Graph Clustering via Inexact Patterns

Marisol Flores-Garrido, Jesús Ariel Carrasco-Ochoa,
and José Fco. Martínez-Trinidad

Instituto Nacional de Astrofísica, Óptica y Electrónica
Luis Enrique Erro #1, 72840, Santa María Tonantzintla, Puebla, Mexico
{mflores,ariel,fmartine}@inaoep.mx

Abstract. Graph pattern mining is an important task in Data Mining and several algorithms have been proposed to solve this problem. Most of them require that a pattern and its occurrences are identical, thus, they rely on solving the graph isomorphism problem. In the last years, however, some algorithms have focused in the case in which label and edge structure differences between a pattern and its occurrences are allowed but maintaining a bijection among vertices, using inexact matching during the mining process. Recently, an algorithm that allows structural differences in vertices was proposed. This feature allows it to find patterns missed by other algorithms, but, do these extra patterns actually contain useful information? We explore the answer to this question by performing an experiment in the context of unsupervised mining tasks. Our results suggests that by allowing structural differences in both, vertices and edges, it is possible to obtain new useful information.

Keywords: Graph Clustering, Inexact Matching, Graph Mining.

1 Introduction

Graph pattern mining is an important Data Mining field, which is not surprising considering, on one hand, the importance that graphs have acquired as a modeling tool in a diversity of areas, from chemoinformatics [1] to web analysis [2], and, on the other hand, that finding frequent patterns can be an important step in tasks like characterize sets [3], classify [4], build indexes [5], etc.

Different algorithms have been proposed to mine graph patterns [6–9], but most of them have in common the fact that they require a pattern to be identical to any of its occurrences; to this end, algorithms usually rely on the graph isomorphism problem. But in the last years, some works have addressed the case in which some differences between a pattern and its occurrences should be allowed. For instance, motivated by the fact that sometimes data could be noisy and have some mislabeled vertices, Jia et al. [10] proposed an algorithm that uses probability matrices and the notion of *approximately isomorphic* graphs to identify occurrences that are not exactly the same as the pattern. Also, Chen et al. [11] considered that, in certain contexts, edges could be allowed to connect vertices in a different way (respect to the pattern) and introduced an algorithm that allows differences in edges. However, although these algorithms broaden the

E. Bayro-Corrochano and E. Hancock (Eds.): CIARP 2014, LNCS 8827, pp. 391–398, 2014.

notion of what an acceptable pattern is within certain contexts, they still require a bijection between the vertices of matching graphs. This is, every pattern vertex *must have* a correspondent vertex in a subgraph, and viceversa, in order to allow the subgraph to be considered an occurrence of the pattern.

In 2014, an algorithm, MaxAFG [12], that allows, besides label differences, structural differences in both edges *and vertices* was proposed. In this way, the algorithm is able to find patterns missed by other state-of-the-art algorithms. Clearly, allowing structural differences in vertices leads to a bigger number of discovered patterns, but, are they useful?

In this work, we explore the usefulness for clustering of those extra patterns found by MaxAFG. We rely on embedding graphs into a vector space via their dissimilarity respect to the patterns before applying a traditional clustering algorithm. Our experiments suggest that those extra patterns, found by allowing structural differences in vertices, do provide useful information about the analyzed data.

The rest of this work is organized as follows: in the next section related work is summarized. The main ideas of the algorithm MaxAFG are given in section 3. Our proposal for using the mined patterns for graph clustering is shown in section 4. In section 5, we show our experiments and, finally, in Section 6, we present our conclusions and some directions for future work.

2 Related Work

As mentioned before, several works have been proposed in the last years that focus on using inexact matching in graph mining.

First, we have algorithms that allow label differences. In 2011, Jia et al. [10] proposed the algorithm APGM, motivated by the scenario in which noisy data leads to wrong labels in vertices. APGM uses a substitution matrix, in which the entry ij represents the probability of label i being wrongly replaced by label j. The authors then, measure the similarity between graphs as the product of probabilities, and define two graphs as *approximately isomorphic* if their similarity is below a given threshold. Later, in 2012, Acosta et al. [13] proposed VEAM, which extends APGM to the case in which edge labels can be replaced, in addition to vertex labels.

As for allowing structural differences between graphs, in 2007, Chen et al. [11] introduced the algorithm gApprox, which finds patterns, in a single graph, that could have structural differences in edges. The algorithm computes the support of each pattern by using an upper bound of the support measure proposed by Kuramochi and Karypis [14, 15], whose idea is to avoid overlapping between occurrences.

Although these algorithms hint that relaxing the conditions for a graph to be considered a pattern occurrence could lead to useful information, they all require that two matching graphs have a bijection between their vertices. In order to explore a different scenario, in [16], we proposed MaxAFG which, to the best of our knowledge, is the only algorithm that allows structural differences in vertices.

3 MaxAFG Algorithm

The algorithm MaxAFG combines a dissimilarity function based on the graph edit distance, a search strategy to find occurrences of a given pattern allowing structural differences and a pattern-growth depth-first search scheme, to mine frequent graph patterns in a single graph. In this section, we briefly describe the main ideas of this algorithm.

Dissimilarity function. MaxAFG uses a dissimilarity function based on the edit distance. In order to compute the dissimilarity between two graphs, $g_1 = (V_1, E_1, L_1)$ and $g_2 = (V_2, E_2, L_2)$, it is required a one-to-one binary relation $m \subseteq V_1 \times V_2$. Then, the dissimilarity between g_1 and g_2 is given by

$$f_{dis}(g_1, g_2) = \kappa v_{edit} + (1 - \kappa) e_{edit}$$

where κ is the edit cost associated with vertices, and v_{edit} and e_{edit} are defined as follows:

$$v_{edit} = \sum_{v \in V_1 \setminus R_{V_1}} d_v(v, m(v)) + |R_{V_1}| + |R_{V_2}|$$

$$e_{edit} = \sum_{(u,v) \in E_1 \setminus R_{E_1}} d_e\left((u, v), (m(u), m(v))\right) + |R_{E_1}| + |R_{E_2}|.$$

In the v_{edit} expression, R_{V_1} represents the set of vertices in V_1 that are not related with any vertex in V_2, R_{V_2} represents the set of vertices in V_2 that are not related with any vertex in V_1, and $d_v(v_1, v_2)$ represents the cost of replacing $L_1(v_1)$ by $L_2(v_2)$. R_{E_1}, R_{E_2} and $d_e((u, v), (u', v'))$ are defined in an analogous way for the edge sets from g_1 and g_2. Thus, by including the terms R_{V_1} and R_{V_2}, this measure takes into account the case in which there are unmatched vertices between the graphs; in this way, structural differences in vertices are allowed. R_{E_1} and R_{E_2} are related with allowed structural differences in edges. Finally, $d_v(v_1, v_2)$ and $d_e((u, v), (u', v'))$ leave room for label differences.

Search Strategy. Besides the dissimilarity measure described, a search strategy was proposed to identify frequent patterns using inexact matching. In order to allow a pattern having occurrences in which there are "extra" vertices (respect to the pattern it represents), each time a new pattern P' is formed, as $P' = P \cup \{v\}$, and the occurrence g of the pattern P is trying to grow to a vertex u (matching v), the requirement of having an edge between g and u is replaced by the requirement of having a *path*, in a similar way that it is done in the algorithm GraMi [17], but considering inexact matching. On the other hand, in order to have occurrences with "missing" vertices respect to the pattern, the occurrences of P that could not grow are included in the occurrences of P', keeping track of the fact that there is a missing vertex and updating accordingly the edit cost. Structural differences in edges and labels are allowed following a strategy similar to that of gApprox [11].

Combining f_{dis} and the search strategy described before with a pattern-growth approach, MaxAFG allows finding maximal frequent approximate subgraphs in a single graph, allowing structural differences in vertices and edges, as well as label differences. For identifying maximal patterns, a pattern is saved only when it cannot be expanded into a new one that satisfies the frequency threshold. The support of a pattern is calculated using the function proposed by Bringmann and Nijssen [18], which defines the support of a pattern P in a graph G as

$$\sigma(P,G) = \min_{v \in V} |\{\varphi(v) : \varphi \text{ is a mapping between } P \text{ and one of its occurrences in } G\}|,$$

i.e., for each node v in P it is found the number of nodes in G to which v is mapped to; then, the support of P is established as the minimum of these numbers.

More details of MaxAFG, including complexity, can be found in [16].

4 Pattern-Based Clustering

In order to explore the usefulness of the set of mined patterns, $\mathcal{M}$, in a graph clustering task, we embedded the graphs in a vector space via their dissimilarity respect to the patterns in $\mathcal{M}$.

Usually, graphs are embedded in a vector space through a set of features, but, according to Duin and Pekalska [19], in human recognition processes it is more natural to rely on similarities or dissimilarities between objects than to find explicit features of the objects that could be used for recognition (an idea that led to prototype-based embeddings [20]).

Thus, we represent each graph g as a vector v_g in an n-dimensional space, where $n = |\mathcal{M}|$ and the ith element of v_g is the dissimilarity between g and the ith pattern in $\mathcal{M}$; $i = 1, \ldots, n$. We expect that two graphs with similar structure have small (or large) dissimilarities respect to the same patterns, thus, embedding the graphs in this way somehow preserves the structural similarities among the graphs.

To measure the dissimilarity between graphs and patterns, we used f_{dis}. Once graphs have been embedded into vectors, traditional clustering algorithms can be applied.

5 Experiments

In order to show the usefulness of the patterns found allowing structural differences in vertices and edges, we compared the clustering results against those obtained when the patterns mined by gApprox [11] are used to embed the graphs, since gApprox is the the closest algorithm we could find in the literature. Because we use maximal patterns and gApprox does not focus in them, we mined

the patterns using gApprox and, then, we selected the maximal ones in a post-mining step. In both cases, with MaxAFG and gApprox, we used f_{dis} in the mining and embedding steps; since what we wanted to find out is if allowing structural differences in vertices leads to discover information, implicit in the mined patterns, that could be useful in clustering. For clustering, we used the well-known *k-means* algorithm, with random initialization; the value of k was chosen according to the number of classes in the used databases.

In our experiments, we used two graph databases:

- **Structural image skeleton database.** Composed by 36 graphs representing the skeleton of real-image silhouettes [21]. The database is divided in 9 classes (elephant, fork, heart, horse, human, L-star, star, tortoise and whale, Fig. 1); each class has 4 graphs. Vertices are labeled with body parts, while edges are labeled with the distance between the vertices they connect. The collection has 13 different vertex labels, 211 edge labels, an average graph order of 7 and an average graph size of 6.
- **Image database.** It was created by Acosta et al. [13] and consists of graphs built from images, following a quadtree strategy, so that each graph represents an image and the labels, in vertices and edges, represent colors and angles associated to the image. Images, and therefore graphs, are labeled as "landscape" (l) or "seascape" (s). Fig. 2 (taken from [22]) shows an example of an image in this database and its associated graph. From this database, we chose 200 graphs, 100 from each class, with the average graph size and order being 20 and 35, respectively.

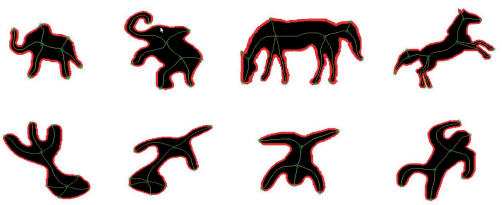


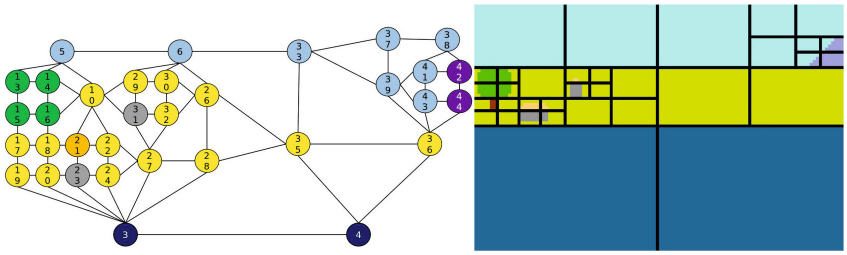
Fig. 1. Example of the images in the SIS database

Since we knew the graph labels, we used the true labels in order to validate our clustering results. In this step we used the General Rand Index [23], which, given the clusterings C and C' over a set S, is defined as

$$R(C, C') = \frac{2(|S_{11}| + |S_{00}|)}{|S|(|S| - 1)},$$

where S_{11} is the set of pairs that are in the same cluster under C and C', and S_{00} is the set of pairs that are in different clusters under C and C'.

The experiments were performed in a computer with an Intel Core i7 (3.4 GHz) processor with 32GB in RAM, running the Ubuntu 12.04 distribution of



(a) (b)

Fig. 2. Graph from the database used in the classification experiment; in the left it is shown the graph whereas in the right it is shown the image represented by the graph

Table 1. Results when clustering using patterns mined by gApprox and MaxAFG, respectively. The clustering quality is measured using the General Rand Index [23].

	Skeleton DB	Image DB
gApprox	.955	.676
MaxAFG	.981	.768

the 64 bit GNU Linux operating system. The implementation of the proposed algorithm and gApprox was done in Python, using the NetworkX library.

The Table 1 shows the best result that we obtained after running the experiment 100 times, measuring the clustering quality with the General Rand Index. As we can see, in both databases, we find an improvement when using the patterns mined by MaxAFG.

The essential difference between gApprox and MaxAFG is that, while both allow label differences and structural differences in edges, the former does not allow structural differences in vertices. As argued in [16], all of the patterns found by gApprox, including the maximal ones, are subgraphs of the patterns found by MaxAFG. Thus, the quality improvement that we see in the experiment results could be directly attributed to those (larger) patterns identified by MaxAFG. Considering the experiment results, we conclude that the patterns found by MaxAFG are indeed able to capture useful information about the input graph, missed when a bijection between the vertices of matching graphs is required.

6 Conclusions and Future Work

In this work, we explored the usefulness, in the context of graph clustering, of patterns found by the algorithm MaxAFG. MaxAFG uses inexact matching and allows structural differences in vertices and edges. Although it finds patterns missed by other state-of-the-art algorithms, the question rises about the usefulness of these extra patterns. We used MaxAFG to analyze a collection of graphs and, then, used the found patterns to embed graphs into a vector space and find clusters.

We found that using the patterns mined by MaxAFG resulted in a better clustering quality, in terms of the general Rand Index, than using the maximal obtained from the patterns mined by gApprox, an algorithm that allows structural differences in edges but not in vertices. Although the obtained results are not outstanding from the clustering point of view, they do show that allowing structural differences in vertices can lead to useful knowledge providing a more insightful analysis of graph data.

A line suggested for further study is reducing the size of the embedding vector space, either by previously choosing a subset of patterns or by using a method for dimensionality reduction. Also, the idea of finding clusters without embedding the graphs into a vector space could be explored.

Acknowledgements. This work was partly supported by the National Council of Science and Technology of Mexico (CONACyT) through the project grants CB2008-106443 and CB2008-106366; and the scholarship grant 256879.

References

1. Mahé, P., Ueda, N., Akutsu, T., Perret, J.: Graph kernels for molecular structure-activity relationship analysis with support vector machines. Journal of Chemical Information and Modeling 45(4), 939–951
2. Schenker, A., Last, M., Bunke, H., Kandel, A.: Classification of web documents using graph matching. International Journal of Pattern Recognition and Artificial Intelligence 18(3), 475–496 (2004)
3. Borgelt, C., Berthold, M.R.: Mining molecular fragments: Finding relevant substructures of molecules. In: Proceedings. 2002 IEEE International Conference on Data Mining, pp. 51–58 (2002)
4. Deshpande, M., Kuramochi, M., Wale, N., Karypis, G.: Frequent substructure-based approaches for classifying chemical compounds. IEEE Trans. Knowl. Data Eng. 17(8), 1036–1050 (2005)
5. Yan, X., Yu, P.S., Han, J.: Graph indexing: A frequent structure-based approach. In: SIGMOD Conference, pp. 335–346. ACM (2004)
6. Ranu, S., Singh, A.K.: Graphsig: A scalable approach to mining significant subgraphs in large graph databases. In: IEEE 25th International Conference on Data Engineering, pp. 844–855 (2009)
7. Nijssen, S., Kok, J.N.: A quickstart in frequent structure mining can make a difference. In: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2004, pp. 647–652. ACM (2004)
8. Yan, X., Han, J.: Gspan: Graph-based substructure pattern mining. In: Proceedings of the 2002 IEEE International Conference on Data Mining, ICDM 2002, pp. 721–724. IEEE Computer Society (2002)
9. Gago Alonso, A., Medina Pagola, J.E., Carrasco-Ochoa, J.A., Martínez-Trinidad, J.F.: Mining frequent connected subgraphs reducing the number of candidates. In: Daelemans, W., Goethals, B., Morik, K. (eds.) ECML PKDD 2008, Part I. LNCS (LNAI), vol. 5211, pp. 365–376. Springer, Heidelberg (2008)
10. Jia, Y., Zhang, J., Huan, J.: An efficient graph-mining method for complicated and noisy data with real-world applications. Knowl. Inf. Syst. 28(2), 423–447 (2011)

11. Chen, C., Yan, X., Zhu, F., Han, J.: Gapprox: Mining frequent approximate patterns from a massive network. In: ICDM, pp. 445–450. IEEE Computer Society (2007)
12. Flores-Garrido, M., Carrasco-Ochoa, J.A., Martínez-Trinidad, J.F.: AGraP: an algorithm for mining frequent patterns in a single graph using inexact matching. In: Knowl. Inf. Syst. (2014), doi:10.1007/s10115-014-0747-x
13. Acosta-Mendoza, N., Gago-Alonso, A., Medina-Pagola, J.E.: Frequent approximate subgraphs as features for graph-based image classification. Knowledge-Based Systems 27, 381–392 (2012)
14. Kuramochi, M., Karypis, G.: Grew - a scalable frequent subgraph discovery algorithm. In: Proceedings of the Fourth IEEE International Conference on Data Mining, pp. 439–442 (2004)
15. Kuramochi, M., Karypis, G.: Finding frequent patterns in a large sparse graph. Data Min. Knowl. Discov. 11(3), 243–271 (2005)
16. Flores-Garrido, M., Carrasco-Ochoa, J.A., Martínez-Trinidad, J.F.: Mining maximal frequent patterns in a single graph using inexact matching. Knowledge-Based Systems 66, 166–177 (2014)
17. Saeedy, M.E., Kalnis, P.: GraMi: generalized frequent pattern mining in a single large graph. Technical report, Division of Mathematical and Computer Sciences and Engineering, King Abdullah University of Science and Technology (2011)
18. Bringmann, B., Nijssen, S.: What is frequent in a single graph? In: Washio, T., Suzuki, E., Ting, K.M., Inokuchi, A. (eds.) PAKDD 2008. LNCS (LNAI), vol. 5012, pp. 858–863. Springer, Heidelberg (2008)
19. Pekalska, E., Duin, R.P.W.: The Dissimilarity Representation for Pattern Recognition: Foundations and Applications. Machine Perception and Artificial Intelligence. World Scientific (2005)
20. Foggia, P., Percannella, G., Vento, M.: Graph matching and learning in pattern recognition in the last 10 years. International Journal of Pattern Recognition and Artificial Intelligence 28(01), 1450001 (2014)
21. Pinilla-Buitrago, L.A., Martínez-Trinidad, J.F., Carrasco-Ochoa, J.A.: New penalty scheme for optimal subsequence bijection. In: Ruiz-Shulcloper, J., Sanniti di Baja, G. (eds.) CIARP 2013, Part I. LNCS, vol. 8258, pp. 206–213. Springer, Heidelberg (2013)
22. Acosta-Mendoza, N.: Clasificación de imágenes basada en subconjuntos de subgrafos frecuentes aproximados. Master's thesis, Instituto Nacional de Astrofísica, Óptica y Electrónica, Puebla, México (2013)
23. Rand, W.M.: Objective criteria for the evaluation of clustering methods. Journal of the American Statistical Association 66(336), 846–850 (1971)