Algorithmic Data Science

Petra Mutzel
TU Dortmund, Department of Computer Science, Otto-Hahn-Str. 14, 44221 Dortmund, Germany
https://ls11-www.cs.tu-dortmund.de
petra.mutzel@cs.tu-dortmund.de

Abstract

The area of algorithmic data science provides new opportunities for researchers in the algorithmic community. In this paper we will see examples that demonstrate that algorithm engineering is the perfect basis for algorithmic data science. But there are also many open interesting questions for purely theoretically interested computer scientists. In my opinion, these opportunities should be taken because this will be fruitful for both areas, algorithmics as well as data sciences. I like to call for more participation in algorithmic data science by our community. Now we have the opportunity to shape this new emerging field.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Theory of computation → Graph algorithms analysis; Theory of computation → Complexity theory and logic; Mathematics of computing → Combinatorial algorithms

Keywords and phrases Algorithmic Data Science, Graph Similarity, Weisfeiler-Leman

Digital Object Identifier 10.4230/LIPIcs.STACS.2019.3

Category Invited Talk

Funding This work has been supported by the German Science Foundation (DFG) within the Collaborative Research Center SFB 876 "Providing Information by Resource-Constrained Data Analysis", project A6 "Resource-efficient Graph Mining".

1 Introduction

"Data is the new oil. It's valuable, but if unrefined it cannot really be used." – You may have read this quote, originally phrased by the Mathematician Clive Humby in 2006, several times. However, more than 12 years later, it is still relevant.

The ongoing digital transformation of business and society also affects science. The incoming amount of data that is stored and communicated is still increasing. Autonomously driving cars are not only recording data by various sensors but also sending their data (e.g., location, speed) to other cars that will be analysed in order to avoid critical situations. Smart home sensors are not only convenient but are very helpful for senior citizens who prefer to spend their life at home instead of institutional care. There are many other application domains, e.g., social media interaction, web mining, and video streaming systems. Concerning science, data analysis already became essential in many areas, e.g., biology, chemistry, medicine, neuroscience, linguistics, and geography.

Data science is the field responsible for extracting information out of (unstructured) data. This includes data integration, data cleaning, mathematical modelling, data analysis, and visualisation. Originally settled in the field of statistics, with increasing data sizes, also computer science and applied mathematics became increasingly relevant to the field. However, in contrast to common believe, for the analysis of the data, not only machine learning knowledge is needed. The area of data science is very broad, and not all the tasks need statistical concepts or machine learning. There are many problems for which fundamental and deep knowledge of theoretical aspects of computer science is needed.

In this paper we will see that data analysis provides many opportunities for researchers based in theoretical computer science as well as in algorithmic theory and algorithm engineering. However, there is only a very small overlap of researchers having published in both, leading theoretical computer science venues and data mining venues. The author is convinced that the field of data science will need more researchers from the theoretical computer science community. Experience shows that when co-authors from both communities join their forces, the outcome is quite profitable. For example, very recently, the mixed team Ben-David, Hrubeś, Moran, Shpilka, and Yehudayoff [8] has shown that simple scenarios of learnability are undecidable using the standard axioms of mathematics. For their proof they used the equivalence between learnability and compression. The purpose of this paper is to call for more activity of our community concerning data science tasks. In order to be more visible and to make our algorithmic basis clear, we could, e.g., start an initiative and make the name *algorithmic data science* our own.

The paper is organised as follows. Section 2 motivates the need for algorithmic data science performed by theoretical computer scientists and provides a brief introduction into data science and its main analysis tasks. Section 3 is more specific and gives an introduction to the wide area of graph similarity. The author has chosen this topic, since it is quite important in data science and machine learning, and it has many relations to different fields, since it connects graph algorithms with graph theory, algebra, descriptive logic, pebble games, linear programming relaxations, and functional analysis. Last but not least, there are plenty of interesting open questions.

2 Motivation

The vast majority of publications in data mining conferences and journals is applied. Similarly to algorithm engineering, in the top ranked publication venues also some kind of theory is mandatory. The area of algorithm engineering is relatively close to algorithmic data science, since it is about the design, theoretical analysis, implementation, and experimental evaluation of algorithms and data structures. The focus is on solving a real problem with realistic data provided by practitioners. Algorithm engineering researchers are always interested in the application of their research results. In order to be successful (in publishing at high ranked algorithmic venues) a solid basis on fundamental theoretical algorithmic knowledge is needed. It is important to be able to assess the limits of what is feasible and, if this does not meet the practical requirements, to be able to extend feasibility by means of guaranteed approvals. Alternatively, the structure of the data may help for making an approach efficient or sometimes also the change of the mathematical model of the problem. Sometimes this even leads to better practical and theoretical methods (see, e.g., subsection 3.1). Algorithmic data science essentially asks for the same. The aim is to develop new algorithms and tools for the analysis of the given data. Also for these tasks, a solid and deep knowledge of algorithmic concepts is needed. In both areas, close cooperation with the respective domain scientists is necessary.

Topics of interest for theoretical computer scientists

Blum, Hopcroft, and Kannan [9] recently published a book titled “Foundations of Data Science”, since they were convinced that the “emergence of the web and social networks as central aspects of daily life presents both opportunities and challenges for theory” ([9], p. 9). They state that in their book they “cover the theory we expect to be useful in the next 40 years” ([9], p. 9). This includes high-dimensional geometry, singular value decomposition,

random walks, sampling, sketching, streaming, clustering, graphical models, and foundations of machine learning.

In his book "Data Mining: The Textbook" [2] Aggarwal argues that the following four problems are fundamental to the process of analysing data: clustering, classification, association pattern mining, and outlier detection. However, an important basis for all of these are distances and similarity.

Clustering is the task to group the given data points so that items within the same group are more similar to each other than to outside items. Already from this definition it becomes clear that there are many possibilities to formally define the clustering problem. Clustering has been studied for a long time in the statistics community and in practical computer science (e.g., data bases, machine learning). Only recently, there is increasing interest by researchers in theoretical computer science (see, e.g., [1, 18]). Clustering has become even more popular during the big data hype, since it can be used for data sparsification and sampling approaches (e.g., [18]). Also for big data approaches on graphs, clustering plays an important role and became a research object of its own in the area of sublinear algorithms (e.g., [14]).

In contrast to clustering which belongs to so-called unsupervised learning, *classification* is a supervised learning approach. In classification a training set of data is provided whose items are labelled by its classes (groups). The aim is to train a classifier so that a new incoming item will be assigned to its correct class. Similar to clustering, also classification problems differ widely and have many applications in practice, e.g., in medicine for making decisions or predictions. Other popular applications are in spam detection. Classification is mostly studied in the machine learning community, while the related *regression problems*, where the labels are continuous, are mostly studied in statistics. However, this separation increasingly disappears. Popular approaches are support vector machines (e.g., in combination with kernel functions), k-nearest neighbour methods, logistic regression, Bayes classifiers, decision tree methods, and deep learning.

A popular data mining task is to find *frequent patterns* in a data set. A given parameter defines the *minimum support* s. An item is said to be frequent if it occurs at least s times. For example, in a graph G the set of (frequent) subgraphs (often restricted to a certain size) are those appearing at least s times in G. In a (temporal) sequence of data the considered patterns are subsequences. The aim is to find inherent regularities in the data. This is useful for deriving similarity measures on given data sets, which is important for clustering, classification, and outlier detection. *Association pattern mining* denotes a generalisation of frequent pattern mining, since it does not only rely on absolute frequencies but also on other statistical quantifications leading to association rules from a statistical perspective. The *confidence of a rule* $A \Rightarrow B$ is defined as the fraction of data points containing A that also contain B. The algorithmic aspects of frequent pattern mining problems have been studied widely in the early data mining literature. In their book [3], Aggarwal and Han provide a great survey on the current state of this topic.

Outlier detection is the problem of finding data items that are different from the majority of the given data set. Outliers may reflect errors in the data or they may belong to certain interesting rare events (e.g., in physics). Applications include fraud detection, network intrusion detection, finding unusual symptoms in medicine, or measuring errors from sensors. Also for this task, many models and approaches have been developed so far. There are statistical models (e.g., statistical tests), models based on spatial proximity (e.g., k-nearest neighbour), density-based techniques, ensemble techniques, and many more. The three above mentioned data analysis tasks clustering, classification, and association pattern mining can be used for finding outliers.

All of the above tasks rely on a notion of similarity or dissimilarity. For vectorial data or spatial data it is natural to use distance functions (two points are close to each other if their distance is small), whereas for graphs the notion of similarity is more common. In this context, also fundamental algorithmic techniques and data structures such as finding nearest neighbours and locality-sensitive hashing are useful.

It is important to develop special methods for certain data domains. For text data, pattern mining algorithms as well as compressed data structures are relevant. Of increasing interest are sequence data and time-series data recorded by sensors. For these data types, topics like sequence similarity, time series forecasting, classification, motifs, clustering, and outlier detection are of interest. Streaming algorithms are essential for dealing with continuous data streams.

Spatial data appears in the public health sector, energy and water supplies, smart cities, and many other domains. Examples are, e.g., remotely-sensed satellite images for weather forecast, climate, or crops. In summer 2018, the University Consortium for Geographic Information Science (UCGIS) published a call to action for bringing the geospatial perspective into data science degrees and curricula [47].

A natural representation for linked data sets are graphs. Some applications lead to sequences of large graphs, such as the web graph, while others to large numbers of small static graphs as in chemical molecule databases. One example of successful algorithmic data science where both communities, the data mining and the algorithmic community merged, is the computation of distances in a graph, which is important for social network analysis (e.g., centrality metrics). Other topics such as connectivity, matchings, network design and partitioning problems have not yet been brought to data science attention, although there will be potential. Concerning the classical basic graph algorithms, Skiena states: "I have not seen these tools applied as generally in data science as I think they should be" ([45], p. 323). A possible reason for this may be that the graphs tend to be quite large. On the other hand, this maybe overcome with graph sketching, graph sparsification, graph sampling, sublinear graph algorithms, and network summarisation. All of these are topics studied in the theoretical computer science community and of interest to data science.

3 Graph Similarity

The basis for many data analysis tasks for graphs and networks like clustering, classification and outlier analysis is graph similarity. One can think of many different models and definitions for this depending on the current applications. Although some similarity notions have been studied from practical and theoretical perspectives already, there are still many open questions.

3.1 Isomorphism based Approaches

Graph similarity is closely related to graph isomorphism. In the data mining literature, the notion of *exact graph matching* is devoted to find strict correspondences between two graphs being matched, or at least their subgraphs.

Graph isomorphism

Obviously, two graphs are most similar if they are isomorphic to each other.

▶ **Definition 1.** *Let $G = (V_G, E_G)$ and $H = (V_H, E_H)$ be two simple graphs. A bijective map $\pi : V_G \to V_H$ is a* graph isomorphism *if the following holds:*

$\forall\, v, w \in V_G : (v, w) \in E_G \iff (\pi(v), \pi(w)) \in E_H$.

The graph isomorphism problem *asks the question if a graph isomorphism between two given graphs exists.*

The complexity of the graph isomorphism problem is still open. NP-completeness of the problem would result in the collapse of the polynomial time hierarchy to its second level [41]. There are quite a few special graph classes for which the graph isomorphism problem is known to be solvable in polynomial time such as planar graphs [26], bounded degree graphs [33], and graphs of bounded tree width [10]. A quite general result by Grohe and Marx shows that the graph isomorphism problem can be solved in $O(n^{f(|H|)})$ time for H-topological-minor-free graphs [22], where f denotes a function. For a long time, fixed-parameter-tractability of the graph isomorphism problem concerning the parameter tree width was open. Recently, Lokshtanov et al. [32] solved the problem by suggesting a graph canonisation approach leading to a $2^{O(k^5 \log k)} n^5$ time algorithm that either solves the graph isomorphism problem for two given graphs or concludes that one of the graphs does not have tree width bounded by k. Grohe et al. [23] have improved this result to an isomorphism test for graphs with tree width k with running time $O(2^{k\ \mathrm{polylog}(k)}\mathrm{poly}(n))$.

The theoretically best algorithm for general graphs known is the quasi-polynomial time algorithm by Babai [6]. Surprisingly, most practical instances on general graphs can be solved quite fast. However, most of the graph pairs provided for graph analysis are not isomorphic to each other and we are interested in their similarity.

Maximum common subgraph

A natural extension of graph isomorphism to graph similarity between G and H is to search for the largest subgraph that is contained in G and H. This gives rise to the *maximum common subgraph problem.*

▶ **Definition 2.** *Let $G = (V_G, E_G)$ and $H = (V_H, E_H)$ be two simple graphs. A graph C is called a* common subgraph *if there exist two vertex sets $R \subseteq V_G$ and $S \subseteq V_H$ so that the induced subgraphs $G[R]$ and $H[S]$ are isomorphic to C. The common subgraph of largest size is called the* maximum common subgraph.

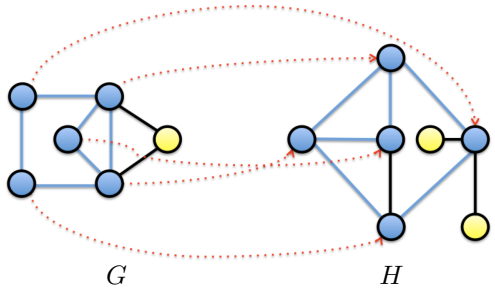


Figure 1 Two graphs and their maximum common subgraph displayed with blue vertices. The red darts demonstrate the mapping between the two isomorphic subgraphs.

Figure 1 shows two graphs and their maximum common subgraph. In contrast to the graph isomorphism problem, the maximum common subgraph problem is well-known to be NP-hard. Also here many variants are possible by asking for non-induced subgraphs

or introducing weights to the vertices and edges. Kann studied the complexity of several variations of the maximum common subgraph problem including approximation [27]. In practice, researchers often use the relationship between the maximum common subgraph problem and the maximum clique problem in the product graph of $G = (V_G, E_G)$ and $H = (V_H, E_H)$. If the graphs are small, then the algorithm by Bron and Kerbosch [12] can be used which enumerates the set of all maximal cliques in a graph. For larger graphs, this can be combined with branch-and-bound techniques. The theoretical fastest clique algorithm by Robson [40] leads to running time $O(2^{0.249|V_G||V_H|})$. However, this algorithm has not been published so far and is quite involved. Therefore, the result by Fomin, Grandoni, and Kratsch [19] is of interest for practitioners, since it introduces a simple algorithm with running time $O(2^{0.288|V_G||V_H|})$. In [30] Kriege suggested an algorithm based on graph canonisation with running time $O(2^{V_H+V_G^{1/2+o(1)}})$ assuming that $|V_G| \leq |V_H|$.

Concerning special graph classes, the maximum common subgraph problem remains NP-hard even if both given graphs are trees [11]. However, if the output is restricted to be connected, then this problem can be solved in polynomial time [35]. Akutsu and Tamura [4] have shown that the maximum common connected subgraph problem is NP-hard in vertex-labeled partial 11-trees of bounded degree. Kriege et al. [28] have shown that the problem remains NP-hard in biconnected partial 2-trees with all but one vertex of degree three or less.

Figure 2 The structure of the caffeine molecule and its molecule graph with vertex labels (atoms) and edge attributes (single or double bonds).

In practice, this problem is highly relevant for chemical molecule databases used for drug design. Molecule graphs are often outerplanar, almost all of them are planar. They have bounded tree width and vertex degree. Very important are the vertex and edge labels corresponding to atoms and activity attributes (see Fig. 2). Chemists want to find small molecules having a similar function as a given molecule or they want to conduct high-throughput screening in order to find promising candidates. Graph similarity approaches work well for answering these type of questions, since there is a direct connection between the structure (atoms and their bonds) and the effect of a molecule. When studying the chemical problem, it turned out that a restricted version of the maximum common subgraph problem which preserves the blocks (maximal biconnected components) and the bridges of the input graphs, is even of more relevance to the chemists. Luckily, this restricted version can be solved in polynomial time [28] in contrast to the original stated problem. For outerplanar bounded degree graphs, the block-and-bridge preserving maximum common connected subgraph can be computed in quadratic time [16]. These examples show that looking at the practical data as well as analysing the given practical problem may often help to find theoretically and practically useful algorithms. A new direction relevant to molecular graphs is to further relax the restriction of isomorphism and instead only require a homeomorphism (see, e.g., [17]). Recently, Kriege, Humbeck and Koch [29] have provided a survey on chemical similarity and substructure search in the area of drug design.

3.2 Distance based Approaches

Distance based approaches for graph comparison have been used widely in the machine learning community and in certain application domains like bioinformatics.

Frobenius distance

A natural distance measure between two graphs G and H is to search a permutation of the vertex set of G so that the number of edge mismatches is minimised. The *Frobenius distance* between two graphs investigated in [25] takes up this idea. Here, a permutation π of the rows and columns of the adjacency matrix A_G is searched that minimises the Frobenius norm of the matrix $A_G^\pi - A_H$. Although this problem has been extensively studied in the machine learning literature (also called *graph matching*) only few theoretical results are known. Recently, Grohe, Rattan, and Woeginger [25] have investigated the complexity of this and related problems. They have shown that this graph similarity problem is NP-hard even if both input graphs are trees or if one input graph is a path. On the other hand, the authors show that in the case that the two graphs are a path and a tree, the problem can be solved in polynomial time. On the positive side, they also show that the weighted version (taking weights of the edges into account) of the graph similarity problem related to the Frobenius norm is tractable if both adjacency matrices are positive semidefinite and have bounded rank, and where one of the matrices has a bounded clustering number. For details, please see [25]. Many problems in this area are worth further studying. It would be of interest if the problem or some restrictions can be approximated in polynomial time. Also results concerning fixed-parameter-tractability would be of great interest for algorithmic data science.

Graph edit distance

A more general distance is the *graph edit distance* in which the goal is to transform graph G into graph H by adding, substituting or deleting vertices and edges with the smallest total cost. This very general problem is often used in bioinformatics (e.g., for sequence comparisons), since it has the advantage that arbitrary vertex and edge attributes (e.g., labels) and many different cost functions and edit operations can be taken into account. However, this flexibility also comes with costs. The graph edit distance is a generalisation of the maximum common subgraph problem, which is NP-complete and hard to approximate with given guarantees [27]. Practical approaches for computing the graph edit distance of two given graphs are often based on backtracking or tree search algorithms and work for small graphs only. Recently, a binary linear programming formulation for computing the graph edit distance has been proposed (Lerouge et al., 2017), which allows to compare graphs of moderate size using state-of-the art general purpose solvers.

3.3 Weisfeiler-Leman Approaches

Despite the fact that the complexity of the graph isomorphism problem is open, experience shows that the problem can be solved quite fast in practice for most instances. The basis for many of the practical algorithms (e.g., nauty [36]) as well as for the quasi-polynomial time algorithm by Babai [6] is the vertex colouring algorithm by Weisfeiler-Leman[1]. The hypothesis

[1] In the literature found as Lehman as well as Leman

that similar graphs tend to have similarly coloured vertex sets led to its usage for classification tasks. For many practical tasks, Weisfeiler-Leman based classification (also called *colour refinement*) successfully competes with or even dominates the best state-of-the-art methods.

Basic Weisfeiler-Leman

The *Weisfeiler-Leman algorithm* (WL) simultaneously colours the vertices of the two given graphs iteratively. In the beginning, all vertices get the same colour c. In each iteration, the vertex sets of each colour class are further separated. This is done by looking at the neighbours of each vertex. E.g., if a vertex v has three neighbours of a colour c while vertex w has only two neighbours of colour c, then v and w will get different colours. The algorithm stops if the colour classes do not change anymore. If now the two given graphs have different colour classes, we know that these graphs cannot be isomorphic to each other.

Figure 3 shows two graphs and their colouring after the second iteration. By looking at the colour histograms of each graph, which are different for our example, it becomes clear that the two graphs cannot be isomorphic to each other. So we do not even need to compute further iterations. We say that the Weisfeiler-Leman algorithm *distinguishes the two graphs* if the colour patterns are different.

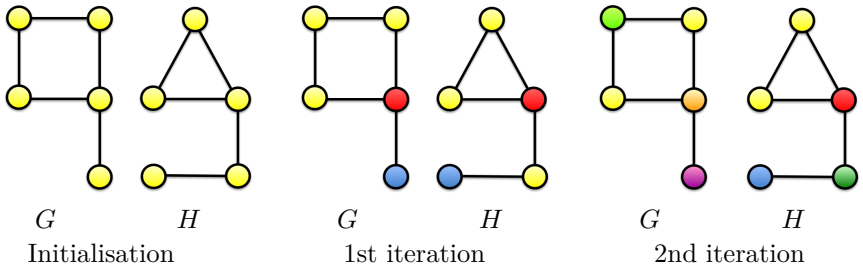
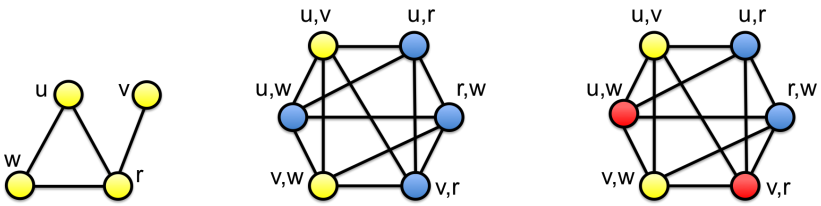


Figure 3 The first iterations of the WL-algorithm for two graphs G and H.

In the case that the colour classes are identical after the final round of the Weisfeiler-Leman algorithm, we cannot be sure if both graphs are isomorphic to each other. Consider, for instance, a k-regular graph (all vertices have degree k). The algorithm would stop after the first round, since every vertex has the same number of neighbours of colour c. Hence all vertices get the same colour. Hence, the WL algorithm can be used as a heuristic with one-sided error for solving the graph isomorphism problem.

WL has the nice property that two random graphs will end up with different colour classes with high probability [7]. In data analysis, the algorithm is been used for solving classification problems via graph kernels. Graph kernels have been used with established learning algorithms such as support vector machines and have proven to be a key technique for solving classification and prediction tasks on graphs [43].

A graph kernel is a similarity measure between graphs, which can be represented as a dot product between feature vectors obtained from the graphs. The colour histograms after every round of the WL-algorithm directly provide such a feature vector. E.g., in our example the feature vector after the first round for G would be $(3, 1, 1)$ (three yellow, one red, one blue) and after the second round $(2, 0, 0, 1, 1, 1, 0)$ (because there is no red, blue or dark green coloured vertex in G). In order to get one feature vector, we can simply concatenate the two feature vectors, hence we get $(3, 1, 1, 2, 0, 0, 1, 1, 1, 0)$. We could also scale some of these vectors up or down. The similarity measure is then given by the dot product of these feature vectors. By doing this for all pairs of graphs in a given data set, we get a similarity function that can be plugged into a learning algorithm, such as a support vector machine.



G initialisation 2-WL 1st iteration 2-WL

Figure 4 An example for the first rounds of 2-WL (set-based) for a graph G.

Higher dimensional Weisfeiler-Leman approaches

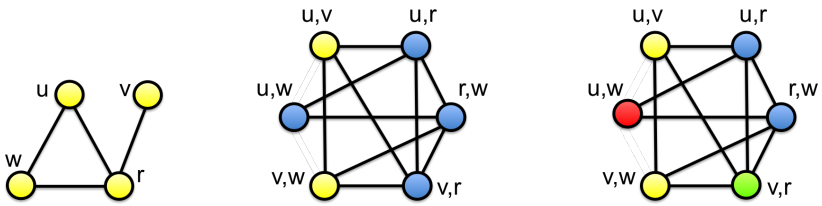
We get a stronger version of the WL algorithm by colouring the set of all k-tuples or k-sets of vertices. There are many different possibilities to generalise the WL approach to k dimensions by defining the neighbourhood of the elements. So be aware about the different definitions in the literature, in particular, if results from other papers concerning the k-WL are used. Most results may be true for the various definitions, but not all of them.

For simplicity we discuss the version in which we consider the set of all unordered k-sets instead of tuples. Then we say that two k-sets of vertices are neighbours if they differ in exactly one element. Initially, the k-sets R and S get the same colour if the induced graphs $G[R]$ and $H[S]$, respectively, are isomorphic to each other. In each iteration two k-sets R and S of the same colour get different colours, if there exists a colour c for which R and S have a different number of neighbours coloured c. Figure 4 shows an example for $k = 2$.

It turns out that, in general, the k-WL algorithm is stronger than the original WL-algorithm in the sense that it is able to distinguish two non-isomorphic graphs (answer "not isomorphic") whenever the original is able to do so. Moreover, for large enough k, the generalised approach would be able to solve the graph isomorphism problem. However, note that already the initialisation phase for $k = n$ asks for solving the graph isomorphism problem. Babai in his quasi-polynomial algorithm uses the k-tuple WL for $k = O(\log n)$ and many other involved algebraic techniques [6]. Cai, Führer, and Immerman [13] have shown that for every k there exist 3-regular graphs G_k and H_k of size $O(k)$ that are not distinguishable by the k-WL. Altogether we can say that the k-WL for $k \geq 2$ is quite strong, but it is also slow to compute.

Because the k-WL algorithm is too slow for using it for classification tasks in data analysis, we have suggested a local version which takes the graph structure into account [37]. Here, we say that two k-sets are neighbours if they differ in exactly one element and there is an edge from a vertex in R (resp. S) to a vertex in S (resp. R). So our new local kernel takes both, local and global graph properties into account. Since for sparse graphs the neighbourhood of a k-set in the local WL is much smaller compared to the neighbourhood in the original k-WL, the algorithm runs much faster on such graphs. Our experiments on several graph classification benchmarks have shown that our kernels often outperform the state-of-the-art in terms of classification accuracies.

Surprisingly, in our experiments, our local version concerning k-sets (we tested for $k = 2, 3$) was at least as strong as the global k-WL and often even stronger. Also the number of colour classes of our local version is in general larger than that of k-WL (see, e.g., Fig. 5). This is nice, since in the best case, every vertex gets a different colour; and then it is easy to distinguish two non-isomorphic graphs. Currently, we are investigating the relationships more deeply; our theoretical as well as new empirical results can be found in [38]. Observe that there are quite a few different definitions of the k-WL in the literature, which differ in their strength. For more information, see, e.g., [21].



G initialisation 2-LWL 1st iteration 2-LWL

Figure 5 An example for the first rounds of the local 2-LWL (set-based) for a graph G.

Linear programming provides a nice relationship between graph isomorphism and the outcome of the Weisfeiler-Leman algorithm. There is a natural integer linear program for the graph isomorphism problem in which the nonnegative integer solutions of the equation system correspond to permutation matrices. Tinhofer [46] has denoted the *nonnegative fractional* solutions of the LP-relaxation of this integer linear program *fractional isomorphisms*. He has shown that two graphs are fractional isomorphic if and only if the Weisfeiler-Leman algorithm is not able to distinguish them.

Weisfeiler-Leman and its relation to descriptive complexity

Cai, Han, and Führer [13] have revealed interesting connections between the k-WL and descriptive complexity. They provide a linear lower bound for the number of variables needed for first-order logic with counting to distinguish a sequence of pairs of graphs G_n and H_n. The authors prove the equivalence of the k-variable language with counting and the $(k-1)$-dimensional WL. Then they introduce combinatorial pebble games and prove that they are logically equivalent in the considered languages.

Atserias and Maneva [5] provide an equation system related to the generalisation of the Weisfeiler-Leman algorithm to k-tuples which is in close relation to the level-k Sherali-Adams relaxation of Tinhofer's equation system. More precisely, they show that the levels of the Sherali-Adams hierarchy of linear programming relaxations applied to Tinhofer's equation system interleave with the levels of k-dimensional Weisfeiler-Leman, and in addition with the levels of indistinguishability in a logic with counting quantifiers and bounded number of variables. The former results have also been obtained by Malkin [34] using polyhedral arguments. Grohe and Otto [24] have further simplified the arguments and strengthened the above result using a modified k-pebble counting game.

Counting homomorphisms

In the graph mining community, graph similarity is often measured by counting small subgraphs. For example, for a given input graph, the number of triangles, paths of length 4, and subtrees of certain sizes and structure are counted. These counts provide the input values for a feature vector for this graph. The dot product yields a graph kernel which can then be used for graph classification using a support vector machine (e.g., see [44]).

Dell, Grohe, and Rattan [15] suggest to count homomorphism vectors restricted to certain subgraphs instead. Their motivation is a classical result due to Lovász stating that a graph G can be characterised by counting the homomorphisms from the set of all graphs F to G. The authors show that if the homomorphism vectors are restricted to trees, then the feature vectors of the homomorphism counts of two graphs are identical if and only if the Weisfeiler-Leman algorithm does not distinguish the graphs. The LP-relaxation of the natural

integer linear program for the graph isomorphism problem has a rational solution if and only if the two feature vectors of the homomorphism counts are identical [15]. Moreover, the authors generalised their result to the k-dimensional Weisfeiler-Leman algorithm. For this, they restrict the homomorphism counts to the graph class of all graphs of tree width at most k. Then they show that the following three statements are equivalent:

(i) The feature vectors of the homomorphism counts restricted to the graph class of graphs with tree width at most k of two given graphs are identical.
(ii) The k-dimensional Weisfeiler-Leman algorithm does not distinguish both graphs.
(iii) The system of linear equations has a nonnegative solution.

The following results have been proven for one direction only: In the case that the system of linear equations has a real solution, then the feature vectors of the homomorphism counts restricted to the graph class of graphs with path width at most k of two given graphs are identical. It is still open if the converse holds. The authors state that it would be very interesting to study the graph similarity measures induced by the homomorphism vectors [15].

Weisfeiler-Leman and deep learning

Deep neural networks are incredibly popular these days, since they have led to qualitative breakthroughs on a wide variety of tasks. They are among the most successful machine learning approaches for voice recognition and image recognition. Today they are used for almost any application related to sound, text, images, videos, graphs, and time series. Although there is plenty of successful empirical research on a wide variety of applications, there are only few theoretical papers explaining their behaviour.

Gilmer et al. [20] have suggested a *message passing neural network* for graphs with vertex and edge features which covers most of the current *graph neural networks*. Similar to Weisfeiler-Leman, also these graph neural networks are based on neighbourhood aggregation. In contrast to WL, the aggregation functions do not need to be discrete, and the architecture of the T-layered network defines the neighbourhood.

Morris et al. [39] have studied the relationship between Weisweiler-Leman and graph neural networks. The authors have shown that graph neural networks can be viewed as a neural version of the Weisfeiler-Leman algorithm, in which the colours are replaced by continuous feature vectors. The neural networks aggregate over the vertex neighbourhoods. The paper shows that although the graph neural networks are more flexible concerning learning tasks, they are not able to distinguish pairs of non-isomorphic graphs that cannot be distinguished by Weisfeiler-Leman. On the other hand, there exist architectures and parameters so that graph neural networks have the same strength as Weisfeiler-Leman. Based on their observation, the authors have suggested so-called *k-graph neural networks* as well as new hierarchical versions, and prove an equivalence concerning strength to the k-dimensional Weisfeiler-Leman approach. Their experimental study has revealed that their new hierarchical approach is able to dominate the state-of-the-art approaches. This a nice example where theory does lead to innovative approaches improving the practical state-of-the-art.

4 Conclusion

This paper tries to motivate the readers to get interested to the area of algorithmic data science. There are plenty of opportunities for achieving new theoretical results as well as practical impact. The paper should not be seen as a survey on the area of data analysis on graphs but rather as an attempt to get the reader interested in algorithmic data science. If you want to learn a bit more, you can find some suggestions in the following.

Skiena [45] wrote a great book that gives a nice introduction into data science focusing on the skills and principles needed for the whole process including data integration, data cleaning, data analysis and visualisation. In particular, the author provides intuition for the presented statistical and mathematical concepts. The recent books by Blum, Hopcroft, and Kannan [9] as well as Aggarwal [2] dig deeper into theory (see section 2). Leskovec, Rajaraman and Ullman [31] cover certain aspects concerning Web mining including the technology of search engines like link-spam detection and recommendation systems. Other interesting books are, e.g., [48] and [42].

References

1 Marcel R. Ackermann, Johannes Blömer, Daniel Kuntze, and Christian Sohler. Analysis of Agglomerative Clustering. *Algorithmica*, 69(1):184–215, 2014. doi:10.1007/s00453-012-9717-4.

2 Charu C. Aggarwal. *Data Mining: The Textbook*. Springer Publishing Company, Incorporated, 2015.

3 Charu C. Aggarwal and Jiawei Han. *Frequent Pattern Mining*. Springer Publishing Company, Incorporated, 2014.

4 Tatsuya Akutsu and Takeyuki Tamura. A Polynomial-Time Algorithm for Computing the Maximum Common Subgraph of Outerplanar Graphs of Bounded Degree. In Branislav Rovan, Vladimiro Sassone, and Peter Widmayer, editors, *37th International Symposium on Mathematical Foundations of Computer Science, MFCS 2012*, pages 76–87, Berlin, Heidelberg, 2012. Springer.

5 Albert Atserias and Elitza Maneva. Sherali–Adams Relaxations and Indistinguishability in Counting Logics. *SIAM Journal on Computing*, 42(1):112–137, 2013. doi:10.1137/120867834.

6 László Babai. Graph isomorphism in quasipolynomial time. In *Proceedings of the Forty-eighth Annual ACM Symposium on Theory of Computing, STOC 2016*, pages 684–697, New York, NY, USA, 2016. ACM.

7 László Babai, Paul Erdös, and Stanley M. Selkow. Random Graph Isomorphism. *SIAM Journal on Computing*, 9(3):628–635, 1980. doi:10.1137/0209047.

8 Shai Ben-David, Pavel Hrubes, Shay Moran, Amir Shpilka, and Amir Yehudayoff. Learnability can be undecidable. *Nature Machine Intelligence*, 1:44–48, 2019. doi:10.1038/s42256-018-0002-3.

9 Avrim Blum, John Hopcroft, and Ravi Kannan. Foundations of Data Science, 2018. URL: https://www.cs.cornell.edu/jeh/book.pdf.

10 Hans L. Bodlaender. Polynomial algorithms for graph isomorphism and chromatic index on partial k-trees. *Journal of Algorithms*, 11(4):631–643, 1990. doi:10.1016/0196-6774(90)90013-5.

11 Franz J. Brandenburg. Subgraph isomorphism problems for k-connected partial k-trees. Unpublished manuscript, 2000.

12 Coen Bron and Joep Kerbosch. Algorithm 457: finding all cliques of an undirected graph. In *CACM*, 1973.

13 Jin-yi Cai, Martin Fürer, and Neil Immerman. An optimal lower bound on the number of variables for graph identificationS. *Combinatorica*, 12(4):389–410, 1992.

14 Artur Czumaj, Pan Peng, and Christian Sohler. Testing Cluster Structure of Graphs. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015*, pages 723–732. ACM, 2015. doi:10.1145/2746539.2746618.

15 Holger Dell, Martin Grohe, and Gaurav Rattan. Lovász Meets Weisfeiler and Leman. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*, volume 107 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 40:1–40:14,

Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ICALP.2018.40.

16 Andre Droschinsky, Nils Kriege, and Petra Mutzel. Finding Largest Common Substructures of Molecules in Quadratic Time. In Bernhard Steffen, Christel Baier, Mark van den Brand, Johann Eder, Mike Hinchey, and Tiziana Margaria, editors, *SOFSEM 2017: Theory and Practice of Computer Science - 43rd International Conference on Current Trends in Theory and Practice of Computer Science*, volume 10139 of *Lecture Notes in Computer Science*, pages 309–321. Springer, 2017. doi:10.1007/978-3-319-51963-0_24.

17 Andre Droschinsky, Nils M. Kriege, and Petra Mutzel. Largest Weight Common Subtree Embeddings with Distance Penalties. In Igor Potapov, Paul G. Spirakis, and James Worrell, editors, *43rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2018*, volume 117 of *LIPIcs*, pages 54:1–54:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018. doi:10.4230/LIPIcs.MFCS.2018.54.

18 Dan Feldman, Melanie Schmidt, and Christian Sohler. Turning big data into tiny data: Constant-size coresets for k-means, PCA and projective clustering. In Sanjeev Khanna, editor, *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013*, pages 1434–1453. SIAM, 2013. doi:10.1137/1.9781611973105.103.

19 Fedor V. Fomin, Fabrizio Grandoni, and Dieter Kratsch. Measure and Conquer: A Simple $O(2^{0.288N})$ Independent Set Algorithm. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm*, SODA '06, pages 18–25, Philadelphia, PA, USA, 2006. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=1109557.1109560.

20 Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural Message Passing for Quantum Chemistry. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70 of *Proceedings of Machine Learning Research (PMLR)*. PMLR, 2017.

21 Martin Grohe. *Descriptive Complexity, Canonisation, and Definable Graph Structure Theory*. Lecture Notes in Logic. Cambridge University Press, 2017.

22 Martin Grohe and Daniel Marx. Structure Theorem and Isomorphism Test for Graphs with Excluded Topological Subgraphs. *SIAM Journal on Computing*, 44(1):114–159, 2015. doi:10.1137/120892234.

23 Martin Grohe, Daniel Neuen, Pascal Schweitzer, and Daniel Wiebking. An Improved Isomorphism Test for Bounded-Tree-Width Graphs. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018*, volume 107 of *LIPIcs*, pages 67:1–67:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018. doi:10.4230/LIPIcs.ICALP.2018.67.

24 Martin Grohe and Martin Otto. Pebble games and linear equations. *The Journal of Symbolic Logic*, 80(3):797–844, 2015. doi:10.1017/jsl.2015.28.

25 Martin Grohe, Gaurav Rattan, and Gerhard J. Woeginger. Graph Similarity and Approximate Isomorphism. In Igor Potapov, Paul G. Spirakis, and James Worrell, editors, *43rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2018*, volume 117 of *LIPIcs*, pages 20:1–20:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018. doi:10.4230/LIPIcs.MFCS.2018.20.

26 John E. Hopcroft and Joseph K. Wong. Linear Time Algorithm for Isomorphism of Planar Graphs (Preliminary Report). In *Proceedings of the Sixth Annual ACM Symposium on Theory of Computing, STOC 1974*, pages 172–184, New York, NY, USA, 1974. ACM. doi:10.1145/800119.803896.

27 Viggo Kann. On the approximability of the maximum common subgraph problem. In Alain Finkel and Matthias Jantzen, editors, *STACS 92*, pages 375–388, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg.

28 Nils Kriege, Florian Kurpicz, and Petra Mutzel. On maximum common subgraph problems in series-parallel graphs. *European Journal of Combinatorics*, 68:79–95, 2018. Combinatorial Algorithms, Dedicated to the Memory of Mirka Miller. doi:10.1016/j.ejc.2017.07.012.

29 Nils M. Kriege, Lina Humbeck, and Oliver Koch. Chemical Similarity and Substructure Searches. In Shoba Ranganathan, Michael Gribskov, Kenta Nakai, and Christian Schönbach, editors, *Encyclopedia of Bioinformatics and Computational Biology*, pages 640–649. Academic Press, Oxford, 2019. doi:10.1016/B978-0-12-809633-8.20195-7.

30 Nils Morten Kriege. *Comparing Graphs: Algorithms & Applications*. Phd thesis, TU Dortmund, 2015.

31 Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of Massive Datasets*. Cambridge University Press, 2 edition, 2014. doi:10.1017/CBO9781139924801.

32 Daniel Lokshtanov, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. Fixed-Parameter Tractable Canonization and Isomorphism Test for Graphs of Bounded Treewidth. *SIAM Journal on Computing*, 46(1):161–189, 2017. doi:10.1137/140999980.

33 Eugene M. Luks. Isomorphism of graphs of bounded valence can be tested in polynomial time. *Journal of Computer and System Sciences*, 25:42–65, 1982.

34 Peter N. Malkin. Sherali-Adams relaxations of graph isomorphism polytopes. *Discrete Optimization*, 12:73–97, 2014.

35 David W. Matula. Subtree Isomorphism in $O(n^{5/2})$. In B. Alspach, P. Hell, and D.J. Miller, editors, *Algorithmic Aspects of Combinatorics*, volume 2 of *Annals of Discrete Mathematics*, pages 91–106. Elsevier, 1978. doi:10.1016/S0167-5060(08)70324-8.

36 Brendan D. McKay and Adolfo Piperno. Practical Graph Isomorphism, II. *J. Symb. Comput.*, 60:94–112, 2014. doi:10.1016/j.jsc.2013.09.003.

37 Christopher Morris, Kristian Kersting, and Petra Mutzel. Glocalized Weisfeiler-Lehman Graph Kernels: Global-Local Feature Maps of Graphs. In Vijay Raghavan, Srinivas Aluru, George Karypis, Lucio Miele, and Xindong Wu, editors, *IEEE International Conference on Data Mining (ICDM), 2017*, pages 327–336. IEEE Computer Society, 2017. doi:10.1109/ICDM.2017.42.

38 Christopher Morris and Petra Mutzel. Towards a practical k-dimensional Weisfeiler-Leman algorithm. Unpublished manuscript, 2019.

39 Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and Leman go neural: Higher-order graph neural networks. *CoRR*, abs/1810.02244, 2018. to appear at AAAI 2019. arXiv:1810.02244.

40 John M. Robson. Finding a maximum independent set in time $O(2^{n/4})$. Technical report, LaBRI, Université Bordeaux I, 2001.

41 Uwe Schöning. Graph isomorphism is in the low hierarchy. *Journal of Computer and System Sciences*, 37(3):312–323, 1988. doi:10.1016/0022-0000(88)90010-4.

42 Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014. doi:10.1017/CBO9781107298019.

43 Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. Weisfeiler-Lehman Graph Kernels. *Journal of Machine Learning Research*, 12:2539–2561, 2011.

44 Nino Shervashidze, SVN Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt. Efficient graphlet kernels for large graph comparison. In David van Dyk and Max Welling, editors, *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pages 488–495, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 2009. PMLR. URL: http://proceedings.mlr.press/v5/shervashidze09a.html.

45 Steven S. Skiena. *The Data Science Design Manual*. Springer, 2017. URL: https://www.springer.com/de/book/9783319554433.

46 Gottfried Tinhofer. A note on compact graphs. *Discrete Applied Mathematics*, 30(2):253–264, 1991.

47 University Consortium for Geographic Information Science. A UCGIS Call to Action: Bringing the Geospatial Perspective to Data Science Degrees and Curricula, 2018. URL: `https://www.ucgis.org/assets/docs/UCGIS-Statement-on-Data-Science-Summer2018.pdf`.

48 Mohammed J. Zaki and Meira Wagner Jr. *Data Mining and Analysis: Fundamental Concepts and Algorithms*. Cambridge University Press, New York, NY, USA, 2014.