Algorithm Selection for Maximum Common Subgraph

Paulius Dilkas

5th November 2017

1 Introduction

Definition 1.1. An undirected *multigraph* is a pair (V, E), where V is a set of vertices and E is a set of edges, together with a map $E \to V \cup V^2$, which assigns one or two vertices to each edge [3]. If an edge is assigned to a single vertex, it is called a *loop*. When several edges map to the same pair of vertices, they are referred to as *multiple edges*.

We will refer to undirected multigraphs simply as graphs.

Definition 1.2. The *maximum common induced subgraph* between graphs G_1 and G_2 is a graph $G_3 = (V_3, E_3)$ such that G_3 is isomorphic to an induced subset of G_1 and G_2 and $|V_3|$ is maximised.

In this paper we will be dealing with the maximum common induced subgraph problem defined for undirected multigraphs, even though most of the benchmark instances do not have multiple edges.

2 Algorithms

The clique encoding [11] solves the maximum common subgraph problem by creating a new (association) graph and transforming the problem into an instance of maximum clique, which is then solved by a sequential version of the maximum clique solver by McCreesh and Prosser [8], which is a branch and bound algorithm that uses bitsets and greedy colouring. Colouring is used to provide a quick upper bound: if a subgraph can be coloured with k colours, then it cannot have a clique of size more than k. It should be noted that creating the association graph can use a significant amount of memory. In fact, since memory usage for a pair of graphs with n and m vertices is $\Theta(n^2m^2)$, pairs of graphs with $n \times m \geq 16000$ had to be abandoned.

$k \downarrow$ 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 [9] 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 heuristic function using the rule 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 the original paper [9]. It is meant to be similar to $k\downarrow$ in that it starts by trying to find a subgraph isomorphism and keeps decreasing the size of common subgraphs that it is interested in until a solution is found. Based on the source code[1], there are a few key differences between McSplit $\downarrow$ and McSplit:

- Instead of always looking for larger and larger common subgraphs, 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.
- Having a big goal size allows the heuristic to be more selective and prune more of the search tree branches.

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 [9] 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.

The algorithms were compiled with gcc 6.3.0 and run on Intel Xeon E5-2697A v4 (2.60 GHz) processors with 512 GB of memory and a 1000 s time limit. $k\downarrow$ was further modified to accept graphs with vertex labels.

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

3.1 Labelled graphs

All of the labelled graphs are taken from the ARG Database [4, 12], 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 $N \in \{2, \ldots, |V|\}$.

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 $\zeta\colon E \to \{0, \ldots, M-1\}$ for some $M \in \{2, \ldots, |E|\}$.

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 [4, 12] 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.

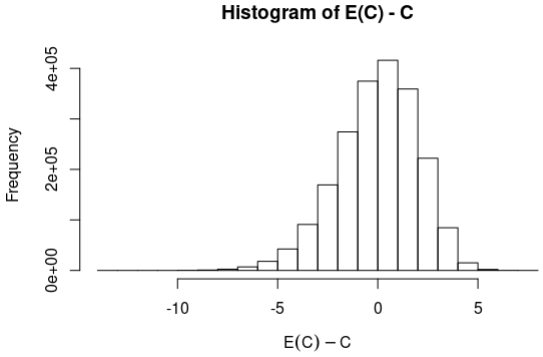

Figure 1: Histogram of the difference between the expected number of vertices assigned each label and the actual number (for all labelled graphs)

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 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 [14]. 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 [13, 15].

phase Random graphs generated to be close to the satisfiable-unsatisfiable phase transition (200 instances) [10].

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

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, 9].

Remark 3.1. This set of instances was taken from the repository[3] for the McSplit paper [9] and has some minor differences from the version on Christine Solnon's website.

Remark 3.2. Since $k \downarrow$ comes from the subgraph isomorphism problem background, it treats the two (pattern and target) graphs differently. Therefore, when graphs are not divided into patterns and targets, we run the algorithms with both orders ((G_1, G_2) and (G_2, G_1)).

4 Features

The initial set of features is based on the algorithm selection paper for the subgraph isomorphism problem [7] and consists of the following:

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

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.

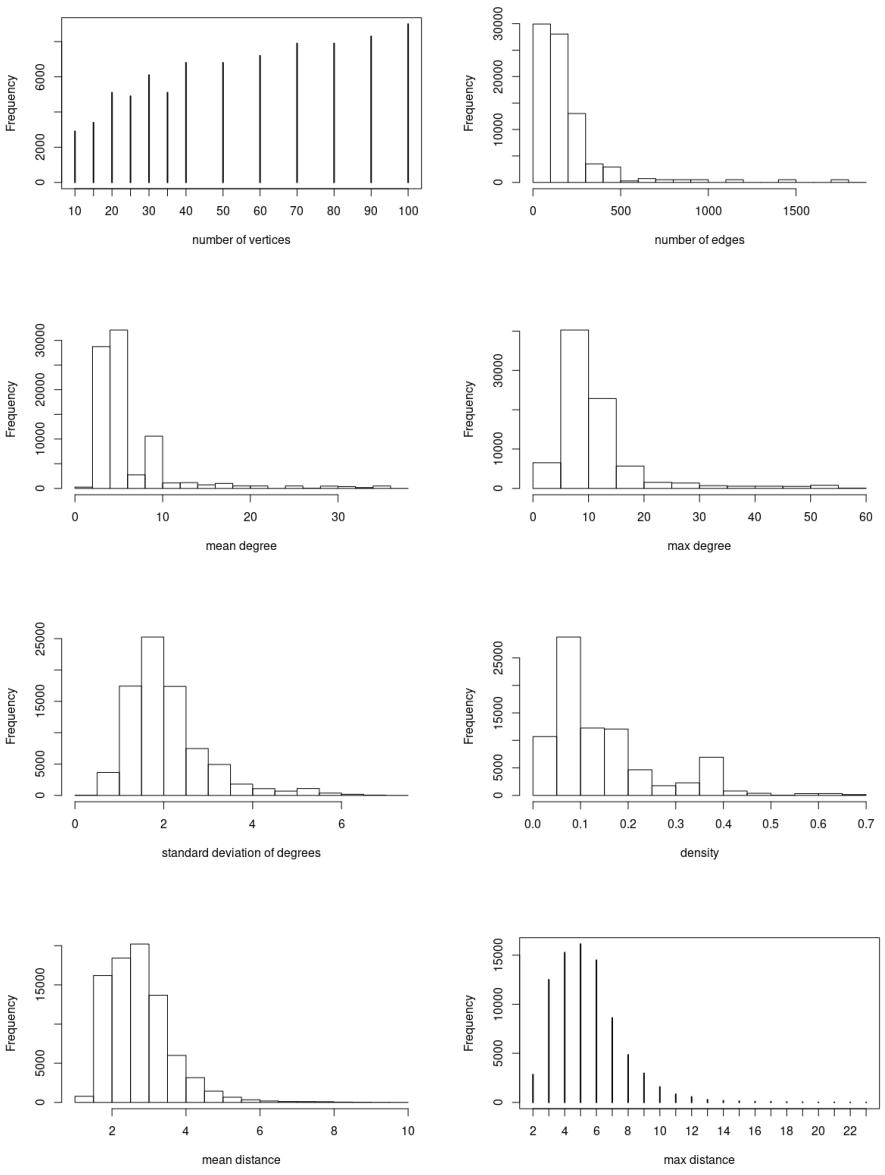

Figure 2: Plots of how various features are distributed for the labelled graphs

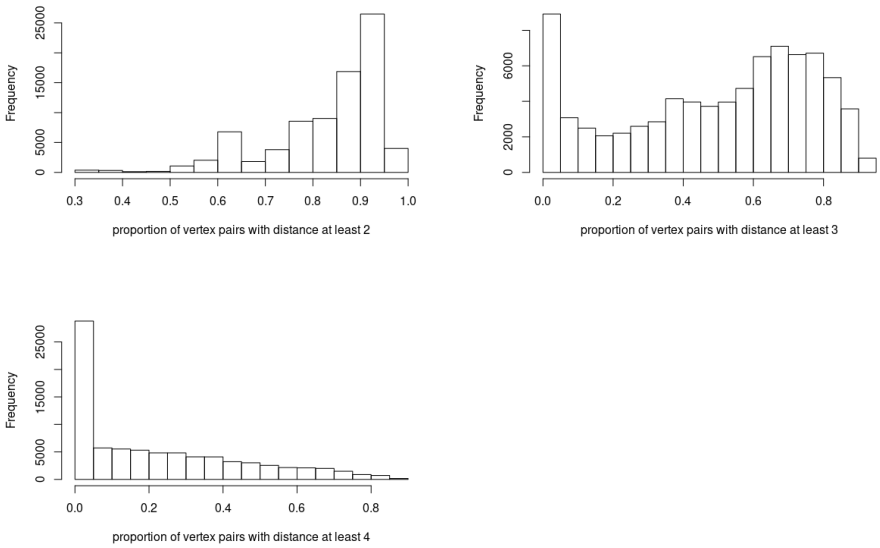

Figure 3: Plots showing typical distances between pairs of vertices for the labelled graphs

4.1 Distributions of Features

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

There are no significant differences between pattern and target graphs. Thus all of the statistical information is provided only for pattern graphs. 0.98% of the graphs have a single loop, other graphs have no loops. 99.81% of graphs are connected. The rest of the information can be found in Figures 2 and 5.

For this database of benchmark instances, we will only consider graphs that are part of a pair of graphs solved by at least one algorithm. 93.19% of graphs are connected. Since most of the data is heavily skewed, Figures 4 and **??** show the distributions as density plots.

5 Selection Model

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

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

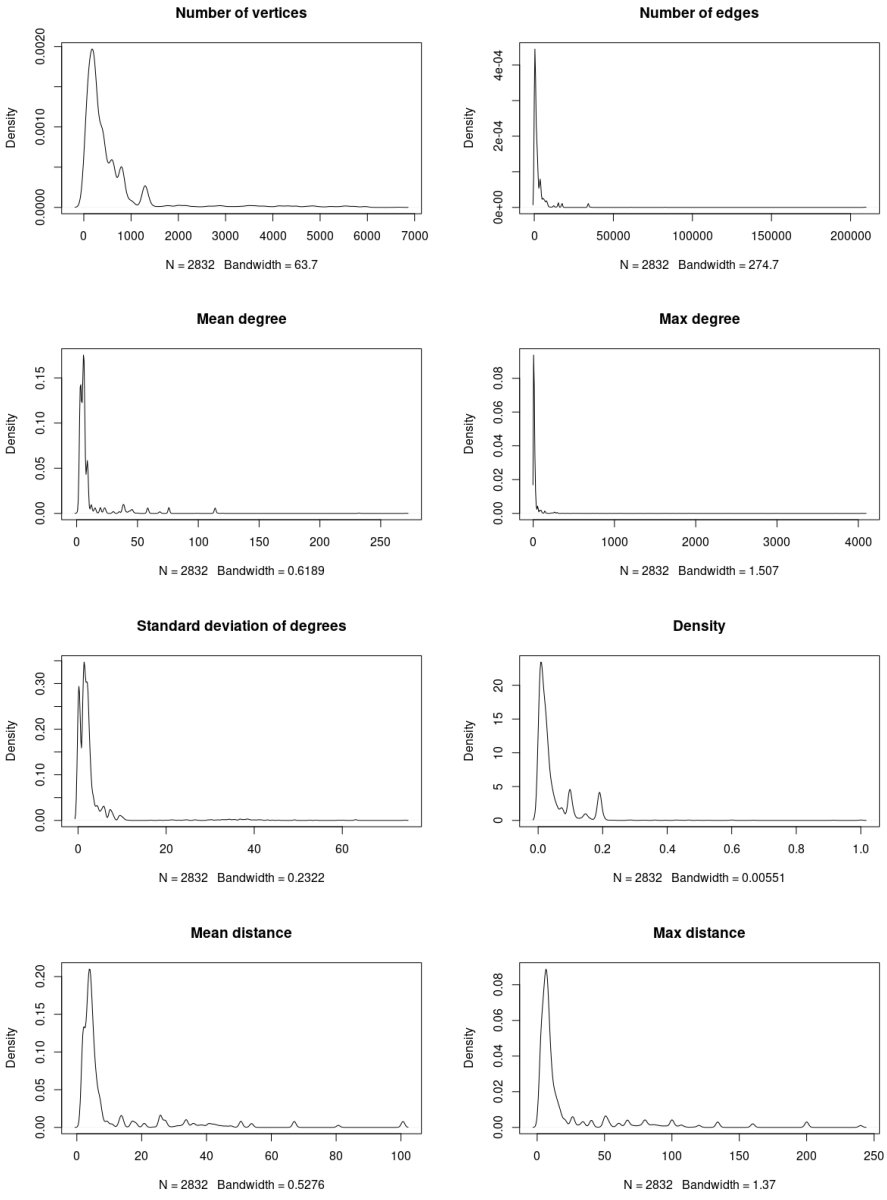

Figure 4: Plots of how various features are distributed for the unlabelled graphs

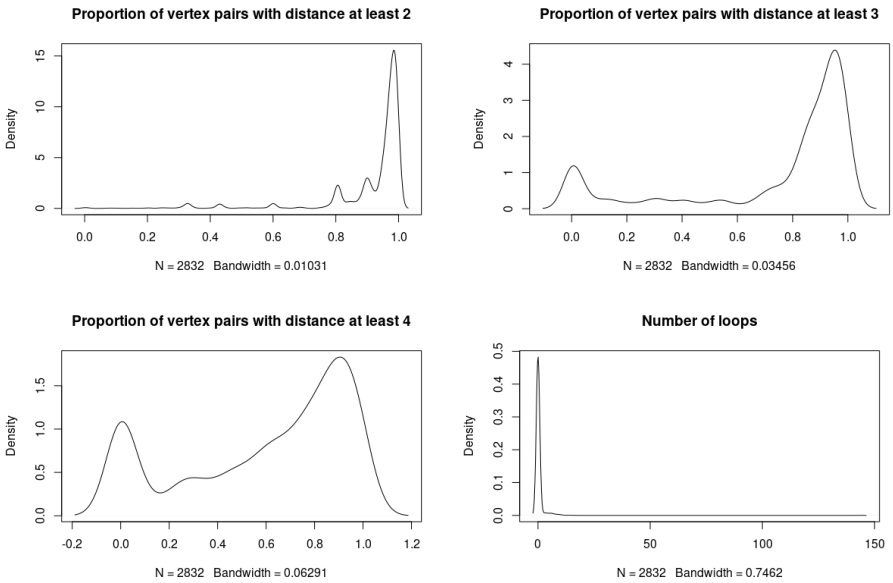

Figure 5: Plots showing typical distances between pairs of vertices and the number of loops for the unlabelled graphs

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] Reinhard Diestel. *Graph Theory, 5th Edition*. Vol. 173. Graduate texts in mathematics. Springer-Verlag, 2016, p. 28. ISBN: 978-3-662-53621-6.

[4] Pasquale Foggia, Carlo Sansone and Mario Vento. "A Database of Graphs for Isomorphism and Sub-Graph Isomorphism Benchmarking". In: *Proceedings of the 3rd IAPR TC-15 International Workshop on Graph-based Representations*. 2001.

[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 and Patrick Prosser. "The Shape of the Search Tree for the Maximum Clique Problem and the Implications for Parallel Branch and Bound". In: *TOPC* 2.1 (2015), 8:1–8:27. DOI: 10.1145/2742359. URL: http://doi.acm.org/10.1145/2742359.

[9] 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.

[10] 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.

[11] 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.

[12] Massimo De Santo et al. "A large database of graphs and its use for benchmarking graph isomorphism algorithms". In: *Pattern Recognition Letters* 24.8 (2003), pp. 1067–1079. DOI: 10.1016/S0167-8655(02)00253-2. URL: https://doi.org/10.1016/S0167-8655(02)00253-2.

[13] 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.

[14] 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.

[15] 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.