Pattern Recognition 60 (2016) 971–982

Contents lists available at ScienceDirect

Pattern Recognition

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

A fast projected fixed-point algorithm for large graph matching ☆

Yao Lu *, Kaizhu Huang, Cheng-Lin Liu

National Laboratory of Pattern Recognition (NLPR), Institute of Automation, Chinese Academy of Sciences, No. 95 Zhongguancun East Road, Beijing 100190, China

ARTICLE INFO

Article history:
Received 10 June 2014
Received in revised form
3 February 2016
Accepted 7 July 2016
Available online 9 July 2016

Keywords:
Graph matching
Projected fixed-point
Large graph algorithm
Feature correspondence
Point matching

ABSTRACT

We propose a fast algorithm for approximate matching of large graphs. Previous graph matching algorithms suffer from high computational complexity and therefore do not have good scalability. By using a new doubly stochastic projection, for matching two weighted graphs of n nodes, our algorithm has time complexity only $O(n^3)$ per iteration and space complexity $O(n^2)$. We proved that our algorithm converges at a super-logarithmic rate. Experiments on large synthetic and real graphs (over 1000 nodes) were conducted to evaluate the performance of various algorithms. Results show that due to its fast convergence, our algorithm is more than an order of magnitude faster than the previous state-of-the-art algorithms, while maintaining comparable accuracy in large graph matching.

1. Introduction

Graph matching, aiming to find the optimal correspondences between the nodes of two graphs, is an important and active topic of research in computer vision and pattern recognition [1,2]. It has been extensively applied in various fields including optical character recognition [3,4], object recognition [5,6], shape matching [6–8], face recognition [9], feature correspondence [10], point matching [11], image retrieval [12], video indexing [13], document processing [14], protein classification [15] and fingerprint identification [16].

Graph matching is in general a NP-hard discrete optimization problem. Exact graph matching algorithms include Ullman's method [17], Nauty [18] and VF2 [19], all of which have exponential time complexity in the worst cases. To match two graphs within a reasonable time, one has to look for approximate solutions. Moreover, due to noise and variability in real world graphs, the usage of exact graph matching algorithms is very limited because the exact algorithms are not robust to noise or variation. The focus of this paper is to design an approximate algorithm for efficiently and robustly matching general large graphs (e.g. graphs of over 1000 nodes) for computer vision and pattern recognition purposes.

One approach of approximate graph matching algorithms is based on tree search [20,21]. Its basic idea is tree search with backtracking while using heuristics to prune unfruitful paths. Another approach of approximate graph matching algorithms is based on continuous relaxation of the discrete problem while using continuous optimization techniques or heuristics to optimize a matching objective. Classic work include Relaxation Labeling [22–24], Umeyama's method [25], Graduated Assignment [26] and Replicator Dynamics [27]. Generally, continuous relaxation based algorithms have lower computational costs than tree search based ones [1].

Recent work on graph matching is mainly focused on the continuous relaxation approach. Most of them suffer from high computational costs and can only match small graphs. In the original papers of recent graph matching algorithms [10,28–36], the experiments were done on graphs of only 20–200 nodes. However, real world images typically contain hundreds to thousands of local features (e.g. SIFT [37]). Hence, there is a huge gap between the potential power of graph matching and its practical use.

In this paper, within the continuous relaxation framework, we propose a novel fast graph matching algorithm called Doubly Stochastic Projected Fixed-Point (DSPFP), which is capable of dealing with large graphs of over 1000 nodes in a PC. By using a recently developed doubly stochastic projection [38], our algorithm has time complexity $O(n^3)$ per iteration and space complexity $O(n^2)$. In addition to its scalability, our algorithm is easy to implement, robust, and able to match undirected weighted

☆We address the scalability problem of graph matching algorithms. Previous algorithms suffer from high time complexity and cannot deal with large graphs (e.g. over 1000 nodes) efficiently. We propose a fast algorithm with a convergence guarantee. Our invention is to make graph matching algorithms practical.

* Corresponding author.
E-mail addresses: yaolubrain@gmail.com (Y. Lu), kaser.huang@gmail.com (K. Huang), liucl@nlpr.ia.ac.cn (C.-L. Liu).

http://dx.doi.org/10.1016/j.patcog.2016.07.015
0031-3203/

attributed graphs of different sizes. We also proved the super-logarithmic convergence rate of the new projected fixed-point algorithm, based on the theory of convex projection [39,40]. To the best of our knowledge, our algorithm is the only iterative graph matching algorithm with a super-logarithmic convergence guarantee. We conduct extensive experiments on benchmark datasets of large graphs. Due to its fast convergence, DSPFP demonstrated an order of magnitude of speed-ups compared to previous state-of-the-art algorithms while maintaining comparable accuracy.

The rest of this paper is organized as follows. In Section 2, we review previous work on continuous relaxation based graph matching algorithms. A simplified analysis of performance of different algorithms is given. In Section 3, we present our formulation of the graph matching problem. In Section 4, we introduce our algorithm, DSPFP, including the derivation, the new and many other projection methods, and convergence analysis. In Section 5, we show extensive experiments conducted on various benchmark datasets in comparison with previous state-of-the-art algorithms in large graph matching. In Section 6, we study the parameter sensitivity and limitations of our algorithm. Finally, we give concluding remarks in Section 7.

2. Previous work

In this section, we review and analyze recent state-of-the-art graph matching algorithms. For a simplified analysis, let us consider the performance of different algorithms in matching two weighted graphs of n nodes.

The Linear Programming approach [41] has time complexity $O(n^6)$ per iteration. PATH [33] calls Hungarian method several times for each iteration, making it slow in practice. Many state-of-the-art graph matching algorithms including Graduated Assignment (GA) [26], Spectral Matching (SM) [10], Spectral Matching with Affine Constraint (SMAC) [29], Integer Projected Fixed Point (IPFP) [30], Reweighted Random Walks Matching (RRWM) [31], and Max-Pooling Matching (MPM) [36] all aim to solve an Integer Quadratic Programming problem, which involves a $n^2 \times n^2$ compatibility matrix. The construction and the computation on the compatibility matrix takes $\Omega(n^4)$ operations. Note that although it is claimed that the compatibility matrix can be very sparse so that sparse matrix techniques can be used for efficient storage and computation, this is not the case for either weighted graphs or densely connected unweighted graphs. Moreover, sparse matrix techniques have overheads despite of their lower time complexity. Therefore, all the above algorithms with an $n^2 \times n^2$ compatibility matrix do not have good scalability with respect to the size of graphs. Path-following [33] and Factorized Graph Matching (FGM) [34] rely on the expensive Frank–Wolfe algorithm [42] or its variants for each iteration. Despite of $O(n^3)$ per iteration time complexity, these algorithms do not scale well in practice. In their experiments [33,34], the size of graphs is up to 100 nodes. The Dual Decomposition (DD) [35] algorithm is even more computational expensive. In the paper [35], it was reported that the average runtime of DD was 59.3 s for a set of graphs (CMU house) of only 30 nodes.

Despite of the accuracy of the above state-of-the-art algorithms, their high computational costs prevent them from large graph matching.

3. Problem formulation

For simplicity, we first consider graphs of equal sizes. Matching graphs of different sizes will be studied in Section 4.5. For two undirected graphs of size n, denote their adjacency matrices (symmetric) by $\mathbf{A}$ and $\mathbf{A}'$ (binary valued for unweighted graphs and real valued for weighted graphs) and attribute matrices by $\mathbf{B}$ and $\mathbf{B}'$, respectively. Each row of $\mathbf{B}$ and $\mathbf{B}'$ is a k-dimensional vector representing the attributes of a node. The size of $\mathbf{A}$ and $\mathbf{A}'$ is $n \times n$ and the size of $\mathbf{B}$ and $\mathbf{B}'$ is $n \times k$. In [26,43,10,29–31,36], the graph matching problem is formulated as the following Integer Quadratic Programming (IQP)

$$\max_{\mathbf{x}} \quad \frac{1}{2}\mathbf{x}^T\mathbf{W}\mathbf{x} + \lambda\mathbf{k}^T\mathbf{x} \tag{1}$$

$$\text{s. t.} \quad \mathbf{C}\mathbf{x} = \mathbf{b}, \quad \mathbf{x} \in \{0,1\}^{n^2}, \tag{2}$$

where $\mathbf{W}$ is an $n^2 \times n^2$ compatibility matrix, λ is a control parameter, $\mathbf{k}$ is vectorized $\mathbf{B}\mathbf{B}'^T$, and $\mathbf{C}$ and $\mathbf{b}$ are constraint constants enforcing $\mathbf{x}$ is a vectorized permutation matrix. The constant $\frac{1}{2}$ is for convenience, to be seen later. Under this formulation, algorithms inevitably have time complexity $\Omega(n^4)$ since the construction of and the computation on the $n^2 \times n^2$ compatibility matrix require $\Omega(n^4)$ operations. The space complexity is also $\Omega(n^4)$. The high complexity overkills their applications in large graph matching. In the experiments of [26,43,29–31,36], graphs are of size up to 100 nodes.

To reduce the high complexity of this formulation, we adopt a particular compatibility matrix $\mathbf{W} = \mathbf{A} \otimes \mathbf{A}'$, where $\otimes$is the Kronecker product. Due to the property of Kronecker product that for arbitrary matrices $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$

$$(\mathbf{Z}^T \otimes \mathbf{X})\mathrm{vec}(\mathbf{Y}) = \mathrm{vec}(\mathbf{X}\mathbf{Y}\mathbf{Z}) \tag{3}$$

where vec is the vectorization of a matrix, the IQP (1) and (2) is equivalent to

$$\max_{\mathbf{X}} \quad \frac{1}{2}\,\mathrm{tr}(\mathbf{X}^T\mathbf{A}\mathbf{X}\mathbf{A}') + \lambda\,\mathrm{tr}(\mathbf{X}^T\mathbf{K}), \tag{4}$$

$$\text{s.t.} \quad \mathbf{X}\mathbf{1} = \mathbf{1}, \quad \mathbf{X}^T\mathbf{1} = \mathbf{1}, \quad \mathbf{X} \in \{0,1\}^{n\times n}, \tag{5}$$

where $\mathrm{tr}(\cdot)$ denotes the matrix trace, $\mathbf{K}$ denotes the $n \times n$ matrix $\mathbf{B}\mathbf{B}'^T$ (see Appendix for derivation), and $\mathbf{1}$ is a vector with all its elements equal to one. As a result, we only have to deal with a few $n \times n$ matrices instead of a $n^2 \times n^2$ one. This formulation was already proposed in [44] and used in PATH [33]. See [44] for more discussion about different formulations of Quadratic Assignment Problem.

Under this formulation, the time complexity of GA and IPFP can be reduced to $O(n^3)$ per iteration directly. However, RRWM and MPM cannot be reduced in a similar way. RRWM requires an eliminating operation on the computed $\mathbf{W}$. Without this eliminating operation, the matching accuracy of RRWM would decrease. And MPM requires a Max-Pooling product of the $\mathbf{W}$ and $\mathbf{x}$, which cannot be reduced to formulation (4) (5).

The formulation (4) and (5) has a clear interpretation in terms of graph similarity. By a few matrix manipulations (see Appendix), (4) is equivalent to

$$\min_{\mathbf{X}} \frac{1}{2} \| \mathbf{A} - \mathbf{X}\mathbf{A}'\mathbf{X}^T \|_F^2 + \lambda \| \mathbf{B} - \mathbf{X}\mathbf{B}' \|_F^2, \tag{6}$$

where $\|.\|_F$ is the Frobenius matrix norm. In (6), the left term can be interpreted as dissimilarity between edges and the right term as dissimilarity between nodes. In all the above formulations, the problem is NP-hard [44].

4. Algorithm

Under the formulation of (4) and (5), we first introduce the