Image and Vision Computing 40 (2015) 38–48

Contents lists available at ScienceDirect

Image and Vision Computing

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

Computation of graph edit distance: Reasoning about optimality and speed-up☆,☆☆

Francesc Serratosa

Universitat Rovira i Virgili, Tarragona, Spain

ARTICLE INFO

Article history:
Received 2 December 2014
Received in revised form 12 June 2015
Accepted 13 June 2015
Available online 25 June 2015

Keywords:
Error-tolerant graph matching
Bipartite graph matching algorithm
Fast Bipartite
Square Fast Bipartite
Hungarian method
Jonker–Volgenant solver

ABSTRACT

Bipartite graph matching has been demonstrated to be one of the most efficient algorithms to solve error-tolerant graph matching. This algorithm is based on defining a cost matrix between the whole nodes of both graphs and solving the nodes' correspondence through a linear assignment method (for instance, Hungarian or Jonker–Volgenant methods). Recently, two versions of this algorithm have been published called Fast Bipartite and Square Fast Bipartite. They compute the same distance value than Bipartite but with a reduced runtime if some restrictions on the edit costs are considered. In this paper, we do not present a new algorithm but we compare the three versions of Bipartite algorithm and show how the violation of the theoretically imposed restrictions in Fast Bipartite and Square Fast Bipartite do not affect the algorithm's performance. That is, in practice, we show that these restrictions do not affect the optimality of the algorithm and so, the three algorithms obtain similar distances and recognition ratios in classification applications although the restrictions do not hold. Moreover, we conclude that the Square Fast Bipartite with the Jonker–Volgenant solver is the fastest algorithm.

1. Introduction

Attributed graphs have been used in some pattern recognition fields such as object recognition [1–3] scene view alignment [4–6] multiple object alignment [7,8], object characterization [9,10] interactive methods [11,12] image registration [13], tracking [14] among a great amount of other applications. Interesting reviews of techniques and applications are [15,16] and [17]. Error-tolerant graph-matching algorithms compute the correspondences between nodes of two Attributed Graphs that minimises some kind of objective function. One of the most widely used methods to evaluate an Error-correcting graph isomorphism is the Graph Edit Distance [18–21]. The basic idea behind the Graph Edit Distance is to define a dissimilarity measure between two graphs based on the minimum amount of required distortion to transform one graph into the other. To this end, a number of distortion or edit operations, consisting of insertion, deletion and substitution of both nodes and edges are defined. Then, for every pair of graphs ($\mathbf{G^p}$ and $\mathbf{G^q}$), there is a sequence of edit operations that transforms one graph into the other. To quantitatively evaluate which sequence is the best, edit cost functions are introduced. The basic idea is to assign a penalty cost to each edit operation according to the amount of distortion introduced in the transformation. Unfortunately, the time and space complexity to compute the minimum of these objective functions is very high. For this reason, almost 20 years ago appeared the Graduated Assignment algorithm [22] that computes a sub-optimal solution of the Error-Tolerant Graph Matching problem in $\mathbf{O}(\boldsymbol{n}^6)$, being $\boldsymbol{n}$ the number of nodes of both graphs.

Other methods different from graph edit distance have been presented in which the flexibility to cope with any kind of domains in node and edges and different structures is reduced but also their computational cost. One example are the spectral methods [23,24], which are based on the eigendecomposition of the adjacency or Laplacian matrix of a graph. In this framework, graphs are unlabelled or only allow severely constrained label alphabets. Other common constraints include restrictions to ordered graphs [25], planar graphs [26,27], bounded-valence graphs [28], trees [29] and graphs with unique node labels [30]. Finally, a general optimisation framework based on a graduated non-convexity and concavity procedure (GNCCP) [31] has been applied to solve the error-tolerant graph matching. They present a comparison to well-know methods like [22] showing good achievements.

The Graph Edit Distance is applicable to a wide range of real-world applications since any type of attributes can be used. In recent years, a number of methods addressing the high computational complexity of graph edit distance computation have been proposed. Probabilistic relaxation labelling [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 [35] a randomized construction of initial mappings is followed by a local search procedure. In [36], a linear programming method for computing the edit distance of graphs with unlabelled edges is reported. And also, dominant sets have been applied to sub-optimally compute the edit

☆ This research is supported by the Spanish projects DPI2013-42458-P and TIN2013-47245-C2-2-R.
☆☆ This paper has been recommended for acceptance by Cheng-Lin Liu.
E-mail address: francesc.serratosa@urv.cat.

http://dx.doi.org/10.1016/j.imavis.2015.06.005
0262-8856/© 2015 Elsevier B.V. All rights reserved.

Table 1
Basic features of BP, FBP and SFBP algorithms.

1st ref.	Algorithm	Linear solver	Computational Cost	Restrictions	Symmetric runtime
[39]	BP	Hungarian	$O((n+m)^3)$	No	No
[40]		Jonker–Volgenant			
[41]	FBP	Hungarian		Insertion and deletion costs	
[42]		Jonker–Volgenant	$O(max(n,m)^3)$		No convergence
	SFBP	Hungarian			Yes
		Jonker–Volgenant			

distance [37]. Finally, in [38] the graph edit distance is approximated by the Hausdorff distance.

Recently, a new algorithm called Bipartite (BP) [39] and two other versions of this algorithm have appeared that are called Fast Bipartite [41] (FBP) and Square Fast Bipartite [42] (SFBP). Note, this algorithm solves the error-tolerant graph-matching problem between attributed and undirected graphs. Therefore, it has not any relation between matching graphs that are classified as bipartite. These three algorithms have a cubic computational cost with respect to the number of nodes and are implemented in two main steps. In the first one, a cost matrix is defined and in a second one, a linear solver on this matrix is applied to find the final distance and node correspondence. Due to BP, FBP or SFBP do not consider the structural information globally but only locally, the obtained distance tends to be larger than the exact one. Some methods [43,44] improve this distance and obtain a new correspondence starting from the correspondence computed by BP, FBP or SFBP at the expense of increasing the runtime.

Table 1 summarises the main properties and differences of these algorithms extracted from [41] and [42]. Two linear solvers have been used; the Hungarian method [45,46] and the Jonker–Volgenant solver [48], which was published later. It is not guaranteed that both methods obtain exactly the same correspondence, although there is a clear tendency of obtaining similar assignations. Therefore, the distance obtained by the three algorithms through the Hungarian method can be different of the distance obtained by the same algorithms but through the Jonker–Volgenant solver. The Hungarian method is usually slower than the Jonker–Volgenant solver but always converges. The Jonker–Volgenant is faster but it has convergence problems in some cost matrices. In this way, the Hungarian method always converges while applied to the cost matrices defined by BP and FBP algorithms but the Jonker–Volgenant method not always finishes on the cost matrix defined by FBP. The Computational Cost of FBP and SFBP is slightly lower than BP but in the expense of introducing some restrictions on the edit costs [41,42]. In Table 1, ***n*** and ***m*** represent the order of both graphs. Finally, it was shown in [41] that the real runtime of these solvers clearly depends on the cost matrix and so, the runtime of comparing two graphs depends on the order of presentation of these graphs. We call this property a non-symmetric runtime. In this way, it is worth to consider the number of nodes of the graphs to decide in which order the graphs are introduced into the matching algorithm.

In some classification applications or tests on databases, it would be useful to set some edit costs such that the edit costs restrictions theoretically imposed to FBP and SFBP do not hold with the aim of increasing the recognition ratio. The aim of this paper is to present a real comparison of the three methods and to show to which extend the fact of not holding the edit cost restrictions affects on the runtime, optimality and recognition ratio. That is, we want to show the applicability of these algorithms on real graph problems not only from the runtime point of view but also from the recognition ratio point of view. To do so, we performed two types of experiments. The first ones are applied on synthetic graphs and we want to discover which is the increase of the obtained graph edit distance when we move away from the edit cost restrictions. The second ones are applied to public graph databases and we want to show the relation between edit costs (although they may violate the restrictions), recognition ratio and run time.

The outline of the paper is as follows: in the next section, we define the Attributed graphs and the Graph Edit Distance. In Section 3, we comment the two most well known linear assignment solvers. In Section 4, we summarise algorithms BP, FBP and SFBP and also the Hungarian and Jonker–Volgenant linear solvers. In Section 5, we move on the experimental part to present a comparison of the applicability of these three algorithms. Section 6 concludes the paper.

2. Attributed graphs and Graph Edit Distance

In this section, we first define Attributed graphs and Error-tolerant graph matching and then we explain the Graph Edit Distance.

2.1. Attributed graphs

An Attributed Graph is defined as a triplet $G = (\Sigma_v, \Sigma_e, \gamma_v, \gamma_e)$, where $\Sigma_v = \{v_a \mid a = 1, \ldots, n\}$ is the set of vertices and $\Sigma_e = \{e_{ab} | a, b \in 1, \ldots, n\}$ is the set of edges. Functions $\gamma_v : \Sigma_v \to \Delta_v$ and $\gamma_e : \Sigma_e \to \Delta_e$ assign attribute values in any domain to vertices and edges. The order of graph G is n. We call $E(v_a)$ to the number of neighbours of node v_a, that is, the number of outgoing edges. Finally, we define the neighbours of a node v_a, named N_a, on an attributed graph G, as another graph $N_a = (\Sigma_v^{N_a}, \Sigma_e^{N_a}, \gamma_v^{N_a}, \gamma_e^{N_a})$. The definition of the neighbours of a node is needed to define two different local sub-structures in Section 4. N_a has the structure of an attributed graph but it is only composed of nodes connected to v_a by an edge. Formally, $\Sigma_v^{N_a} = \{v_b | e_{ab} \in \Sigma_e\}$, $\Sigma_e^{N_a} = \varnothing$ (empty set) and $\gamma_v^{N_a}(v_b) = \gamma_v(v_b), \forall v_b \in \Sigma_v^{N_a}$.

2.2. Error-correcting graph isomorphism

Let $G^p = (\Sigma_v^p, \Sigma_e^p, \gamma_v^p, \gamma_e^p)$ and $G^q = (\Sigma_v^q, \Sigma_e^q, \gamma_v^q, \gamma_e^q)$ be two Attributed graphs of initial order n and m. To allow maximum flexibility in the matching process, graphs are extended with null nodes to be of order $n+m$. We refer to null nodes of G^p and G^q by $\hat{\Sigma}_v^p \subseteq \Sigma_v^p$ and $\hat{\Sigma}_v^q \subseteq \Sigma_v^q$ respectively. We assume that null nodes have indices $a \in [n+1, \ldots, n+m\}$ and $i \in [m+1, \ldots, n+m\}$ for graphs G^p and G^q, respectively. Let T be a set of all possible bijections between two vertex sets Σ_v^p and Σ_v^q. We define the non-existent or null edges as $\hat{\Sigma}_e^p \subseteq \Sigma_e^p$ and $\hat{\Sigma}_e^q \subseteq \Sigma_e^q$. Isomorphism $f^{p,q} : \Sigma_v^p \to \Sigma_v^q$, assigns one vertex of G^p to only one vertex of G^q. The

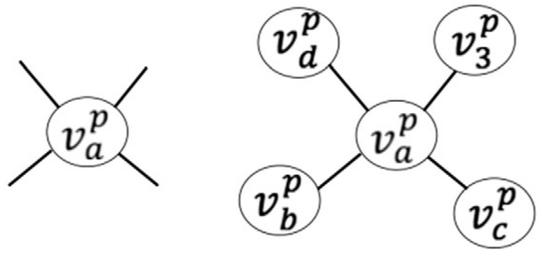

Fig. 1. Example of the local sub-structures: Degree centrality and Clique centrality.