Algorithm Selection for Maximum Common Subgraph

Paulius Dilkas

28th October 2017

1 Introduction

Maximum common induced subgraph for undirected graphs. Some of the graphs also contain loops and multiple edges.

2 Algorithms

Clique encoding [10]

$k \downarrow$ algorithm [5] starts by trying to solve the subgraph isomorphism problem, i.e. finding the pattern graph in the target graph. If that fails, it allows a single vertex of the pattern graph to not match any of the target graph vertices and tries again, allowing smaller and smaller pattern graphs until it finds a solution. The number of vertices of the pattern graph that are allowed this additional freedom is represented by k. More specifically, the algorithm creates a domain for each pattern graph vertex, which initially includes all vertices of the target graph and k wildcards. The domains are filtered with various propagation techniques. Then the search begins with a smallest domain (not counting wildcards), a value is chosen, and domains are filtered again to eliminate the chosen value.

McSplit [8] is a branch and bound algorithm that builds its own bit string labels for vertices in both pattern and target graphs. Once it chooses to match a vertex u in graph G_1 with a vertex v in graph G_2, it iterates over all unmatched vertices in both graphs, adding a 1 to their labels if they are adjacent to u or v and 0 otherwise. That way a vertex can only be matched with vertices that have the same labels. The labels are also used in the upper bound function by noticing that if a particular label is assigned to m vertices in G_1 and n vertices in G_2, then up to $\min\{m, n\}$ pairs can be matched for that label.

McSplit $\downarrow$ is a variant of McSplit mentioned but not explained in [8]. It is meant to be similar to $k \downarrow$ in that it starts by trying to find a subgraph isomorphism and then allows more and more of the pattern graph vertices to not be part of the solution. Based on the source code[1], there are two relatively minor differences between McSplit $\downarrow$ and McSplit:

- Instead of always looking for a larger and larger common subgraph, we have a goal size and exit early if a common subgraph of that size is found.
- The goal size is decreased if the search finishes without a solution.

3 Problem Instances

In order to determine which algorithm should be used for which problem instance, we run all algorithms on two databases that contain a large variety of graphs differing in size, various characteristics, and the way they were generated.

The McSplit paper [8] used the same datasets to compare these (and a few constraint programming) algorithms and found McSplit to win with unlabelled graphs, the clique encoding to win with labelled graphs, and McSplit $\downarrow$ to win with the `largerGraphs` dataset. However, in some cases the difference in performance between McSplit and the clique encoding or between McSplit $\downarrow$ and $k\downarrow$ was very small.

3.1 Labelled graphs

All of the labelled graphs are taken from the ARG Database [3, 4], which is a large collection of graphs for benchmarking various graph-matching algorithms. The graphs are generated using several algorithms:

- randomly generated,
- 2D, 3D, and 4D meshes,
- and bounded valence graphs.

Furthermore, each algorithm is executed with several (3–5) different parameter values. The database includes 81400 pairs of labelled graphs. Their unlabelled versions are used as well.

3.1.1 Characteristics of Graph Labelling

For the purposes of this paper, we look at two types of labelled graphs: those that have their vertices labelled and those that have both vertices and edges labelled. We define them as follows (the definitions are loosely inspired by [1]):

Definition 3.1. A graph $G = (V, E)$ is a *(vertex) labelled graph* if it has an associated vertex labelling function $\mu\colon V \to \{0, \ldots, N-1\}$ for some $N \in \{2, \ldots, |V|\}$.

[1] `https://github.com/jamestrimble/ijcai2017-partitioning-common-subgraph/blob/master/code/james-cpp/mcsp.c`

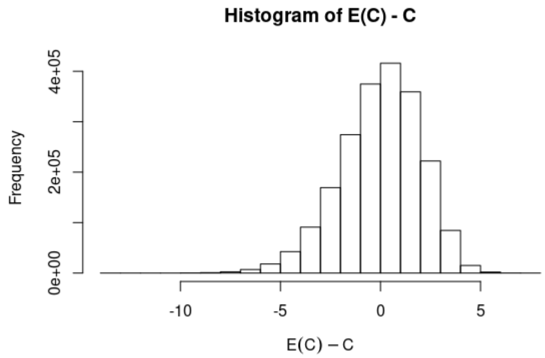

Figure 1: Histogram of the difference between the expected number of vertices assigned each label and the actual number (for all labelled graphs)

Definition 3.2. A graph $G = (V, E)$ is a *fully labelled graph* if it is a vertex labelled graph and it has an associated edge labelling function $\zeta\colon E \to \{0, \dots, M-1\}$ for some $M \in \{2, \dots, |E|\}$.

Specifically, note that:

- If a graph is labelled, then all its vertices (and possibly edges) are assigned a label.
- We are only considering finite sets of labels, represented by non-negative integers.

Now we need a way to choose N and M. For that we formally define how labelling is implemented in the ARG database:

Definition 3.3. A graph $G = (V, E)$ is said to have a $p\%$ *(vertex) labelling* if

$$N = \max\left\{2^n : n \in \mathbb{N},\, 2^n < \left\lfloor \frac{p}{100\%} \times |V| \right\rfloor\right\}.$$

The default value for p is 33%.

The publications associated with the database [3, 4] say nothing about how the labels are distributed among the N values. We calculate the number of vertices that were assigned each label for each graph (represented by C) and compare those values with the numbers we would expect from a uniform distribution (represented by $E(C)$). We plot a histogram of the difference $E(C) - C$ in Figure 1 and observe that the difference is normally distributed around 0.

3.2 Unlabelled graphs

We also include a collection of benchmark instances for the subgraph isomorphism problem[2] (with the biochemical reactions dataset excluded since we are not

[2] http://liris.cnrs.fr/csolnon/SIP.html

dealing with directed graphs). It contains only unlabelled graphs and consists of the following sets:

images-CVIU11 Graphs generated from segmented images. 43 pattern graphs and 146 target graphs, giving a total of 6278 instances.

meshes-CVIU11 Graphs generated from meshes modelling 3D objects. 6 pattern graphs and 503 target graphs, giving a total of 3018 instances. Both `images-CVIU11` and `meshes-CVIU11` datasets are described in [2].

images-PR15 Graphs generated from segmented images [12]. 24 pattern graphs and a single target graph, giving 24 instances.

LV Graphs with various properties (connected, biconnected, triconnected, bipartite, planar, etc.). 49 graphs are paired up in all possible ways, giving $49^2 = 2401$ instances.

scalefree Scale-free networks generated using a power law distribution of degrees (100 instances).

si Bounded valence graphs, 4D meshes, and randomly generated graphs (1170 instances). This is the unlabelled part of the ARG database. `LV`, `scalefree`, and `si` datasets are described in [11, 13].

phase Random graphs generated to be close to the satisfiable-unsatisfiable phase transition (200 instances) [9].

largerGraphs Large random and real-world graphs. There are 70 graphs, giving $70^2 = 4900$ instances. This set is not actually part of the main collection of benchmark instances, but is used in [5, 7, 8].

Note that this set of instances was taken from the repository[3] of [8] and has some minor differences from the version on Christine Solnon's website.

4 Features

The initial set of features was based on the algorithm selection paper for the subgraph isomorphism problem [7]:

- number of vertices,
- number of edges,
- density,
- number of loops,
- mean degree,

[3] `https://github.com/jamestrimble/ijcai2017-partitioning-common-subgraph`

- maximum degree,
- standard deviation of degrees,
- whether the graph is connected,
- mean distance between all pairs of vertices,
- maximum distance between all pairs of vertices,
- proportion of all vertex pairs that have a distance of at least 2, 3, and 4.

We excluded feature extraction running time as a viable feature by itself since it would not provide any insight into what properties of the graph affect which algorithm is likely to achieve the best performance.

Counting the number of (distinct and not) labels was later rethought to be unnecessary and replaced by a boolean feature "labelled" because if labelling is enabled, the number of labels is equal to the number of vertices and the number of distinct labels is equal to 33% of that.

4.1 Distributions of Features

In this section we plot and discuss how the selected features are distributed...

5 Selection Model

We're using LLAMA [6]. Describe k-folding.

References

[1] Zeina Abu-Aisheh. "Anytime and Distributed Approaches for Graph Matching". PhD thesis. Université François-Rabelais de Tours, 2016.

[2] Guillaume Damiand et al. "Polynomial algorithms for subisomorphism of nD open combinatorial maps". In: *Computer Vision and Image Understanding* 115.7 (2011), pp. 996–1010. DOI: `10.1016/j.cviu.2010.12.013`. URL: `https://doi.org/10.1016/j.cviu.2010.12.013`.

[3] M. De Santo et al. "A large database of graphs and its use for benchmarking graph isomorphism algorithms". In: *Pattern Recogn. Lett.* 24.8 (May 2003), 10671079. ISSN: 0167-8655. DOI: `10.1016/S0167-8655(02)00253-2`. URL: `http://dx.doi.org/10.1016/S0167-8655(02)00253-2`.

[4] P. Foggia, C. Sansone and M. Vento. "A Database of Graphs for Isomorphism and Sub-Graph Isomorphism Benchmarking". In: -. 1st Jan. 2001, 176187.

[5] Ruth Hoffmann, Ciaran McCreesh and Craig Reilly. "Between Subgraph Isomorphism and Maximum Common Subgraph". In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.* Ed. by Satinder P. Singh and Shaul Markovitch. AAAI Press, 2017, pp. 3907–3914. URL: http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14948.

[6] Lars Kotthoff. *LLAMA: Leveraging Learning to Automatically Manage Algorithms*. Tech. rep. arXiv:1306.1031. arXiv, June 2013. URL: http://arxiv.org/abs/1306.1031.

[7] Lars Kotthoff, Ciaran McCreesh and Christine Solnon. "Portfolios of Subgraph Isomorphism Algorithms". In: *Learning and Intelligent Optimization - 10th International Conference, LION 10, Ischia, Italy, May 29 - June 1, 2016, Revised Selected Papers*. Ed. by Paola Festa, Meinolf Sellmann and Joaquin Vanschoren. Vol. 10079. Lecture Notes in Computer Science. Springer, 2016, pp. 107–122. ISBN: 978-3-319-50348-6. DOI: 10.1007/978-3-319-50349-3_8. URL: https://doi.org/10.1007/978-3-319-50349-3_8.

[8] Ciaran McCreesh, Patrick Prosser and James Trimble. "A Partitioning Algorithm for Maximum Common Subgraph Problems". In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*. Ed. by Carles Sierra. ijcai.org, 2017, pp. 712–719. ISBN: 978-0-9992411-0-3. DOI: 10.24963/ijcai.2017/99. URL: https://doi.org/10.24963/ijcai.2017/99.

[9] Ciaran McCreesh, Patrick Prosser and James Trimble. "Heuristics and Really Hard Instances for Subgraph Isomorphism Problems". In: *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*. Ed. by Subbarao Kambhampati. IJCAI/AAAI Press, 2016, pp. 631–638. ISBN: 978-1-57735-770-4. URL: http://www.ijcai.org/Abstract/16/096.

[10] Ciaran McCreesh et al. "Clique and Constraint Models for Maximum Common (Connected) Subgraph Problems". In: *Principles and Practice of Constraint Programming - 22nd International Conference, CP 2016, Toulouse, France, September 5-9, 2016, Proceedings*. Ed. by Michel Rueher. Vol. 9892. Lecture Notes in Computer Science. Springer, 2016, pp. 350–368. ISBN: 978-3-319-44952-4. DOI: 10.1007/978-3-319-44953-1_23. URL: https://doi.org/10.1007/978-3-319-44953-1_23.

[11] Christine Solnon. "AllDifferent-based filtering for subgraph isomorphism". In: *Artif. Intell.* 174.12-13 (2010), pp. 850–864. DOI: 10.1016/j.artint.2010.05.002. URL: https://doi.org/10.1016/j.artint.2010.05.002.

[12] Christine Solnon et al. "On the complexity of submap isomorphism and maximum common submap problems". In: *Pattern Recognition* 48.2 (2015), pp. 302–316. DOI: 10.1016/j.patcog.2014.05.019. URL: https://doi.org/10.1016/j.patcog.2014.05.019.

[13] Stéphane Zampelli, Yves Deville and Christine Solnon. “Solving subgraph isomorphism problems with constraint programming”. In: *Constraints* 15.3 (2010), pp. 327–353. DOI: 10.1007/s10601-009-9074-3. URL: https://doi.org/10.1007/s10601-009-9074-3.