Pattern Recognition 48 (2015) 317–330

Contents lists available at ScienceDirect

Pattern Recognition

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

Efficient subgraph matching using topological node feature constraints

Nicholas Dahm [a,*], Horst Bunke [b], Terry Caelli [c], Yongsheng Gao [a]

[a] School of Engineering, Griffith University, 170 Kessels Rd, Nathan, Brisbane, QLD, Australia
[b] Institute for Computer Science and Applied Mathematics, University of Bern, Switzerland
[c] Electrical and Electronic Engineering, University of Melbourne, Australia

ARTICLE INFO

Article history:
Received 31 August 2013
Received in revised form
14 May 2014
Accepted 28 May 2014
Available online 12 June 2014

Keywords:
Graph matching
Subgraph isomorphism
Topological node features

ABSTRACT

This paper presents techniques designed to minimise the number of states which are explored during subgraph isomorphism detection. A set of advanced topological node features, calculated from *n*-neighbourhood graphs, is presented and shown to outperform existing features. Further, the pruning effectiveness of both the new and existing topological node features is significantly improved through the introduction of strengthening techniques. In addition to topological node features, these strengthening techniques can also be used to enhance application-specific node labels using a proposed novel extension to existing pruning algorithms. Through the combination of these techniques, the number of explored search states can be reduced to near-optimal levels.

1. Introduction

In structural pattern recognition, graphs are used to provide meaningful representations of objects and patterns, as well as more abstract descriptions. The representative power of graphs lies in their ability to characterise multiple pieces of information, as well as the relationships between them. These graphs can be used on a variety of applications including network analysis [1], face recognition [2], and image segmentation [3]. However at the heart of graph theory is the problem of graph matching, which attempts to find a way to map one graph onto another in such a way that both the topological structure and the node and edge labels are matched. For domains where data is noisy, an identical match may not be possible, so an inexact graph matching algorithm is used to search for the closest match, minimising some similarity function. In this paper, we deal only with exact graph matching, where a match must be perfect and error-free. Exact graph matching, due to its nature, is generally used on problems where the data is precise and does not contain noise. Examples of exact graph matching problems include finding chemical similarities [4] or substructures [5], protein–protein interaction networks [6], social network analysis [7], and in the semantic web [8].

Within exact graph matching, there are three graph isomorphism problems. These are, in increasing order of complexity, graph isomorphism, subgraph isomorphism, and maximum common subgraph isomorphism. While algorithms for all of these problems have an exponential time complexity in the general case, significant advances have been made towards making these problems tractable. In this paper we focus on techniques applicable to graph and subgraph isomorphism, with particular emphasis on non-induced subgraph isomorphism. Reviews of algorithms and approaches for maximum common subgraph isomorphism can be found in [9–11]. Those looking for a complete survey of graph matching techniques are directed to [9] for the years up to 2004, and [12] for those since.

One key concept in speeding up graph and subgraph isomorphism problems is that of a *topological node feature* (TNF). A TNF is a value assigned to a node which encodes information about the local graph topology into a simple value. For example, the simplest TNF, namely the node *degree*, simply counts the number of adjacent nodes. Topological node features are also known as *subgraph isomorphism consistents* or, in the case of graph isomorphism, *invariants*.

In graph isomorphism, the early Nauty algorithm [13] by McKay was able to make significant advances beyond existing algorithms by using TNFs and a strengthening procedure similar to the tree index method that is presented in this paper. Using these techniques, Nauty is able to effectively describe the graph topology surrounding each node, eliminating mappings to any nodes where the topology is not identical. This idea was extended by Sorlin and Solnon to create the IDL algorithm [14]. Some polynomial-time algorithms have been developed for special cases of graph isomorphism, such as planar graphs [15] and bounded valence graphs [16]. A recent paper by Fankhauser et al. [17] presents a polynomial-time algorithm for graph isomorphism in the general case. Their method is suboptimal, in that some graph pairs are rejected due to unresolved permutations. However the number of rejected pairs

* Corresponding author. Tel.: +61 7 3735 3753; fax.: +61 7 3735 5198.
E-mail addresses: n.dahm@griffith.edu.au (N. Dahm), bunke@iam.unibe.ch (H. Bunke), terry.caelli@gmail.com (T. Caelli), yongsheng.gao@griffith.edu.au (Y. Gao).

http://dx.doi.org/10.1016/j.patcog.2014.05.018
0031-3203/

was shown to be only 11 out of 1 620 000 graph pairs (0.00068%) from the MIVIA (previously SIVA) laboratory graph database [18].

Extending such concepts to subgraph isomorphism is challenging because, while the subgraph may exist within the full graph, the additional nodes and edges in the full graph create noise for TNFs. This essentially means that instead of searching for TNFs with identical values, algorithms must ensure that TNF values from subgraph nodes are less than or equal to the corresponding TNF values from the full graph. One of the earliest and most influential subgraph isomorphism algorithms was Ullmann's algorithm [19]. Based on a tree search with backtracking, and a refinement procedure to prune the search tree, Ullmann's algorithm maintained state of the art performance until being superseded by the VF2 algorithm. The VF2 algorithm by Cordella et al. [20] has established itself as the benchmark for subgraph isomorphism algorithms due to its impressive speed, outperforming Ullmann's in almost all cases. To achieve such impressive results, VF2 takes subsets of the degree TNF, creating either two values (for undirected graphs), or six values (for directed graphs). The effect of separating the degree value into multiple smaller values is that it reduces the chance that TNF noise will prevent an invalid mapping from being detected. Despite the impressive reduction of the computation complexity provided by VF2, the exponential nature of the subgraph isomorphism problem prevents such algorithms from being practical as the number of nodes increases [21]. An updated version of Ullmann's algorithm was recently published [22], which includes a technique called *focus search*. Focus search avoids some unnecessary backup and restore operations on the bit-vector domains (representing compatible node mappings) during the search.

A number of recent papers have shown that subgraph isomorphism can be efficiently solved by utilising *constraint programming* (CP). An early example of this is the nRF+ algorithm by Larrosa and Valiente [23], which is an extension of the non-binary *really full look ahead* (nRF) algorithm. The ILF algorithm by Zampelli et al. [24] explored the use of CP with multiple TNFs. The results show that ILF outperforms VF2 on most cases, even when restricted to only the TNF of degree. Another recent CP paper is the *local all different* (LAD) algorithm by Solnon [25]. The LAD constraint ensures that for every compatible node mapping, the nodes adjacent to the subgraph node can be uniquely mapped to nodes adjacent to the full graph node. Each mapping can be validated in this way by running the Hopcroft–Karp algorithm [26]. Given x nodes adjacent to the subgraph node and y nodes adjacent to the full graph node, the complexity of validating a single mapping is $O(xy\sqrt{x+y})$ in the worst case. Despite the high computational complexity of this step, the pruning power gained from it allows the LAD algorithm to outperform even the ILF algorithm, on most cases.

This paper presents a number of techniques which can be used to speed up subgraph isomorphism through the creation, strengthening, and effective use of TNFs. The problem of subgraph isomorphism is formalised in Section 2, as are other definitions and notations used in this paper. Section 3 introduces the concept of an n-neighbourhood, and proposes some novel topological features which can be calculated from it. In Section 4, the unified framework is presented, which consists of three TNF strengthening techniques. These strengthening techniques, introduced in Sections 4.1–4.3, can be applied to TNFs, as well as application-specific node labels. Section 4.4 then shows how these concepts can be combined to create strengthened features that are resistant to noise. Similar to [21], the techniques discussed in Sections 3 and 4 are designed so that they may be utilised to enhance any subgraph isomorphism algorithm.

An earlier version of this paper appeared in [27]. In this revised and extended version, we provide richer descriptions of the pruning techniques, and employ an improved algorithm for matching nodes using the tree index, which is considerably faster than its predecessor. We also introduce the novel SINEE algorithm in Section 5, which has been specifically designed to maximise the effectiveness of topological node features, as well as the strengthening framework. To evaluate the effectiveness of SINEE, Section 6 provides a more comprehensive experimental analysis than [27], both analytically and practically. Finally, in Section 7, a number of conclusions are drawn from the experimental results, and future extensions of this work are discussed.

2. Definitions and notations

The graphs used in this paper are simple (no self-loops, no duplicate edges) unlabelled graphs. However, the adaptation of the techniques discussed in this paper to non-simple graphs is trivial.

Definition 1 (*Graph*). A graph is defined as an ordered pair $G=(V,E)$, where $V=\{v_1,\ldots,v_n\}$ is a set of vertices and $E\subseteq V\times V$ is a set of edges.

The edges of a graph may be either directed $E=\{(v_x,v_y),\ldots\}$ or undirected $E=\{\{v_x,v_y\},\ldots\}$. For clarity, undirected graphs are used to introduce new concepts. The extension of these concepts to directed graphs is also given, for cases where such an extension is non-trivial.

Subgraph isomorphism detection is performed using a depth-first search. Each state in the search tree represents a permutation of mappings.

Definition 2 (*Subgraph Isomorphism State*). A subgraph isomorphism state S is a quadruple $S=(G_f,G_s,M,A)$, where G_f is the full graph, G_s is the subgraph, M is the set of valid mappings $M=\{(v_a\in V_f\rightarrow v_b\in V_s),\ldots\}$, from full graph nodes to subgraph nodes, and A is the set of assigned mappings, such that $A\subseteq M$.

At the root of the search tree is the initial state, where $A=\varnothing$. The leaf nodes of the search tree are (possibly invalid) subgraph isomorphisms, where $|A|=|V_s|$.

3. Topological *n*-neighbourhood features

A topological node feature is defined as any feature which is calculated solely from the graph topology, as viewed from a particular node. Existing TNFs utilised for subgraph isomorphism include:

- *degree*: The number of adjacent nodes.
- *clusterc (clustering coefficient)*: The number of edges between adjacent nodes (this does not include edges to the node being evaluated).
- $ncliques_k$: The number of cliques of size k that include a particular node.
- $nwalksp_k$: The number of walks of length k that pass through a particular node.

Both ncliques_k and nwalksp_k are vectors, holding values for each different k.

An n-neighbourhood (nN) of a node v is an induced subgraph formed from all the nodes that can be reached within n steps from v. This induced subgraph is centered around node v and not only contains all nodes up to n steps away, but also contains all edges between those nodes, as depicted in Fig. 1. It is denoted as $n\text{N}(v,n)$. It should be noted that while nNs are induced subgraphs, the structure they describe can be used for both induced, and non-induced, subgraph isomorphism. For each graph node v, a unique nN may be created for each value $n=1,2,\ldots,m$, where $n\text{N}(v,m)=G$ (the