

Table 2 presents the mean number of graph patterns mined obtained for each dataset from the five-fold cross-validation. It includes the whole set of all frequent graph patterns, as well as the closed and maximal subsets.

The number of patterns mined directly affects the dimensionality of the Dissimilarity-based Graph Embeddings. Table 2 illustrates how varying the dissimilarity threshold δ impacts the number of patterns mined. Increasing the dissimilarity threshold allows for greater tolerance in the similarity of the graphs, which generally leads to the discovery of more patterns. Additionally, the number of patterns decreases when closed or maximal patterns are considered, especially in the case of maximal patterns. In some cases, frequent and closed patterns may yield the same results. Mining a higher number of frequent patterns yields a more substantial reduction of closed and maximal patterns.

The box plots in Fig. 1 summarize the distribution of the metric values (accuracy and F1-score, respectively) obtained from the six different classifiers ((k-Nearest Neighbors (KNN), Linear Support Vector Machines (Linear SVM), Decision Tree, Random Forest, Neural Networks, and Gradient Boosting Classifier (GBM)) evaluated across the four PTC graph collections. All classifiers were implemented using the scikit-learn library in Python with their default parameter settings (see Table 3) to ensure a fair and unbiased comparison and were chosen to represent diverse strategies for supervised classification. In these plots, for each classifier, three grouped bars indicate the performance under the different pattern types. Each box corresponds to a different kind of pattern (frequent graph patterns represented by a blue box, closed graph patterns represented by an orange box, or maximal graph patterns represented by a green box).

The box plot allows for a direct comparison of how each classifier performs depending on the type of pattern considered. They highlight differences in the prediction among the classifiers and show whether certain classifiers consistently achieve higher or lower accuracy across all pattern types or whether performance varies with the pattern type used.

Fig. 1a presents the mean accuracy scores of the different classifiers across all combinations of dissimilarity and support thresholds. The classifiers, such as KNN and Linear SVM, tend to show lower accuracy across all three pattern subsets (all, closed, and maximal) compared to Decision Tree, Random Forest, GBM, and Neural Networks, which achieve much higher accuracy scores across all three pattern types.

When comparing across pattern types, the differences are barely noticeable. For most classifiers, accuracy scores are slightly higher with closed or maximal patterns compared to all frequent patterns, suggesting that reducing redundancy in the pattern sets (closed and maximal) leads to more effective classification. Overall, the results highlight that the classifier plays an important role in the accuracy performance; however, the type of pattern also affects the accuracy performance. The closed and maximal pattern subsets obtained a barely noticeable performance improvement.

Fig. 1b presents the F1-scores of the different classifiers among all the combinations of dissimilarity and support thresholds. Linear SVM yields relatively lower F1 scores (it reflects a weaker overall balance between precision and recall) across all pattern types compared to Decision Tree and Random Forest, which achieve the highest F1 scores.

Across pattern types, closed and maximal patterns again provide a slight improvement over all frequent patterns, suggesting that removing redundancy in the pattern space benefits classifier performance.

The results from Fig. 1a and Fig. 1b show how the type of pattern representation impacts the performance, with the closed and maximal pattern subsets generally leading to better results.

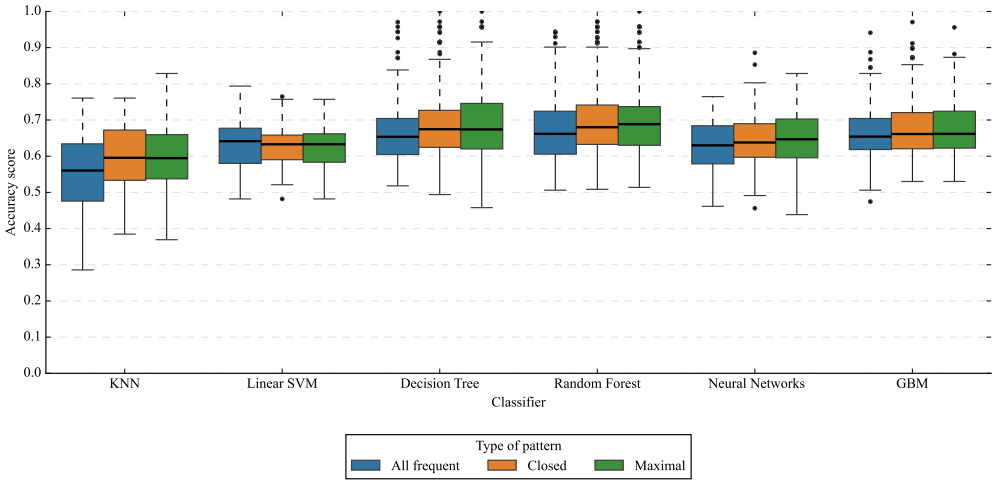
The box plots in Fig. 2 summarize the distribution of the metric values (accuracy and F1-score, respectively) obtained from all the classifiers. In these plots, each box represents a different type of pattern: all frequent graph patterns (represented by a blue box), closed graph patterns (represented by an orange box), or maximal graph patterns (represented by a green box), along with a corresponding combination of dissimilarity and support thresholds. For each parameter setting, three grouped boxes illustrate the performance of all frequent, closed, and maximal graph patterns. This plot allows us to compare how the type of pattern mined and the variations in the thresholds influence the classifiers' performance.

Fig. 2a shows that the Dissimilarity-based Graph Embeddings, built with approximate graph patterns, consistently achieved higher accuracy values, with the combination of a dissimilarity threshold of $\delta = 2$ and a support threshold of $\sigma = 80\%$ yielding the highest accuracy among all combinations, particularly with maximal graph patterns, represented by the green box, which achieve the best results. The results also indicate that the lowest performance are those obtained with a dissimilarity threshold $\delta = 0$, which corresponds to the exact graph patterns. In this case, in most combinations, the maximal graph patterns yielded the best results in comparison with the other type of patterns (closed and all frequent), with all frequent patterns (non-pruned) obtaining the worst results.

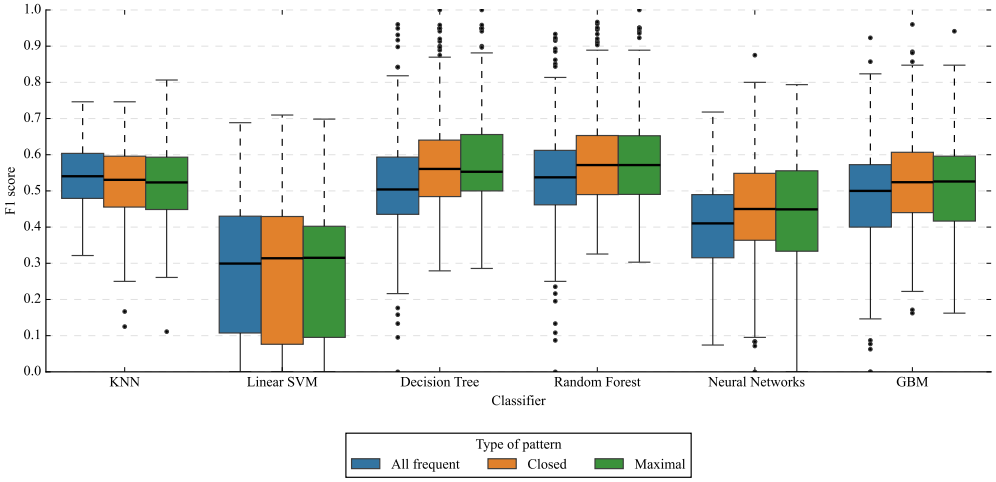
Fig. 2b shows that the Dissimilarity-based Graph Embeddings built with approximate patterns consistently achieved higher F1-score values. The combination of dissimilarity threshold $\delta = 2$ and support threshold $\sigma = 80\%$ yielded the highest F1-score among all combinations. In contrast, the lowest performance are those obtained with a dissimilarity threshold $\delta = 0$, corresponding to the exact graph patterns.

The results generally align with the trends observed in the accuracy plot. Closed and maximal patterns consistently yield higher F1 scores than all frequent patterns, suggesting that pattern subsets improve the balance between precision and recall when used to build Dissimilarity-Based Graph Embeddings for Supervised Classification.

Another aspect to consider is the computation time of the steps required to obtain the Dissimilarity-based Graph Embeddings. These steps include mining the entire set of graph patterns,



(a) Box plot showing the accuracy distribution across various classifiers.

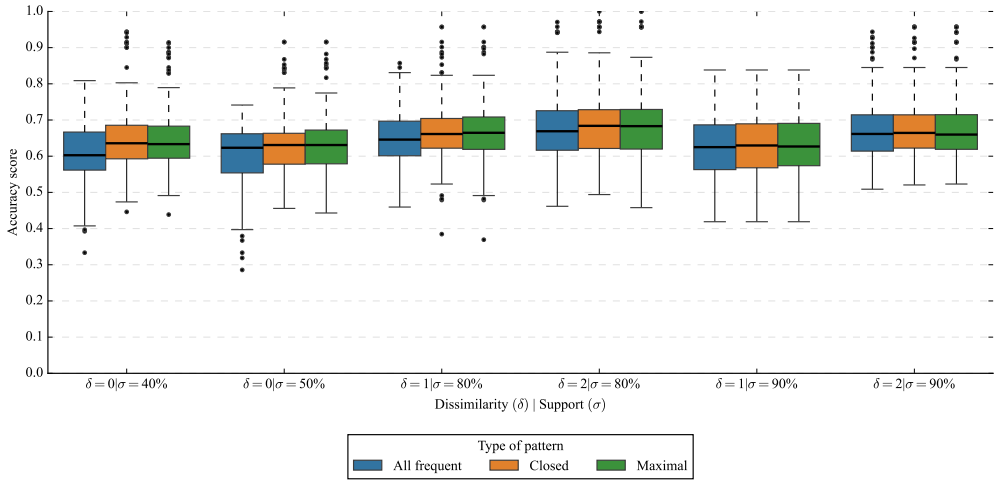


(b) Box plot showing the F1-score distribution across various classifiers.

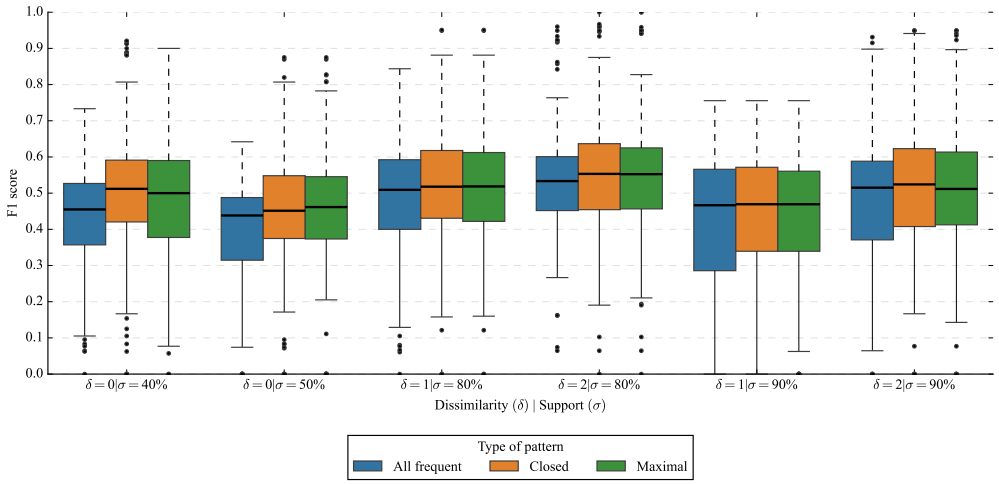
Fig. 1. Performance comparison across various classifiers (k-Nearest Neighbors (KNN), Linear Support Vector Machines (Linear SVM), Decision Tree, Random Forest, Neural Networks, and Gradient Boosting Classifier (GBM))

extracting closed and maximal graph patterns, and the embedding process for all types of graph patterns (all, closed, and maximal). Table 4 reports the mean execution times across the five folds and four graph collections.

The results indicate that the mining graph pattern stage was consistently the most time-consuming, dominating the total computational cost. The step for extracting closed and maximal graph patterns was comparatively brief. Once the graph patterns were obtained, the embedding processes for the three types of graph patterns (all, closed, and maximal) were not particularly time-consuming. Both all and closed graph patterns obtained similar performance, while the embedding process for maximal graph patterns was faster. Overall, the total execution time reflects a strong influence of the mining graph patterns stage. These results underscore the inherently time-consuming nature of mining patterns from dense and large graphs, where



(a) Box plot showing the accuracy distribution across various classifiers varying dissimilarity δ and support σ thresholds.



(b) Box plot showing the F1-score distribution across various classifiers varying dissimilarity δ and support σ thresholds.

Fig. 2. Performance comparison across various classifiers (k-Nearest Neighbors (KNN), Linear Support Vector Machines (Linear SVM), Decision Tree, Random Forest, Neural Networks, and Gradient Boosting Classifier (GBM)) varying dissimilarity δ and support σ thresholds

the combinatorial explosion of candidate structures significantly increases computational time.

6 Conclusion and Future Work

This paper compares the performance of Dissimilarity-based Graph Embedding built from all frequent, closed, and maximal graph patterns for supervised classification. The objective of Dissimilarity-based Graph Embedding using graph patterns is to represent a graph as a vector, where

Table 4. Mean execution times (in seconds) for each stage, computed across the five folds and four graph collections. The final row reports the total time

Stage	Dissimilarity δ | Support σ					
	$\delta = 0$		$\delta = 1$		$\delta = 2$	
	$\sigma = 40\%$	$\sigma = 50\%$	$\sigma = 80\%$	$\sigma = 90\%$	$\sigma = 80\%$	$\sigma = 90\%$
Mining graph patterns	1253.65	1480.47	4096.70	4118.38	13235.23	17290.18
Maximal/Closed extraction	0.10	0.25	0.12	0.23	1.35	2.95
Embedding all	35.25	71.10	7.78	23.64	270.25	780.95
Embedding closed	35.25	70.87	7.74	23.60	269.61	780.31
Embedding maximal	29.37	53.84	7.69	21.67	251.22	728.10
Total time	1353.71	1677.69	4120.15	4187.76	14029	19585.44

each entry captures the dissimilarity between the graph and each graph pattern mined. We employed REAFUM [20] to mine both exact and approximate graph patterns from the graph collections. Then, a post-processing step was implemented to identify from the whole set of graph patterns those that are closed and maximal graph patterns, preserving the graph patterns already mined and obtaining a subset of closed and maximal patterns. The sets of all frequent, closed, and maximal graph patterns were used to build the Dissimilarity-based Graph Embeddings. These embeddings were assessed using different supervised classifiers, including k-Nearest Neighbors (KNN), Linear Support Vector Machines (SVM), Decision Trees, Random Forest, Neural Networks, and Gradient Boosting Classifier (GBM), on four benchmark graph collections. The performance was evaluated using accuracy and F1-score metrics.

Compared to using all graph patterns, our experiments consistently yielded higher accuracy and F1-scores when using closed and maximal graph patterns. This suggests that compact and non-redundant subsets enhance the performance of dissimilarity-based graph embeddings for supervised classification. Moreover, employing approximate graph patterns resulted in better performance than embeddings built from exact graph patterns, outperforming them in both accuracy and F1-score. The study underscores the importance of further research on algorithms for mining approximate graph patterns, which allow variations in both labels and structure within graph collections, as well as algorithms that can mine closed and maximal graph patterns, since this approach reduces the number of patterns mined but also benefits if fewer resources are needed.

Additionally, the study points out the necessity of creating efficient graph pattern-mining algorithms that are suitable for dense and large graphs to ensure the scalability of our proposed approach.

As future work, we will explore the performance of Dissimilarity-based Graph Embeddings using other kinds of graph patterns. Additionally, we will perform a systematic exploration of the parameters and will expand our evaluation to a wider range of graph datasets to better assess the robustness and generalizability of the proposed approach. Moreover, we will extend our methodology by examining other types of embeddings.

Acknowledgments

This research was financially supported by the Secretaría de Ciencia, Humanidades, Tecnología e Innovación (SECIHTI) through the scholarship grant 778343.

References

1. **Acosta-Mendoza, N., Gago-Alonso, A., Carrasco-Ochoa, J. A., Martínez-Trinidad, J. F., Medina-Pagola, J. E. (2018).** Mining generalized closed patterns from multi-graph collections. **Mendoza, M., Velastín, S.**, editors, Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications, Springer International Publishing, Cham, pp. 10–18.

2. **Acosta-Mendoza, N., Gago-Alonso, A., Pagola, J. (2012).** Frequent approximate subgraphs as features for graph-based image classification. Knowledge-Based Systems, Vol. 17, pp. 381–392. DOI: 10.1016/j.knosys.2011.12.002.
3. **Aggarwal, C. (2010).** Managing and mining graph data. Springer, New York.
4. **Alam, M. T., Ahmed, C. F., Samiullah, M., Leung, C. K. (2021).** Discriminating frequent pattern based supervised graph embedding for classification. Advances in Knowledge Discovery and Data Mining: 25th Pacific-Asia Conference, PAKDD 2021, Virtual Event, May 11–14, 2021, Proceedings, Part II, Springer-Verlag, Berlin, Heidelberg, pp. 16–28. DOI: 10.1007/978-3-030-75765-6$_2$.
5. **Bunke, H., Riesen, K. (2011).** Improving vector space embedding of graphs through feature selection algorithms. Pattern Recognit., Vol. 44, No. 9, pp. 1928–1940.
6. **Cerruela-García, G., Cuevas-Muñoz, J. M., García-Pedrajas, N. (2022).** Graph-based feature selection approach for molecular activity prediction. Journal of Chemical Information and Modeling, Vol. 62, No. 7, pp. 1618–1632. DOI: 10.1021/acs.jcim.1c01578.
7. **Chen, X., Cai, J., Chen, G., Gan, W., Broustet, A. (2024).** Fcsg-miner: Frequent closed subgraph mining in multi-graphs. Information Sciences, Vol. 665, pp. 120363. DOI: https://doi.org/10.1016/j.ins.2024.120363.
8. **Flores-Garrido, M., Carrasco-Ochoa, J. A., Martínez-Trinidad, J. F. (2014).** Mining maximal frequent patterns in a single graph using inexact matching. Know.-Based Syst., Vol. 66, No. 1, pp. 166–177. DOI: 10.1016/j.knosys.2014.04.040.
9. **Flores-Garrido, M., Carrasco-Ochoa, J.-A., Martínez-Trinidad, J. (2015).** Agrap: an algorithm for mining frequent patterns in a single graph using inexact matching. Knowledge and Information Systems, Vol. 44, No. 2, pp. 385–406. DOI: 10.1007/s10115-014-0747-x.
10. **Holder, L. B., Cook, D. J., Djoko, S. (1994).** Substructure discovery in the subdue system. Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining, AAAI Press, pp. 169–180.
11. **Islam, M. A., Ahmed, C. F., Alam, M. T., Leung, C. K.-S. (2024).** Graph-based substructure pattern mining with edge-weight. Applied Intelligence, Vol. 54, No. 5, pp. 3756–3785. DOI: 10.1007/s10489-024-05356-7.
12. **Jaramillo-Olivares, D., Carrasco-Ochoa, J. A., Martínez-Trinidad, J. F. (2025).** Exact versus approximate patterns in dissimilarity-based graph embedding for supervised classification. **López-Monroy, A. P., Rosales-Pérez, A., Carrasco-Ochoa, J. A., Martínez-Trinidad, J. F., Olvera-López, J. A.**, editors, Pattern Recognition, Springer Nature Switzerland, Cham, pp. 56–67.
13. **Jaramillo-Olivares, D., Carrasco-Ochoa, J. A., Martínez-Trinidad, J. F. (2025).** An algorithm for mining frequent approximate subgraphs with structural and label variations in graph collections. Applied Sciences, Vol. 15, No. 14. DOI: 10.3390/app15147880.
14. **Jiang, C., Coenen, F., Zito, M. (2013).** A survey of frequent subgraph mining algorithms. The Knowledge Engineering Review, Vol. 28, No. 1, pp. 75–105. DOI: 10.1017/S0269888912000331.
15. **Kumar, S., Mallik, A., Khetarpal, A., Panda, B. (2022).** Influence maximization in social networks using graph embedding and graph neural network. Information Sciences, Vol. 607, pp. 1617–1636. DOI: https://doi.org/10.1016/j.ins.2022.06.075.
16. **Kuramochi, M., Karypis, G. (2001).** Frequent subgraph discovery. Proceedings 2001 IEEE international conference on data mining, IEEE, pp. 313–320.
17. **Li, L., Ding, P., Chen, H., Wu, X. (2022).** Frequent pattern mining in big social graphs. IEEE Transactions on Emerging Topics in Computational Intelligence, Vol. 6, No. 3, pp. 638–648. DOI: 10.1109/TETCI.2021.3067017.

18. **Li, M. M., Huang, K., Zitnik, M. (2022).** Graph representation learning in biomedicine and healthcare. Nature Biomedical Engineering, Vol. 6, No. 12, pp. 1353–1369. DOI: 10.1038/s41551-022-00942-x.

19. **Li, P. (2024).** Machine learning techniques for pattern recognition in high-dimensional data mining. 2024 4th International Conference on Electronic Information Engineering and Computer Communication (EIECC), IEEE, pp. 1493–1497.

20. **Li, R., Wang, W. (2015).** REAFUM: Representative Approximate Frequent Subgraph Mining. pp. 757–765. DOI: 10.1137/1.9781611974010.85.

21. **Molefe, M. E., Tapamo, J. R. (2024).** Classifying roads with multi-step graph embeddings. Comput. Sist., Vol. 28, No. 1.

22. **Mrzic, A., Meysman, P., Bittremieux, W., Moris, P., Cule, B., Goethals, B., Laukens, K. (2018).** Grasping frequent subgraph mining for bioinformatics applications. BioData mining, Vol. 11, No. 1, pp. 1–24.

23. **Nguyen, T. T., Huynh, T. T., Weidlich, M., Tho, Q. T., Yin, H., Aberer, K., Nguyen, Q. V. H. (2023).** Scalable maximal subgraph mining with backbone-preserving graph convolutions. Information Sciences, Vol. 644, pp. 119287. DOI: https://doi.org/10.1016/j.ins.2023.119287.

24. **Peng, H., Zhang, D. (2023).** Cfgm: An algorithm for closed frequent graph patterns mining. Information Sciences, Vol. 625, pp. 327–341. DOI: https://doi.org/10.1016/j.ins.2022.12.089.

25. **Preti, G., De Francisci Morales, G., Riondato, M. (2023).** Maniacs: Approximate mining of frequent subgraph patterns through sampling. ACM Trans. Intell. Syst. Technol., Vol. 14, No. 3. DOI: 10.1145/3587254.

26. **Proença, H. M., Grünwald, P., Bäck, T., van Leeuwen, M. (2022).** Robust subgroup discovery. Data Min. Knowl. Discov., Vol. 36, No. 5, pp. 1885–1970.

27. **Rage, U. K. (2025).** Finding useful patterns in graph databases. In Hands-on Pattern Mining. Springer Nature Singapore, Singapore, pp. 153–163.

28. **Ramraj, T., Prabhakar, R. (2015).** Frequent subgraph mining algorithms – a survey. Procedia Computer Science, Vol. 47, pp. 197–204. DOI: https://doi.org/10.1016/j.procs.2015.03.198. Graph Algorithms, High Performance Implementations and Its Applications (ICGHIA 2014).

29. **Rossi, R. A., Ahmed, N. K. (2015).** The network data repository with interactive graph analytics and visualization. AAAI, pp. .

30. **Salem, S., Alokshiya, M., Hasan, M. A. (2021).** RASMA: a reverse search algorithm for mining maximal frequent subgraphs. BioData Min., Vol. 14, No. 1, pp. 19.

31. **Sancheti, V., Thomas, L. T., Pudi, V. (2024).** Pmcs: Partition-based maximal frequent subgraph mining using mcs. 2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC), pp. 159–168. DOI: 10.1109/COMPSAC61105.2024.00032.

32. **Shaul, Z., Naaz, S. (2021).** cgspan: Closed graph-based substructure pattern mining. 2021 IEEE International Conference on Big Data (Big Data), pp. 4989–4998. DOI: 10.1109/BigData52589.2021.9671995.

33. **Swoboda, P., Kainmüller, D., Mokarian, A., Theobalt, C., Bernard, F. (2019).** A convex relaxation for multi-graph matching. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 11148–11157. DOI: 10.1109/CVPR.2019.01141.

34. **Tao, T., Wang, Q., Ruan, Y., Li, X., Wang, X. (2023).** Graph embedding with similarity metric learning. Symmetry, Vol. 15, No. 8. DOI: 10.3390/sym15081618.

Article received on 08/09/2025; accepted on 12/12/2025.
**Corresponding author is Daybelis Jaramillo-Olivares.*