Technical Report

Pattern Recognition and Image Processing Group
Institute of Computer Aided Automation
Vienna University of Technology
Favoritenstr. 9/183-2
A-1040 Vienna AUSTRIA
Phone: +43 (1) 58801-18351
Fax: +43 (1) 58801-18392
E-mail: {e0726670}@student.tuwien.ac.at
URL: http://www.prip.tuwien.ac.at/

PRIP-TR-130 September 21, 2013

Overview of Existing Software Tools for Graph Matching

Katrin Lasinger

Abstract

This report was created in the course of the seminar "Selected Chapters in Image Processing" at the Vienna University of Technology under the supervision of Walter Kropatsch. The report aims to give an overview of publicly available software toolkits for graph matching. Existing graph datasets for graph matching benchmarking as well as common data structures to store graphs are presented. Libraries and toolkits for exact and inexact matching are covered in the report and their implemented algorithms are stated.

1 Introduction

Graph matching algorithms have been used for pattern recognition tasks since the early seventies. The paper "Thirty years of graph matching in Pattern Recognition" by Conte et al. [2] gives a survey of the literature in graph matching algorithms and types of applications of graph-based techniques up to the early 2000's. Algorithms can be divided into exact and inexact matching approaches. The first group covers approaches like graph and subgraph isomorphism that require strict correspondences between two graphs or at least their subgraphs. Inexact matching algorithms have less strict constraints and tolerate some differences between the graphs. Some reasons why this might be necessary are noisy data or intrinsic variability of the patterns. For inexact matching methods we can furthermore distinguish between optimal and approximate (or suboptimal) algorithms. Optimal algorithms will always find the best solution but are usually computationally more expensive than approximate algorithms. A more recent survey of graph matching techniques, especially for Computer Vision applications, is given by Vento and Foggia [15]. Besides exact and approximate graph matching algorithms, the survey covers graph embeddings and graph kernels which have gained growing interest in recent years.

Another review of exact and inexact graph matching approaches is given by Riesen et al. [13]. In the review a graph is defined by a four-tuple $g = (V, E, \mu, \nu)$, where V is the finite set of nodes, $E \subseteq VxV$ is the set of edges, $\mu : V \rightarrow L$ is the node labeling function and $\nu : E \rightarrow L$ is the edge labeling function. This definition allows the representation of attributed, directed graphs. If unlabeled graphs are required, the same label ϵ can be assigned to all nodes and edges. For undirected graphs the reverse edges need to be inserted for each edge. Thus, arbitrary structured graphs can be handled. Typically, graph matching software tools require special types of graphs stored in a certain data format.

The aim of this report is to give an overview of available toolkits for both exact and inexact graph matching. The presented toolkits and their used algorithms are described shortly. For copyright information under which the software products are distributed, please refer to the webpages of the corresponding toolkits. Furthermore, common graph data structures are summarized in this report and available datasets for graph matching are listed. The Technical Committee #15 of the International Association for Pattern Recognition (IAPR) provides an updated list of data sets, algorithms for

graph matching and other useful tools[1]. Some of the listed items are covered in this report. However, for up-to-date links and potentially new datasets or toolkits, please refer to the webpage of the technical committee.

The remainder of this report is structured as follows. First, common data structures for different kinds of graphs are presented. In Section 3 existing datasets for benchmarking are listed. Section 4 gives an overview of existing graph matching toolkits, their functionality and requirements.

2 Graph Types and Data Structures

Different application require different data structures to represent the graph data. In addition it is not always possible that all graph properties are covered by one graph structure. E.g. directed vs. undirected graphs or labeled vs. unlabeled graphs. This section covers some of the available data structures to represent different types of graphs. Some applications, however, use their own proprietary data structures instead of standardized formats.

One existing data structure is the Graph eXchange Language (GXL) format[2], introduced by Holt et al. [8]. It is an XML based language that allows the flexible and adaptable storage and exchange of typed, attributed and directed ordered graphs. Multigraphs (multiple edges per node pair possible), hypergraphs (more than two nodes per edge) and hierarchical graphs are supported.

Another XML based file format for graphs is GraphML[3]. It supports various graph types such as typed, attributed, directed, undirected and mixed graphs. Multigraphs, hypergraphs and hierarical graphs are supported.

Text-based file formats are the Graph Modeling Language (GML)[4] and the Trivial Graph Format (TGF)[5]. GML supports attributed, directed and undirected graphs. With TGF directed graphs can be stored and at most one label per edge and node is supported. Edges can be stored in both directions to mimic undirected edges. Nested graph structures are not supported. Another common format, especially for visual applications, is the DOT graph

[1] http://iapr-tc15.greyc.fr/links.html

[2] http://www.gupro.de/GXL/

[3] http://graphml.graphdrawing.org/

[4] http://www.fim.uni-passau.de/en/fim/faculty/chairs/theoretische-informatik/projects.html

[5] http://docs.yworks.com/yfiles/doc/developers-guide/tgf.html

description language[6] with the file extension *.gv*. It is used in the graph visualization software Graphviz[7]. The DOT format can store a single attributed, directed or undirected graph per file.

Some graph software tools use compact data formats to store graphs efficiently. E.g. for unlabeled graphs, one can store the total number of nodes and subsequently for each individual node the number of edges and for each edge the ID to the corresponding node is stored. Thus, for a graph with two nodes and one directed edge one only needs to store 4 numbers (2 1 1 0 - 2 nodes, node 0 has one edge to node 1 and node 1 has no edges). When storing this graph in binary format instead of text format even more memory usage can be reduced. Such a data structure, for example, is used in the ARG database as described in Section 3. The compact data format comes with the drawback that only specific graph types are supported for certain data structures.

To convert from one graph data structure to another only few converters exist. The graph visualization software Graphviz includes converters for its DOT/GV format to GXL and GML and backwards. For details and limitations of the converters please refer to the tools *gml2gv* and *gxl2gv* in the Graphviz documentation [8]. A converter from the DOT format to GraphML is provided by Dirk Bächle [9]. The software program Wolfram Mathematica[10] supports a variety of graph data structures, amongst others GXL, GraphML, DOT, TGF and GML. Import and export of all these formats is supported and thus conversions between formats are possible. Since the different file formats are based on different graph models (e.g. not all formats support multi-edges or self loops) conversions between formats can lead to data loss.

3 Graph Datasets

To test a software toolkit, evaluate an implemented algorithm or compare the performance of different approaches graph databases are a useful resource. There already exist some databases for benchmarking graph based matching algorithms. Main differences between them are the way of generation of the

[6] `http://www.graphviz.org/content/dot-language`
[7] `http://www.graphviz.org/`
[8] `http://graphviz.org/Documentation.php`
[9] `https://bitbucket.org/dirkbaechle/dottoxml`
[10] `http://www.wolfram.com/mathematica/`

data (random or due to measurements) and the structures of corresponding graphs (for exact graph matching or more general matching tasks).

One such database, the **IAM Graph Database Repository**[11], is provided by Riesen and Bunke [11]. Graphs are stored in the Graph eXchange Language (GXL) format. The database consists of currently ten different datasets, all containing training, validation and testing subsections for supervised learning algorithms. The datasets represent line drawings, gray scale and color images, HTML web-pages, molecular compounds and proteins. Exemplary datasets are *Letter*, *Fingerprint* or *AIDS*. The datasets can be used for inexact graph matching problems.

Another extensive database of labeled as well as unlabeled graphs, the **ARG database**, was realized by the MIVIA Group of the University of Salerno[12]. The graphs were randomly generated according to different generation models. The graphs can be used for isomorphism as well as subgraph isomorphism matching. Graphs were generated pairwise so that either an isomorphism or a graph-subgraph isomorphism relation holds. For labeled graphs a non-trivial common subgraph exists. In total the database holds 168 diverse kinds of graphs with 143,600 unlabeled graphs and 166,000 labeled graphs of various node sizes. A detailed description of the generated database is provided by Foggia et al. [7] and De Santo et al. [6]. The graphs are stored in a binary format with different formats for labeled and unlabeled graphs respectively. A file is containing one graph and is composed of 16 bit words. Attributes can only be represented as integer numbers in labeled graphs. Functions to read the two file formats, together with more detailed information, are provided as C code on the webpage.

A dataset with randomly generated unlabeled graphs is the **Scale-free networks database**, introduced by Zampelli et al. [16]. Graphs are stored in text files in a compact data format as described in Section 2. The dataset is contained in the benchmarks collection of Christine Solnon to evaluate the LAD software program[13]. The dataset consists of 6 classes of instances (1 directed and 5 undirected) with each class containing 20 different instances. The classes differ in the number of nodes and the minimum and maximum node degrees. Graphs are stored in pairs (pattern and target). All but one class are feasible subgraph isomorphism problems.

[11] `http://www.iam.unibe.ch/fki/databases/iam-graph-database`

[12] `http://mivia.unisa.it/datasets/graph-database/arg-database/`

[13] `http://liris.cnrs.fr/csolnon/benchmarks.tgz`

Larrosa and Valiente [10] proposed the **LV benchmark**[14] for the subgraph isomorphism problem. The benchmark is based on the Standford GraphBase [9]. The dataset consists of 113 undirected and 59 directed graphs, providing for a total of 8211 problem instances.

There furthermore exist two different chemistry databases: the **GREYC's Chemistry dataset** provided by the GREYC laboratory[15] and various datasets by the Cheminformatics website[16].

4 Software Toolkits

Kasper Riesen et al. [12] presented the **Graph Matching Toolkit (GMT)**[17], a software toolkit for graph edit distance computation implemented in Java. Graph edit distance can be used for inexact graph matching. The software handles directed and undirected as well as labeled and unlabeled graphs. For the software toolkit five different graph edit distance algorithms are implemented. The A*-algorithm with bipartite heuristic can be used for exact graph edit distance computation, meaning that the optimal solution is found. Computationally less expensive but non-optimal approaches are beam search and bipartite graph edit distance using an assignment algorithm (Munkres', the Hungarian or the Volgenant-Jonker algorithm). The toolkit can be run from the command line or using the GUI. Graphs need to be stored in the GXL format. Thus, the IAM Graph Database Repository is supported.

Another graph matching toolkit, the C++ library **VFLib**[18], is provided by the Mivia Lab of the University of Salerno. The library implements the VF2 graph matching algorithm [3]. This algorithm conducts exact graph matching for graph and subgraph isomorphism. The algorithm can deal with unlabeled as well as with attributed relational graphs. It was tested on the ARG database. The authors claim that their algorithm is computationally efficient, especially when working with large graphs. The library includes an abstract class *ARGLoader*. Users can implement this class to write their own graph loader. Thus the library can be used with any desired graph file format. Additionally, the library already comes with some simple implemen-

[14] `http://www.lsi.upc.edu/~valiente/research.html`
[15] `https://brunl01.users.greyc.fr/CHEMISTRY/index.html`
[16] `http://cheminformatics.org/datasets/`
[17] `http://www.fhnw.ch/wirtschaft/iwi/gmt`
[18] `http://mivia.unisa.it/datasets/graph-database/vflib/`

tations of the *ARGLoader*. A simple implementation allowing basic graph editing operations, an implementation for unattributed graphs using the binary format of the ARG database and an implementation for attributed graphs are included. Further, the user can choose between nine different matching algorithms for isomorphism (VF, VF2, Ullmann, Schmidt-Druffel), graph-subgraph isomorphism (VF, VF2, Ullmann) and monomorphism (VF, VF2).

Christine Solnon provides the software program **LAD**[19] for solving the subgraph isomorphism problem. The implemented algorithm uses AllDifferent-based filtering by Solnon [14]. The filtering aims to prune branches that do not contain solutions. The original approach from Version 1 for undirected graphs was extended to directed and labeled graphs in Version 2 of the software program. According to the results on the website, the algorithm was tested on three different databases (subset of ARG database, Scale-free network database and LV database) and compared with two other approaches, one being the VF2 algorithm from the VFLib. The LAD algorithm outperforms the other two approaches in both number of solved instances and CPU time in most of the times for the given benchmarks.

The **Graph Matching Tookbox in Matlab**[20] is provided by Timothee Cour. The tool handles partial as well as full matching utilizing the kronecker bistochastic normalization algroithm by Cour et al. [4].

5 Conclusion

Six different graph data structures were presented in this report. Although many of them support arbitrary graph structures, no single format has yet been established as standard format for all software products. One reason might be the trade-off between compact and descriptive data structures. Only few converters exist and especially for proprietary formats no conversion tools are available. However, for some toolkits, like the VFLib, custom loaders can be created.

There already exists a variety of software toolkits and algorithms for exact graph matching problems and at least one for inexact graph matching, using the graph edit distance and supporting different algorithms. Four software

[19] `http://liris.cnrs.fr/csolnon/LAD.html`

[20] `http://www.timotheecour.com/software/graph_matching/graph_matching.html`

tools were presented in this paper, one of them for inexact graph matching. These tools can be a good resource for beginners in the field of graph matching or for people with little or no programming skills who want to make use of graph matching algorithms. Further the different toolkits are suitable for researchers who want to compare new algorithms with existing methods. Especially for the comparison and evaluation of new algorithms benchmarking datasets are useful resources. Five of these datasets were covered in this report.

Although present in literature [1, 5], no data structures or matching software tools for combinatorial maps and generalized maps were found. Combinatorial and generalized maps add additional topological information to the data structure by subdividing nD objects into cells (0D vertices, 1D edges, 2D faces, 3D volumes, ...). This additional topological information may be useful for 2D or 3D image processing and matching problems. Thus, software toolkits that support these data structures would be a useful resource in future.

References

[1] C. Combier, G. Damiand, and C. Solnon. From maximum common submaps to edit distances of generalized maps. *Pattern Recognition Letters*, 33(15):2020 – 2028, 2012. Graph-Based Representations in Pattern Recognition.

[2] D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(03):265–298, 2004.

[3] L. Cordella, P. Foggia, C. Sansone, and M. Vento. A (sub)graph isomorphism algorithm for matching large graphs. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(10):1367–1372, 2004.

[4] T. Cour, P. Srinivasan, and J. Shi. Balanced graph matching. *Advances in Neural Information Processing Systems*, 19:313, 2007.

[5] G. Damiand, C. Solnon, C. de la Higuera, J.-C. Janodet, and E. Samuel. Polynomial algorithms for subisomorphism of nd open combinatorial maps. *Comput. Vis. Image Underst.*, 115(7):996–1010, July 2011.

[6] M. De Santo, P. Foggia, C. Sansone, and M. Vento. A large database of graphs and its use for benchmarking graph isomorphism algorithms. *Pattern Recognition Letters*, 24(8):1067 – 1079, May 2003.

[7] P. Foggia, C. Sansone, and M. Vento. A database of graphs for isomorphism and sub-graph isomorphism benchmarking. In *Proc. of the 3rd IAPR TC-15 International Workshop on Graph-based Representations*, pages 176–187, 2001.

[8] R. C. Holt, A. Winter, and A. Schrr. Gxl: Towards a standard exchange format. In *Proceedings 7th Working Conference on Reverse Engineering (WCRE 2000)*, 2000.

[9] D. E. Knuth. *The Stanford GraphBase: A Platform for Combinatorial Computing*. ACM, New York, NY, USA, 1993.

[10] J. Larrosa and G. Valiente. Constraint satisfaction algorithms for graph pattern matching. *Mathematical Structures in Computer Science*, 12:403–422, 8 2002.

[11] K. Riesen and H. Bunke. Iam graph database repository for graph based pattern recognition and machine learning. In *Structural, Syntactic, and Statistical Pattern Recognition*, volume 5342 of *Lecture Notes in Computer Science*, pages 287–297. Springer Berlin Heidelberg, 2008.

[12] K. Riesen, S. Emmenegger, and H. Bunke. A novel software toolkit for graph edit distance computation. In *Graph-Based Representations in Pattern Recognition*, volume 7877 of *Lecture Notes in Computer Science*, pages 142–151. Springer Berlin Heidelberg, 2013.

[13] K. Riesen, X. Jiang, and H. Bunke. Exact and inexact graph matching: Methodology and applications. In C. C. Aggarwal and H. Wang, editors, *Managing and Mining Graph Data*, volume 40 of *Advances in Database Systems*, pages 217–247. Springer US, 2010.

[14] C. Solnon. Alldifferent-based filtering for subgraph isomorphism. *Artif. Intell.*, 174(12-13):850–864, Aug. 2010.

[15] M. Vento and P. Foggia. *Graph-Based Methods in Computer Vision: Developments and Applications*, chapter Graph Matching Techniques for

Computer Vision, page 141. Idea Group, Information Science Reference, Hershey USA, 2013.

[16] S. Zampelli, Y. Deville, and C. Solnon. Solving subgraph isomorphism problems with constraint programming. *Constraints*, 15(3):327–353, 2010.