Pattern Recognition 48 (2015) 331–343

Contents lists available at ScienceDirect

Pattern Recognition

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

Approximation of graph edit distance based on Hausdorff matching

Andreas Fischer [a,*], Ching Y. Suen [a], Volkmar Frinken [b], Kaspar Riesen [c], Horst Bunke [d]

[a] Concordia University, Centre for Pattern Recognition and Machine Intelligence, 1455 de Maisonneuve Blvd West, Montreal, Canada, H3G 1M8
[b] Kyushu University, Faculty of Information Science and Electrical Engineering, 744 Motooka, Nishi-ku, Fukuoka 819-0395, Japan
[c] University of Applied Sciences and Arts Northwestern Switzerland, Riggenbachstrasse 16, 4600 Olten, Switzerland
[d] University of Bern, Institute of Computer Science and Applied Mathematics, Neubrückstrasse 10, 3012 Bern, Switzerland

ARTICLE INFO

Article history:
Received 27 September 2013
Received in revised form
8 May 2014
Accepted 21 July 2014
Available online 1 August 2014

Keywords:
Graph edit distance
Hausdorff distance
Approximation algorithms
Graph classification
Graph embedding
Handwriting recognition

ABSTRACT

Graph edit distance is a powerful and flexible method for error-tolerant graph matching. Yet it can only be calculated for small graphs in practice due to its exponential time complexity when considering unconstrained graphs. In this paper we propose a quadratic time approximation of graph edit distance based on Hausdorff matching. In a series of experiments we analyze the performance of the proposed Hausdorff edit distance in the context of graph classification and compare it with a cubic time algorithm based on the assignment problem. Investigated applications include nearest neighbor classification of graphs representing letter drawings, fingerprints, and molecular compounds as well as hidden Markov model classification of vector space embedded graphs representing handwriting. In many cases, a substantial speedup is achieved with only a minor loss in accuracy or, in one case, even with a gain in accuracy. Overall, the proposed Hausdorff edit distance shows a promising potential in terms of flexibility, efficiency, and accuracy.

1. Introduction

Due to their ability to represent properties of objects and binary relationships at the same time, graphs have found widespread applications in pattern recognition [1–3]. Graph-based representations have been successfully used in various fields of research including bioinformatics [4], web content mining [5], image classification [6], graphical symbol recognition [7], character recognition [8], and computer network analysis [9] to name just a few.

Yet, after the initial enthusiasm induced by the representational power and flexibility of graphs in the late seventies, a number of problems became evident. First, working with graphs is unequally more challenging than working with feature vectors, as even basic mathematical operations cannot be defined in a standard way, but must be provided depending on the specific application. Secondly, graphs suffer from their own flexibility. For instance, computing the distance of a pair of objects, which is an important task in many areas, is linear in the number of data items in the case where vectors are employed. The same task for graphs, however, is much more complex, since one cannot simply compare the sets of nodes and edges, which are generally unordered and of different sizes.

In the last decades various procedures for evaluating proximity, that is similarity or dissimilarity, of graphs have been proposed in the literature [1]. The process of evaluating the similarity of two graphs is commonly referred to as *graph matching*. Roughly speaking, there are two categories of tasks in graph matching, viz. *exact graph matching* and *error-tolerant graph matching*. In the former case, for a matching to be successful, it is required that a strict correspondence is found between the two graphs being matched, or at least among their subparts. Prominent examples include methods for graph and subgraph isomorphism [10,11].

Due to the intrinsic variability of the objects under consideration and the noise resulting from the graph extraction process, it cannot be expected that two graphs representing the same class of objects are completely, or at least to a large part, identical in their structure. Especially if the nodes and edges of a graph are attributed with real-valued labels, it is most probable that the actual graphs differ somewhat from their ideal model. Obviously, such noise crucially hampers the applicability of exact matching techniques, and consequently exact graph matching is rarely used in real-world pattern recognition applications. In order to overcome this drawback, various approaches to error-tolerant graph matching have been proposed [1].

Graph edit distance (GED) offers an intuitive way to integrate tolerance to errors into the graph matching process and is applicable

* Corresponding author. Tel.: +1 514 848 2424x7950; fax: +1 514 848 2830.
E-mail address: andreas.fischer@polymtl.ca (A. Fischer).

http://dx.doi.org/10.1016/j.patcog.2014.07.015
0031-3203/

to virtually all types of graphs. Originally, edit distance has been developed for string matching [12]. A generalization to graphs first emerged in [13] in the context of error-correcting isomorphism and has been extended to a general graph distance subsequently [14,15]. The key idea is to model structural variation by edit operations reflecting modifications in structure and labeling. A standard set of edit operations is given by *deletions*, *insertions*, and *substitutions* of both nodes and edges.[1] Given two graphs, the idea of graph edit distance is to delete some nodes and edges in the first graph, substitute (relabel) some of the remaining nodes and edges, and insert some new nodes and edges such that the first graph is transformed into the second graph. An edit cost is assigned to each edit operation and the graph edit distance corresponds with the minimum cost among all valid graph transformations. It is possible to integrate domain specific knowledge about object similarity, if available, when defining the costs of the elementary edit operations. Furthermore, if in a particular case prior domain knowledge is not available, automatic procedures for learning the edit costs from a set of sample graphs exist [17–19].

The flexibility of graph edit distance to cope with any kind of graph structure, node labels, and edge labels is an advantage over other graph matching methods that often impose constraints on the graphs. For example spectral methods [20,21], which are based on the eigendecomposition of the adjacency or Laplacian matrix of a graph, primarily target unlabeled graphs or allow only severely constrained label alphabets. Other common constraints include restrictions to ordered graphs [22], planar graphs [23,24], bounded-valence graphs [25], trees [26], and graphs with unique node labels [9]. The absence of such constraints makes graph edit distance applicable to a wide range of real-world applications.

However, the flexibility comes at the cost of high computational complexity. Optimal algorithms for computing the graph edit distance are typically based on combinatorial search procedures that explore the space of all valid graph transformations. Often A* search techniques using some heuristics are employed [27–30]. A* is a best-first search algorithm [31] which is *complete* and *admissible*, that is it always finds a solution if there is one and it never overestimates the cost of reaching the goal. The time complexity is exponential with respect to the number of nodes of the involved graphs, thus constraining graph edit distance to small graphs in practice.

In recent years, a number of methods addressing the high computational complexity of graph edit distance computation have been proposed. Probabilistic relaxation labeling [32,33] adopts a Bayesian perspective on graph edit distance and iteratively applies edit operations to improve a maximum *a posteriori* criterion. As an alternative to this hill climbing approach, genetic algorithms have been proposed for optimization in [34]. In either case, the method is usually more efficient than A* search but is prone to finding only local optima in the search space. The same holds true for the methods proposed in [35,36] where a randomized construction of initial mappings is followed by a local search procedure. For A* search, efficiency improvements by means of suboptimal beam search techniques are proposed in [37]. In [38], a linear programming method for computing the edit distance of graphs with unlabeled edges is reported. Based on the assignment problem, a polynomial time calculation of upper and lower bounds is suggested.

Assignment edit distance (AED) is a general method to approximate graph edit distance for unconstrained graphs in cubic time with respect to the number of graph nodes. Originally, it has been introduced as a novel heuristic for optimal graph edit distance computation based on fast node assignments [30]. This heuristic has eventually been used in [39] as a stand alone graph edit distance approximation. The method is based on a fast optimization procedure mapping nodes and their local structure of one graph to nodes and their local structure of another graph. In order to find an optimal match between the sets of local structure the Hungarian algorithm [40],[2] is deployed. Since only local structures are matched the resulting assignment edit distance is not optimal. Yet a series of graph classification experiments empirically demonstrated a substantial speedup without significant loss in accuracy when using this approximation algorithm [39]. Further speedups were reported in [42] with a different cubic time assignment algorithm, viz. the algorithm of Jonker and Volgenant [43]. In recent years, the algorithmic framework of assignment edit distance has been successfully employed for several applications where graphs have been embedded in vector spaces [44]. In [45] the method has been adopted to the problem of exact graph matching, namely graph isomorphism and subgraph isomorphism. The graph matching framework has been made publicly available as a stand-alone software tool [46].[3]

In this paper, we continue this line of research by introducing the Hausdorff edit distance (HED), which approximates the graph edit distance more efficiently in quadratic rather than cubic time. It combines the idea of assignment edit distance, that is to find a match between nodes and their local structure, with a more efficient pairwise node matching. In a straight-forward fashion, each node of one graph is compared with each node of the other graph similar to comparing subsets of a metric space using the Hausdorff distance [47]. Taking into account costs for deletion, substitution, and insertion of a node and its adjacent edges, an optimal match is found for each node individually. The sum of all matching costs is then considered as a distance measure which is less than or equal to the graph edit distance.

Despite the fact that an optimal match is found for each node individually and not for all nodes conjointly, the proposed Hausdorff edit distance performs astonishingly well in a series of graph classification experiments. They empirically demonstrate a substantial speedup when compared with graph edit distance and assignment edit distance. In many cases, the speedup is achieved with only a minor loss in accuracy or, in one case, even with a gain in accuracy. We report results for nearest neighbor classification of graphs representing letter drawings, fingerprints, and molecular compounds. Furthermore, we have investigated the Hausdorff edit distance in the context of a more complex document analysis system for automatic handwriting recognition in historical manuscripts.

The remainder of this paper is organized as follows. First, a description of graph edit distance and its cubic time approximation is provided in Section 2. Afterwards, the proposed Hausdorff edit distance is detailed in Section 3. Graph classification experiments are first presented for k-nearest neighbor recognition in Section 4, followed by a report on handwriting recognition using graph embedding and hidden Markov models in Section 5. Finally, we draw some conclusions in Section 6.

Note that this paper is an extended version of an earlier conference publication [48]. While the previous publication was focused on the handwriting recognition application, this paper provides a generalization of the algorithm that is able to cope with virtually all types of graphs. The detailed description of the proposed Hausdorff edit distance in Section 3 is completely new

[1] Note that other operations, such as *merging* and *splitting* of nodes have been proposed [16].

[2] The Hungarian algorithm is a refinement of an earlier version by Kuhn [41] and is also referred to as Munkres or Kuhn–Munkres algorithm.

[3] http://www.fhnw.ch/wirtschaft/iwi/gmt.