Pattern Recognition Letters 33 (2012) 2011–2019

Contents lists available at SciVerse ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Faster subgraph isomorphism detection by well-founded total order indexing

Markus Weber [a,*], Marcus Liwicki [a], Andreas Dengel [a,b]

[a] German Research Center for Artificial Intelligence (DFKI) GmbH, Trippstadter Straße 122, 67663 Kaiserslautern, Germany
[b] Knowledge-Based Systems Group, Department of Computer Science, University of Kaiserslautern, P.O. Box 3049, 67653 Kaiserslautern, Germany

ARTICLE INFO

Article history:
Available online 11 May 2012

Keywords:
Graph isomorphism
Subgraph isomorphism
Tree search
Decision tree
Indexing

ABSTRACT

In this paper an extension to index-based subgraph matching is proposed. This extension significantly speeds up the indexing time for graphs where the nodes are labeled with a rather small amount of different classes. Furthermore, the needed storage amount is significantly reduced. In order to reduce the complexity, we introduce a weight function for labeled graphs. Using this weight function, a well-founded total order is defined on the weights of the labels. Inversions which violate the order are not allowed. A computational complexity analysis of the new preprocessing is given and its completeness is proven. Furthermore, in a number of practical experiments with randomly generated graphs the improvement of the new approach is shown. In experiments performed on random sample graphs, and on real-world datasets. The number of permutations for the real-world datasets have been decreased to a fraction of 10^{-5} and 10^{-8} in average compared to the original approach by Messmer. The novel indexing strategy makes indexing of larger graphs feasible, allowing for fast detection of subgraphs.

Graphs play a major role in structural pattern recognition. An important task in this field is to find similar structures (error-tolerant graph matching) or the same structure (exact graph matching). The focus of this paper is on the latter task, which is important if exactly the same sub-structure needs to be retrieved.

Exact graph matching is needed when the user searches for specific constellations in molecules (Schomburg et al., 2004), in computer vision for the recognition of 3-D objects (Kim and Kak, 1991; Wong, 1992), shape matching in image analysis (Cheng and Huang, 1984; Bunke and Messmer, 1995), or room-constellations in floor plans (Weber et al., 2010). In most applications, the retrieval result should be available in real-time and the database of reference structures does not change too often. For those situations it is advisable to build an index of the reference structures in advance.

Such a method has been proposed by Messmer and Bunke (1999). It builds an index using the permutated adjacency matrix of the graph. The real-time search is then based on a tree based search. While the method has shown to be efficient for reference set with small graphs, it is infeasible for graphs with more than 19 vertices.

This paper proposes a method to overcome this problem. Assuming that the number of labels for the nodes is relatively small, we introduce a well-founded total order and apply this during index building. This optimization decreases the amount of possible permutations dramatically and allows building indexes of graphs with even more than 30 vertices.

Note that a preliminary version of this paper has been published in Weber et al. (2011). However, the focus of Weber et al. (2011) was on a short description of the approach and first experiments on random graphs. This paper elaborates more on the algorithm, its validity and its complexity. Furthermore, additional experiments have been performed new experiments on random graphs with 100–150 vertices. Moreover experiments on two real-world databases, i.e., the AIDS Antiviral Screen Database of Active Compounds (Development Therapeutics Program, 2004) and the Mutagency database (Kazius et al., 2005) have been accomplished.

The rest of this paper is organized as follows. First, Section 1 gives an overview over related work. Subsequently, Section 2 introduces definitions and notation which are used and Section 3 describes the new preprocessing step. Next, Section 4 shows that the number of computational steps is significantly decreased on random graphs as well as on realistic databases. Finally, Section 5 concludes the work.

1. Related work

In (Conte et al., 2004; Gao et al., 2009), a survey of the work done in the area of graph matching is given. Conte et al. (2004) defines two taxonomies, one which almost contains all the graph matching algorithms proposed from the late seventies, and describes the different classes of algorithms. The second considers the types of common applications of graph-based techniques in the Pattern Recognition and Machine Vision field. Using this

* Corresponding author.
E-mail address: Markus.Weber@dfki.de (Markus Weber).

0167-8655/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.patrec.2012.04.017

taxonomy, our approach can be assigns to exact matching, as it is a modified version of Messmer's method (Messmer and Bunke, 1999) which is assign to this category.

The focus of Gao et al. (2009) is the calculation of error-tolerant graph-matching; where calculating a graph edit distance (GED) is an important way. Mainly the GED algorithms described are categories into algorithms working on attributed or non-attributed graphs. Ullman's method (Ullmann, 1976) for subgraph matching is known as one of the fastest methods. The algorithm attains efficiency by inferentially eliminating successor nodes in the tree search. To filter unmatched graphs, enumerated paths are used as index features in GraphGrep (Giugno and Shasha, 2002). While TreePi (Zhang et al., 2007) and FG-Index (Cheng et al., 2007) use frequent subtrees/subgraphs as index features, GIndex (Yan et al., 2004) uses discriminative frequent fragments to improve filtering and reduce index size. GString (Jiang et al., 2007) reduces the problem of graph querying to subsequence matching. A graph decomposition based approach is taken in (Williams et al., 2007) to hash canonical subgraphs for fast accessing. A similar approach is taken in SAGA (Tian et al., 2007) in which answers are generated by assembling hits of enumerated fragments. In (Jamil, 2011), a new data model for the storage and management of graph objects has been proposed. It relies on the idea of structural unification, a novel graph representation based on minimum structures, and an indexing mechanism for storing minimum graph structures.

Bunke (2000), Bunke (2000) discussed several approaches in graph-matching. One way to cope with error-tolerant subgraph matching is using the maximum common subgraph as a similarity measure. Furthermore the application of graph edit costs which is an extension of the well-known string edit distances. A further group of suboptimal methods are approximative methods, they are based on neural networks (Jain and Wysotzki, 2005), such as the Hopfield network, Kohonen map (Xu and Oja, 1990) or Potts MFT neural net. Moreover methods as genetic algorithms (Cross et al., 1996; Wang et al., 1997), Eigenvalues (Umeyama, 1988), and linear programming (Almohamad and Duffuaa, 1993) are used.

Recently, He and Singh proposed GraphQL as a query language for graphs He and Singh, 2008. GraphQL assumes an underlying optimization based on prudent access structures and cost model. Graph matching is challenging in presence of large databases (Weber et al., 2010; Bunke, 2000; Bunke, 1997; Riesen and Bunke, 2008. Consequently, methods for preprocessing or indexing are essential. Preprocessing can be performed by graph filtering or concept clustering. The main idea of the graph filtering is to use simple features to reduce the number of feasible candidates. Another concept, clustering, is used for grouping similar graphs. In principle, given a similarity (or dissimilarity) measure, such as GED Bunke and Shearer, 1998, any clustering algorithm can be applied. Graph indexing can be performed by the use of decision trees.

Messmer and Bunke (1999) proposed a decision tree approach for indexing the graphs. They are using the permutated adjacency matrix of a graph to build a decision tree. This technique is quite efficient during run time, as a decision tree is generated beforehand which contains all model graphs. However, the method has to determine all permutations of the adjacency matrices of the search graphs. Thus, as discussed in their experiments, the method is practically limited to graphs with a maximum of 19 vertices. The main contribution of this paper is to improve the method of Messmer and Bunke for special graphs by modifying the index building process.

2. Definitions and notations

In this section some basic definitions are given which will be used throughout the paper.

Definition 1. A labeled graph is a 6-tuple, $G = (V, E, L_v, L_e, \mu, \upsilon)$, where

- V is a set of vertices.
- $E \subseteq V \times V$ is a set of edges.
- L_v is a set of labels for the vertices.
- L_e is a set of labels for the edges.
- $\mu: V \to L_v$ is a function which assigns a label to the vertices.
- $\upsilon: E \to L_e$ is a function which assigns a label to the edges.

The labels L_v set is a finite set and the labeling function μ assigns the type of an entity to a concrete vertex.

A common representation for a labeled graph is an adjacency matrix.

Definition 2. An adjacency matrix is $n \times n$ matrix M.

$$M = (m_{ij}), \quad i,j = 1,\ldots,n, \quad \text{where}$$
$$m_{ii} = \mu(v_i)$$

and

$$m_{ij} = \upsilon((v_i, v_j)) \quad \text{for } i \neq j.$$

Fig. 1 shows an example illustration of a graph and a possible corresponding adjacency matrix. Furthermore, the so called row-column representation is given. In a row-column representation the matrix is represented by its row-column elements a_i, where a_i is a vector of the form

$$a_i = (m_{1i}, m_{2i}, \ldots, m_{ii}, m_{i(i-1)}, \ldots, m_{i1}).$$

The following definition for the subgraph is given by:

Definition 3. Given a graph $G = (V, E, L_v, L_e, \mu, \upsilon)$, a subgraph of G is a graph $G' = (V', E', \mu', \upsilon', L'_v, L'_e)$ such that

1. $V' \subseteq$ V.
2. $E' = E \cap (V' \times V')$.
3. $\mu'(v) = \begin{cases} \mu(v) & \text{if } v \in V' \\ \text{undefined} & \text{otherwise} \end{cases}$
4. $\upsilon'(e) = \begin{cases} \upsilon(e) & \text{if } e \in E' \\ \text{undefined} & \text{otherwise} \end{cases}$

Let $G = (V, E, L_v, L_e, \mu, \upsilon)$ be a graph with $V = \{v_1, v_2, \ldots, v_n\}$. As stated above, G can also be represented by an adjacency matrix M. Note that the matrix M is not unique for a graph G. If M represents G, then any permutation of M is also a valid representation of G.

Definition 4. A $n \times n$ matrix $P = (p_{ij})$ is a permutation matrix if

1. $p_{ij} \in \{0,1\}$ for $i,j = 1,\ldots,n$.
2. $\sum_{i=1}^{n} p_{ij} = 1$ for $j = 1,\ldots,n$.
3. $\sum_{j=1}^{n} p_{ij} = 1$ for $i = 1,\ldots,n$.

Let G be a graph represented by an $n \times n$ adjacency matrix M and P be an $n \times n$ permutation matrix P with P^T as the transposed matrix, then the $n \times n$ matrix

$$M' = PMP^T$$

is also a valid representation of G.

Definition 5. Let $G = (V, E, \mu, \upsilon, L_v, L_e)$ be a graph, then $A(G)$ is the set of all permuted adjacency matrices of G,

$$A(G) = \{M_P | M_P = PMP^T \text{ where } P \text{ is a } n \times n \text{ permutation matrix}\}.$$