Large-scale Graph Indexing using Binary Embeddings of Node Contexts

Pau Riba, Josep Lladós, Alicia Fornés, and Anjan Dutta

Computer Vision Center - Computer Science Department
Universitat Autònoma de Barcelona, Spain
{priba,josep,afornes}@cvc.uab.es
http://www.cvc.uab.es

Abstract. Graph-based representations are experiencing a growing usage in visual recognition and retrieval due to their representational power in front of classical appearance-based representations in terms of feature vectors. Retrieving a query graph from a large dataset of graphs has the drawback of the high computational complexity required to compare the query and the target graphs. The most important property for a large-scale retrieval is the search time complexity to be sub-linear in the number of database examples. In this paper we propose a fast indexation formalism for graph retrieval. A binary embedding is defined as hashing keys for graph nodes. Given a database of labeled graphs, graph nodes are complemented with vectors of attributes representing their local context. Hence, each attribute counts the length of a walk of order k originated in a vertex with label l. Each attribute vector is converted to a binary code applying a binary-valued hash function. Therefore, graph retrieval is formulated in terms of finding target graphs in the database whose nodes have a small Hamming distance from the query nodes, easily computed with bitwise logical operators. As an application example, we validate the performance of the proposed methods in a handwritten word spotting scenario in images of historical documents.

Keywords: Graph matching, graph indexing, application in document analysis, word spotting, binary embedding

1 Introduction

The practical success of machine learning methods applied to simple image representations faded away other schemes representationally richer but practically unfeasible. However, to tackle with complex recognition problems, methods not exclusively based on appearance but enriched with more abstract visual information, such as visual structure of objects, are required. Although the first attempts of part-based descriptors suggesting graph representations were presented long ago [10], it has been in the last decade when a resurgence of structural models has been perceived in computer vision. Graph representations are implicitly or explicitly drivers of more powerful approaches for visual recognition and retrieval.

Graphs are robust representations offering a representation paradigm able to deal with many-to-many relationships among visual features and their parts. The use of graph matching is an effective solution to deal with visual recognition. Graph matching is among the most important challenges of graph processing. Roughly speaking, the problem consists in finding the best correspondence between the sets of vertices of two graphs preserving the underlying structures and the corresponding labels and attributes. Graph matching plays an important role in many applications of computer vision and pattern recognition, and several algorithms have been proposed in the literature [4]. One of the most popular error-tolerant graph matching methods is based on graph edit distance [19]. However the error-tolerant nature involves an inexact (sub)graph isomorphism computation which is a known NP-Complete problem. Consequently, methods based on graph edit distance are only applicable to graphs of small size. Approximate or suboptimal variations of graph edit distance have been proposed to overcome this difficulty [16]. In the last years new approaches based on graph embeddings and graph kernels have emerged rapidly [6, 15]. These methods are based on finding an explicit or implicit transformation of the graph to a n-dimensional space so the problem of graph similarity is elegantly reduced to a machine learning problem using classical classification schemes (e.g. SVM). Other solutions to reduce the complexity of graph matching are based on graph serialization [7, 17] consisting in transforming the graph to a sequence so the problem can be solved by an alignment algorithm in quadratic time. More recently, Zhou et al. proposed an efficient approach based on graph factorization [23] applied to deformable object recognition and alignment.

Although the existence of many suboptimal methods for graph matching, the scalability to large scale scenarios is still a challenge. The huge increase of user-generated contents (e.g. image repositories and videos in social networks) has resulted in a need for services including algorithms for searching by content in large databases. As stated before, structural information can play an important role in developing such tools for content-based image retrieval (CBIR). In terms of the complexity of graph matching, it can not be solved by comparing a query graph with thousand or million graphs of the database in a sequential way. Graph indexing approaches must be introduced. This is the motivation of the work presented in this paper.

The problem of graph-based indexing or hashing has been addressed in the literature, especially from the application point of view. In general, it is solved by graph factorization techniques where the database of graphs is decomposed in smaller ones that represent a codebook of compounding ones. The indexation is therefore stated in terms of indexing the constituent graphs organized in a lookup table structure. Messmer in [13] proposed an approach where the constituent graphs are organized in a decision tree. At run time, subgraph isomorphisms are detected by means of decision tree traversal. The complexity for indexing is polynomial in the number of input graph vertices, but the decision tree is of exponential size. A similar approach based on the construction of a graph lattice was proposed in [20]. The performance for large scale retrieval is achieved

by matching many overlapping and redundant subgraphs. In [3] an indexing technique for graph databases is proposed. It is based on constructing a nested inverted-index, called FG-index, based on the set of frequent subgraphs. Some works have explored the use of local substructures for indexing. Hence, Yan et al. [22] proposed a graph-based index in terms of frequent subgraphs.

Binary codes are compact descriptors that capture the local context of an image keypoint, according to a local neighborhood pattern, and represent it with a vector of bits. One of the most promising local descriptors is the efficient BRIEF descriptor [2]. BRIEF is a binary descriptor that aims at quickly comparing local features while requiring few amounts of memory. The BRIEF descriptor outputs a set of bits obtained by comparing intensities of pairs of pixels within the local key-region. The good property of binary codes is that, since they are represented as vectors of bits, the comparison between two of them can be quickly computed with basic logical operations (usually XOR) using directly the features of the CPU.

In this paper we propose a graph hashing approach inspired by the ideas of binary encoding for CBIR. We propose to extend the attributes associated to graph nodes by an embedding function describing the local context of the node. By local context we mean the structure of a subgraph centered at the node of radius k (the radius means the length of the path to the farthest node). This vector of attributes is converted to a binary code applying a binary-valued hash function. Therefore, graph retrieval is formulated in terms of finding target (sub)graphs in the database whose nodes have a small Hamming distance from the query nodes. This indexation based on binary codes can be easily computed with bitwise logical operators (XOR) taking advantage of the hardware benefits.

The rest of this paper is organized as follows: in Section 2 we describe the scientific contribution of our work. Section 3 presents an application example where our proposed graph indexing approach is applied to the problem of handwritten word spotting in historical documents. Finally Section 4 draws the conclusion.

2 Binary Embedding Formulation

In this section we describe the main contribution of this work consisting in the encoding of the local topological context of graph nodes by binary vectors. It allows to construct a fast indexing scheme in terms of the Hamming distance.

2.1 Binary topological node features

An *attributed graph* G is defined as a 4-tuple. $G = (V, E, L_V, L_E)$ where V is the set of nodes; $E \subseteq V \times V$ is the set of edges; L_V and L_E are two labeling functions defined as $L_V : V \to \Sigma_V \times A_V^k$ and $L_E : E \to \Sigma_E \times A_E^l$, where Σ_V and Σ_E are two sets of symbolic labels for vertices and edges, respectively, A_V and A_E are two sets of attributes for vertices and edges, respectively, and $k, l \in \mathbb{N}$. We will denote the number of vertices in a graph by $|V|$ and the number of edges by $|E|$.

An embedding function $\phi : \mathcal{G} \rightarrow \mathbb{R}^n$ transforms a graph $G \in \mathcal{G}$ to an n-dimensional feature vector. Hence, the distance between two graphs can be computed by a distance in a metric space, and the problem of graph classification can be solved by a statistical learning approach.

The *Morgan index* M is a node feature, originally used to characterize chemical structures [14], that computes the node context in terms of its local context. This index is iteratively computed for each node $v \in V$ as follows:

$$M(v,k) = \begin{cases} 1 & \text{if } k = 0; \\ \sum_u M(u, k-1) & \text{otherwise.} \end{cases}$$

where u is a vertex adjacent to v. The Morgan index of order k associated to a given node v $M(v,k)$ counts the number of paths of length k incident to node v and starting somewhere in the graph. The Morgan index can be computed by the values of the exponentiation of the adjacency matrix. An interesting property of the adjacency matrix A of any graph G is that the (i,j)th entry of A^n denotes the number of walks of length n from the node v_j to the node v_i. Therefore, the Morgan index of order k a node v_i is equivalent to the sum of the cells of the i-th row of the matrix A^k, formally $M(v_i,k) = \sum_j A^k(i,j),\ j = 1 \dots |V|$.

Inspired by the topological node features proposed by Dahm et al. [5] we define the *context* of a node v as a node embedding function computed in terms of the topological information of a subgraph centered at v. This context is described in terms of the Morgan index, but it is enriched taking into account the labels of the neighboring nodes. Hence, let us define a variation of the Morgan index concept as follows. Let us denote as $M_l(v,k)$ the Morgan index of node v, order k and label l which counts the number of paths of length k incident at node v and starting at nodes labeled as l. According to this, the *context* of a node v is formally defined as:

$$\nu(v) = [M_{l_1}(v,1), \dots, M_{l_1}(v,K), M_{l_2}(v,1), \dots, M_{l_2}(v,K), \dots, M_{l_{|\Sigma_V|}}(v,K)],$$

where K is the maximum length of the paths incident in v that is considered. The value of K is dependent of each experimental setup. In the application case described in Section 3.3 we have set $K = 3$. Thus, every graph node is attributed by a $K \cdot |\Sigma_V|$ feature vector characterizing the number of paths incident at v of lengths up to K and starting at nodes for all the possible labels in Σ_V.

The context vector $\nu(v)$ is converted to a binary code $\hat{\nu}(v) = \{0,1\}^{K \cdot |\Sigma_V|}$ in terms of a list of corresponding threshold values T_i. These values are application dependent, and in the use case described in Section 3.3 are set to the mean of each dimension.

Figure 1 illustrates the computation of the binary codes. In this example, the codes associated to nodes have length 6 ($K = 3$ and $|\Sigma_V| = 2$). The threshold value is set to the mean of each $M_l(v,k)$.

2.2 Indexing

Given a query graph G_q and a database of graphs $\{G_1, \dots, G_T\}$, a *focused graph retrieval* problem is defined as finding the subgraphs of G_i similar to G_q. Thus, it

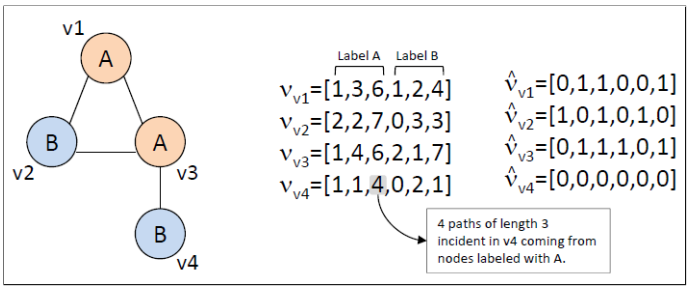


Fig. 1. Example of the binary code computation from a graph.

consists in finding inexact subgraph matchings between the query and the target graphs. The graph indexing scheme proposed in this paper follows the paradigm of focused retrieval. In terms of a visual retrieval application, this process can be understood as not only retrieving the images of a database where a query object is likely to appear, but finding the position in each retrieved image. Our proposed graph indexing approach follows this objective.

An inverted file indexing architecture in terms of node contexts is constructed. It stores a mapping from the binary topological features to the nodes of the target graphs in the database. This inverted file is therefore formulated as a lookup table $H : \{0,1\}^b \rightarrow \{v_i\}_{v_i \in V}$ that indexes a b-bit vector and returns a list of nodes whose context (binary code) is similar to the input code.

The last step is the actual subgraph matching process. With the indexing table H we only retrieve individual nodes, so it is necessary to implement a node consistency verification. With this purpose, we define a *partition* P of a graph G as a decomposition of it in n small subgraphs, $P(G) = \{g_1, \ldots, g_n\}$, where $g_i \subseteq G$. Hence, the lookup table H is reformulated as a hashing function that instead of returning nodes similar to the input binary code, it returns subgraphs where these target nodes appear. Formally, given a query graph G_q and a database of graphs $\{G_1, \ldots, G_T\}$, for each node of the query graph $v \in V_q$, the indexation function H returns the subgraphs of the database, after a partition has been previously defined, containing this vertex $H(v) = \{g_i\}$, where g_i is a subgraph of one of the target graphs $\{G_1, \ldots, G_T\}$. The definition of the partition under which the database of graphs is decomposed in small graphs is application dependent. The subgraphs g_i can be seen as voting bins, according to a Hough-based principle. Thus, the final result consists of the subgraphs receiving a high number of votes.

Concerning the practical implementation of H that computes the Hamming distance between binary codes, the most straightforward solution is a brute-force linear scan, i.e. to compute the Hamming distance between the query vector and each vector in the database. Computing the Hamming distance between two vectors consists in computing the XOR and counting the number of 1's in the resulting vector. This computation can be computed very fast on modern CPU's,

with logic operations being part of the instruction set. A fast hashing process like Locality Sensitive Hashing (LSH) [11] can be added to speed up the indexation.

3 Application Example

This section experimentally illustrates the graph indexing approach with a practical application consisting in word spotting in historical manuscripts.

3.1 Handwritten word spotting

The preservation of historical handwritten document collections is of key importance for archives, museums and libraries. Their goal is not only the digitization of paper documents, but also the extraction of the information that these documents contain towards the creation of digital libraries. Since the cost of the manual transcription by human experts is prohibitive, the challenge is to enable the automatic extraction of information through document image analysis techniques.

Since handwriting recognition techniques require large amounts of annotated images to train the recognizer, word spotting is a viable solution to make historical manuscripts amenable to searching and browsing, especially when training data is difficult to obtain. Word spotting is defined as the task of retrieving all the instances of a given query word. In this scenario, the user selects one by looking at the documents, and the system retrieves all words with a similar shape. The first advantage is that word spotting can be performed on-the-fly: the user can crop a word in a new document collection, and the system searches for similar words without any training step. The second advantage is that, since the query word is treated as a shape, these approaches are also able to retrieve graphical elements, such as stamps, symbols, or seals.

Most existing word spotting techniques use statistical representations (e.g. SIFT, HOG) of the word images [1, 18]. However, there also exists few approaches using structural representations. The main motivation is that the nature of handwriting suggests than the structure is more stable than the pure appearance of the handwritten strokes. This is specially important when dealing with the elastic deformations of different handwriting styles.

As stated in the comparison of statistical versus structural representations for handwritten word spotting reported in [12], the main disadvantages of structural approaches are the time complexity and scalability to large document collections. Although some methods [9] only use the graph nodes (avoiding the edges), and other approaches [21] propose an embedding using a bag of graphlets (codebook of small graphs, with order 2 or 3), these approaches are still far away of being able to cope with large databases in an efficient way.

However, we believe that the graph indexing using binary embeddings proposed in this paper can be the key to make the graph-based word spotting approaches comparable to statistical ones in terms of time and memory requirements for large document collections.

3.2 Experimental setup

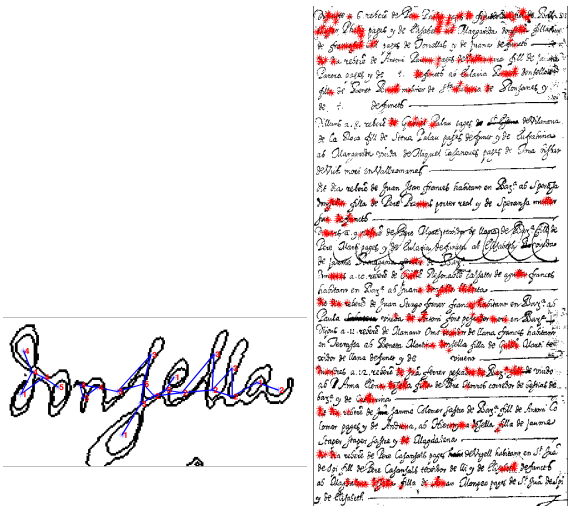
For the experiments, we have used some pages from the Barcelona Historical Handwritten Marriages Database (BH2M) [8]. It consists of 174 images of manuscripts from the 17th century. The images are part of a collection of marriage records from the archive of the Barcelona Cathedral. The use of word spotting in this collection allows to search names, places, occupations, etc. To illustrate the performance of the method proposed in this paper, 11 pages and 5 instances of 8 query words have been selected. Images are represented by attributed graphs where nodes correspond to basic primitives (graphemes) like loops, vertical lines, arcs, etc. and nodes represent adjacency relations between primitives. This results in 40 query graphs and 11 graphs corresponding to the database images. It has to be noticed that the graphs corresponding to pages of documents contain several connected graphs (between 200-300 in average) corresponding to words or parts of them. In total, the 11 pages contain 3,609 words. In terms of size, query graphs have an average of 25 nodes, and a graph representing a page of a document has an average of 4,500 nodes. If the whole database is considered as a unique large graph, it consists of 50,556 nodes. The partition of the database to define the voting bins is roughly associated to possible words in the images in terms of bounding boxes of connected components. The generation of page graphs takes long time, but it is computed off-line. The extraction of graphemes and the construction of the corresponding graph has a complexity of $O(n^2)$. Concerning the setup of the method, the node contexts have been computed with the paths up to order $K = 3$, and the number of possible node labels is $|\Sigma_V| = 10$. Thus, the length of binary vectors is 30.

3.3 Results

To visually assess the performance of the method, the result of a query word graph is shown in Fig. 2. Figure 2(a) shows a query word and the corresponding graph. Figure 2(b) illustrates the locations where query nodes are detected. It can be appreciated that in the locations where a true positive exists there is a higher density of votes.

In Table 1 we quantitatively report the performance metrics. For each query word, we show the figures of averaging the 5 query instances. For each query, we show the precision, recall and F1-score measures. We can observe that we are obtaining a quite high recall values, i.e. most of the true positives are retrieved, however the precision is quite low, so there is a high number of false positives. It must be noticed that these values depend on the acceptance threshold that is set to consider a retrieval as correct. The graph indexation presented in this paper must not be seen as a final graph matching but a coarse step to quickly locate subgraphs of the database likely to match to the query graph. However, a more accurate matching should be done afterwards with the retrieved subgraphs.

In terms of computational cost, although the implementation is not optimized, the elapsed time for indexing a graph corresponding to a page is 0,02



(a) (b)

Fig. 2. Qualitative results: (a) A query word and its corresponding graph; (b) A full page and the locations where query nodes are detected.

seconds (target graph of 4,500 nodes). The elapsed time of a standard implementation of a bipartite graph-matching method is 1,02 seconds. Hence, the time is drastically reduced.

4 Conclusions

In this paper we have presented an approach for computing fast inexact subgraph matching for large scale retrieval purposes. The main contribution of the proposed approach is the definition of a binary embedding for graph nodes based on the called local context. The node context has been defined as the topology of the paths of order k incident in the node and coming from nodes of a given label. A hashing architecture has been designed using binary codes as indexation keys. An application consisting in word spotting into historical handwritten document images has been used as experimental scenario. Although the results are in a preliminary stage, they are encouraging. In terms of a retrieval problem, high recall values are obtained, although the precision is low. The time complexity is linear in terms of the number of nodes of the database. It leads us to conclude that a graph indexation scheme as it is proposed is very useful to compute inexact subgraph matchings in large-scale scenarios as a filtering step aiming to prune the database, so a more accurate matching method can be computed afterwards only in the retrieved subgraphs. Finally, in terms of the application, we have demonstrated that compact structural descriptors are useful signatures for handwriting recognition, despite the variability of handwriting.

Query	Transcription	Precision	Recall	F1-score
	Eularia	0.0080	0.8462	0.0158
	Hieronyma	0.0118	0.7875	0.0232
	Jua$	0.0149	0.5389	0.0291
	defunct	0.0271	0.7886	0.0524
	donsella	0.0420	0.8215	0.0796
	pages	0.0590	0.9352	0.1107
	rebere$	0.0645	0.7676	0.1187
	viudo	0.0133	0.6455	0.0261
	Total	**0.0301**	**0.7664**	**0.0569**

Table 1. Quantitative results of word spotting based on graph indexing.

Acknowledgments

This work has been partially supported by the Spanish project TIN2012-37475-C02-02 and the European project ERC-2010-AdG-20100407-269796.

References

1. J. Almazán, A. Gordo, A. Fornés, and E. Valveny. Word spotting and recognition with embedded attributes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(12):2552–2566, Dec 2014.
2. M. Calonder, V. Lepetit, C. Strecha, and P. Fua. BRIEF: Binary robust independent elementary features. In *Computer Vision, European Conference on*, volume 6314 of *Lecture Notes on Computer Science*, pages 778–792. 2010.
3. J. Cheng, Y. Ke, W. Ng, and A. Lu. Fg-index: Towards verification-free query processing on graph databases. In *Management of Data, International Conference on*, SIGMOD '07, pages 857–872, New York, NY, USA, 2007. ACM.
4. D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. *Pattern Recognition and Artificial Intelligence, International Journal of*, 18(03):265–298, 2004.
5. N. Dahm, H. Bunke, T. Caelli, and Y. Gao. A unified framework for strengthening topological node features and its application to subgraph isomorphism detection. In *Graph-Based Representations in Pattern Recognition*, volume 7877 of *Lecture Notes in Computer Science*, pages 11–20. Springer Berlin Heidelberg, 2013.
6. F.X. Dupé and L. Brun. Edition within a graph kernel framework for shape recognition. In *Graph-Based Representations in Pattern Recognition*, volume 5534 of *Lecture Notes in Computer Science*, pages 11–20. Springer Berlin Heidelberg, 2009.

7. A. Dutta, J. Llados, and U. Pal. A symbol spotting approach in graphical documents by hashing serialized graphs. *Pattern Recognition*, 46(3):752 – 768, 2013.
8. D. Fernández-Mota, J. Almazán, N. Cirera, A. Fornés, and J. Lladós. Bh2m: The barcelona historical, handwritten marriages database. In *Pattern Recognition, 22nd International Conference on*, pages 256–261. IEEE, 2014.
9. A. Fischer, C.Y. Suen, V. Frinken, K. Riesen, and H. Bunke. A fast matching algorithm for graph-based handwriting recognition. In *Graph-Based Representations in Pattern Recognition*, pages 194–203. Springer, 2013.
10. M.A. Fischler and R.A. Elschlager. The representation and matching of pictorial structures. *Computers, IEEE Transactions on*, C-22(1):67–92, Jan 1973.
11. P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pages 604–613. ACM, 1998.
12. J. Llados, M. Rusinol, A. Fornes, D. Fernandez, and A. Dutta. On the influence of word representations for handwritten word spotting in historical documents. *Pattern Recognition and Artificial Intelligence, International Journal of*, 26(05), 2012.
13. B.T. Messmer and H. Bunke. A decision tree approach to graph and subgraph isomorphism detection. *Pattern Recognition*, 32(12):1979 – 1998, 1999.
14. H. L. Morgan. The generation of a unique machine description for chemical structures-a technique developed at chemical abstracts service. *Chemical Documentation, Journal of*, 5(2):107–113, 1965.
15. M. Neuhaus and H. Bunke. *Bridging the Gap Between Graph Edit Distance and Kernel Machines*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2007.
16. K. Riesen and H. Bunke. Approximate graph edit distance computation by means of bipartite graph matching. *Image and Vision Computing*, 2009.
17. A. Robles-Kelly and E.R. Hancock. Graph edit distance from spectral seriation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(3):365–378, March 2005.
18. M. Rusiñol, D. Aldavert, R. Toledo, and J. Lladós. Efficient segmentation-free keyword spotting in historical document collections. *Pattern Recognition*, 48(2):545–555, 2015.
19. A. Sanfeliu and K.S. Fu. A distance measure between attributed relational graphs for pattern recognition. *Systems, Man and Cybernetics, IEEE Transactions on*, SMC-13(3):353–362, May 1983.
20. E. Saund. A graph lattice approach to maintaining and learning dense collections of subgraphs as image features. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(10):2323–2339, October 2013.
21. P. Wang, V. Eglin, C. Garcia, C. Largeron, J. Lladós, and A. Fornés. A coarse-to-fine word spotting approach for historical handwritten documents based on graph embedding and graph edit distance. In *Pattern Recognition, 22nd International Conference on*, pages 3074–3079. IEEE, 2014.
22. X. Yan, P.S. Yu, and J. Han. Graph indexing: a frequent structure-based approach. In *Management of data, Proceedings of the 2004 ACM SIGMOD international conference on*, pages 335–346. ACM, 2004.
23. Feng Zhou and F. De la Torre. Factorized graph matching. In *Computer Vision and Pattern Recognition, IEEE Conference on*, pages 127–134, June 2012.