Network Science **7** (2): 215–241, 2019
doi:10.1017/nws.2019.13

CAMBRIDGE UNIVERSITY PRESS

ORIGINAL ARTICLE

Assessing the computational complexity of multilayer subgraph detection†

Robert Bredereck[1], Christian Komusiewicz[2], Stefan Kratsch[3], Hendrik Molter[4,*], Rolf Niedermeier[4] and Manuel Sorge[5]

[1]Institut für Softwaretechnik und Theoretische Informatik, TU Berlin, Germany (e-mail: robert.bredereck@tu-berlin.de)
[2]Fachbereich Mathematik und Informatik, Philipps-Universität Marburg, Marburg, Germany (e-mail: komusiewicz@informatik.uni-marburg.de) [3]Humboldt-Universität zu Berlin, Berlin, Germany (e-mail: kratsch@informatik.hu-berlin.de)
[4]Institut für Softwaretechnik und Theoretische Informatik, TU Berlin, Germany (e-mails: h.molter@tu-berlin.de, rolf.niedermeier@tu-berlin.de) and [5]Faculty of Mathematics, Informatics and Mechanics, University of Warsaw, Poland (e-mail: manuel.sorge@mimuw.edu.pl)
*Corresponding author. Email: h.molter@tu-berlin.de

Action Editor: Ulrik Brandes

Abstract

Multilayer graphs consist of several graphs, called layers, where the vertex set of all layers is the same but each layer has an individual edge set. They are motivated by real-world problems where entities (vertices) are associated via multiple types of relationships (edges in different layers). We chart the border of computational (in)tractability for the class of subgraph detection problems on multilayer graphs, including fundamental problems such as maximum-cardinality matching, finding certain clique relaxations, or path problems. Mostly encountering hardness results, sometimes even for two or three layers, we can also spot some islands of computational tractability.

Keywords: parameterized computational complexity; exact algorithms; multi-modal data; matching; Hamiltonian path; community detection

1. Introduction

Multilayer graphs consist of several graphs, called *layers*, where the vertex set of all layers is the same but each layer has an individual edge set (Magnani & Rossi, 2011; Kivelä et al., 2014; Boccaletti et al., 2014). They are also known as multidimensional networks (Berlingerio et al., 2013), multiplex networks (Mucha et al., 2010), and edge-colored multigraphs (Cai & Ye 2014; Agrawal et al. 2018a, 2018b), among others (Kivelä et al., 2014). In recent years, multilayer graphs have gained considerable attention because observational data often come in a multimodal nature. Typical topics studied here include clustering (Boden et al., 2017; Dong et al., 2012, 2014; Jiang & Pei, 2009; Chen et al., 2018a), detection of network communities (Kim & Lee, 2015; Zeng et al., 2007), data privacy (Rossi et al., 2015), matching problems (Chen et al., 2018b), and general network properties (Berlingerio et al., 2013).

In several of these applications, the goal is to identify vertex subsets of a multilayer graph that exhibit a certain structure in each layer. For example, motivated by applications in genome

†An extended abstract (Bredereck et al., 2017) appeared in the proceedings of the 10th International Conference on Algorithms and Complexity (CIAC 2017), held in Athens, Greece, May 24–26, 2017. This long version now contains a reorganization and much broader motivation and interpretation of the results, as well as full proofs of all results. Work started while all authors were with TU Berlin.

© Cambridge University Press 2019

https://doi.org/10.1017/nws.2019.13 Published online by Cambridge University Press

comparison in computational biology, Gai et al. (2003) searched for maximal vertex subsets in a two-layer graph that induce a connected graph in each layer. Jiang & Pei (2009) and Boden et al. (2017) searched for vertex subsets that induce dense subgraphs in many layers. Such vertex subsets model communities in a multimodal social network.

To the best of our knowledge, however, a systematic study on computational complexity classification of such problems is lacking. Typically, authors observe the generalization of hardness results for the one-layer case to the multilayer one (Boden et al., 2017; Jiang & Pei, 2009) or perform case studies for special types of subgraphs (Agrawal et al. 2018a, 2018b). From a general algorithmic viewpoint, we are aware only of Cai & Ye (2014), who derived some complexity results for general classes of subgraphs, focusing mostly on the two-layer case. Our aim in this article is hence to build a general foundation for studying the worst-case computational complexity of a wide class of multilayer subgraph problems, and to provide results that may stimulate more specific algorithmic analyses.

We first give a general problem definition that encompasses the problems sketched above. All of these problems can be phrased as finding a large vertex subset that induces graphs with an interesting property in many layers. Motivated by the heterogeneity of the desired properties and analogous to the one-layer case (Garey & Johnson, 1979), we give a problem definition with a generic *graph property* Π as part of the problem name. Here, Π is formally a fixed set of graphs. For example, Π may be the set of all connected graphs or the set of all complete graphs. This allows us to study a wide variety of graph properties; we remark that throughout the paper we always assume that testing membership is decidable for the graph properties Π under consideration.

Π Multi-Layer Subgraph (Π-ML-Subgraph)
Input: A set of undirected graphs $G_1, \ldots, G_t$ all on the same vertex set V and two positive integers k and ℓ.
Question: Is there a vertex set $X \subseteq V$ with $|X| \geq k$ such that for at least ℓ of the input graphs G_i it holds that the induced subgraph $G_i[X]$ has the property Π?

We study Π-ML-Subgraph mostly in the context of parameterized computational complexity analysis (Cygan et al., 2015; Downey & Fellows, 2013; Flum & Grohe, 2006; Niedermeier, 2006). In contrast to classical computational complexity analysis, where mostly only the input size is used to estimate the worst-case running time of an algorithm, we consider running times depending also on secondary measures of the input instances, so-called *parameters*. We study Π-ML-Subgraph with respect to the following natural parameters:

- the total number t of layers,
- the number k of vertices to select, and
- the number ℓ of layers to select, as well as
- their dual parameters $|V| - k$ and $t - \ell$,

and combinations of these parameters. A combination of two parameters can be thought of as the parameter obtained by taking the sum of the two individual parameters. The parameters $|V| - k$ and $t - \ell$ are also called *deletion* parameters because they correspond to the equivalent problem of deleting at most $|V| - k$ vertices and $t - \ell$ layers to obtain a desired subgraph in each remaining layer.

In analyzing the parameterized computational complexity with respect to a parameter p, we aim to find efficient algorithms if p is small. This is formalized in the concept of *fixed-parameter tractability (FPT)* which states that there exists an algorithm that produces a solution in $f(p) \cdot \text{poly}(|I|)$ time, where f is a computable function and $|I|$ is the size of the input instance. In contrast, polynomial-time algorithms for a constant parameter value p may have running time $O(|I|^{g(p)})$ for some computable function g. Such algorithms, called *XP algorithms*, usually have a prohibitively large running time, even for small values of p. Using *W[1]-hardness*, a concept analogous to NP-hardness, one can show that a problem is unlikely to admit an FPT

https://doi.org/10.1017/nws.2019.13 Published online by Cambridge University Press

Table 1. Result Overview; k is the number of vertices to select, ℓ is the number of layers to select, and t is the total number of layers. A graph property Π is *vertex-partitionable* if one can compute partitions of a graph into maximal components that each satisfies Π in polynomial time; for details see Definition 1 in Section 4. A graph property is *staggered* if we can build a certain gadget based on the property; see Definition 2 in Section 4 for details. For all FPT, XP, and W[1]-hardness results, we also have corresponding NP-hardness results. (In the case of vertex-partitionable graph properties, we get NP-hardness if the property is also staggered, which is the case for all properties we consider in this paper.)

Graph Property Π	Complexity of Π-ML-Subgraph	Reference
Hereditary and		
- contains finitely many graphs	Polynomial-time solvable	Proposition 1
- includes all complete *and* all edgeless graphs	FPT w.r.t. (k, ℓ)	Proposition 1
- includes *either* all complete *or* all edgeless graphs	W[1]-hard w.r.t. k for all ℓ	Proposition 1
- characterizable by finitely many forbidden induced subgraphs	FPT w.r.t. $(\lvert V\rvert - k, t - \ell)$	Proposition 2
Vertex-partitionable	FPT w.r.t. t and XP w.r.t. ℓ	Proposition 3
Staggered	W[1]-hard w.r.t. (k, ℓ)	Theorem 1
Matching	Polynomial-time solvable for all $\ell \leq 2$ W[1]-hard w.r.t. k for all $\ell \geq 3$	Theorem 2
c-Factor	W[1]-hard w.r.t. k for all $\ell \geq 2$	Theorem 3
Hamiltonian	W[1]-hard w.r.t. k for all $\ell \geq 2$	Theorem 4

algorithm. To fully assess the computational complexity behavior of problems, it is important to study them under various parameterizations and combinations thereof (Downey & Fellows, 2013; Niedermeier, 2010; Komusiewicz & Niedermeier, 2012). Note that W[1]-hardness when parameterized by a combined parameter (p, q) implies W[1]-hardness when parameterized by either p or q. On the contrary, W[1]-hardness with respect to a parameter p may still allow for an FPT algorithm with respect to a combination of p with an additional parameter.

1.1 Our results

We give an overview of our results in Table 1. Observe that for Π-ML-Subgraph, NP-hardness and W[1]-hardness for either k or $|V| - k$ in the single-layer case directly imply hardness of the multi-layer case. Our analysis of Π-ML-Subgraph starts with several easy observations on *hereditary* graph properties Π, that is, Π is closed under vertex deletions. Such properties Π have been well-studied in the single-layer case. Using Ramsey arguments and a theorem due to Khot and Raman (2002), we show a trichotomy for the complexity of Π-ML-Subgraph with respect to the inclusion of edgeless or complete graphs in Π, distinguishing between polynomial-time solvability, fixed-parameter tractability with respect to the combined parameter (k, ℓ), and W[1]-hardness with respect to k for all ℓ (Proposition 1). Second, we generalize a result due to Cai (1996) by showing that, for graph properties Π characterized by a finite number of forbidden induced subgraphs, Π-ML-Subgraph is fixed-parameter tractable with respect to the combined parameter $(t - \ell, |V| - k)$, and that in this case it additionally admits a polynomial-size problem kernel (Proposition 2).

Next, we turn to graph properties that are not necessarily hereditary. An easy example would be connectedness. For finding connected graphs of order at least k in at least ℓ of t layers, there is a simple FPT algorithm with respect to t that is also an XP algorithm with respect to $t - \ell$ or with respect to ℓ. This algorithm admits a generalization to each graph property that implies certain good-natured partitions of the input graphs (Proposition 3), for example, so-called c-cores (Seidman, 1983) and c-trusses (Cohen, 2008). On the flip side, we spot W[1]-hardness

https://doi.org/10.1017/nws.2019.13 Published online by Cambridge University Press

for Π-ML-SUBGRAPH for the combined parameter k and ℓ for a specific large class of graph properties Π that we call *staggered* (Theorem 1). Such properties allow to efficiently construct graphs that have three parts that are obligatory, optional, and forbidden for the solution, respectively. Staggered graph properties Π include connected graphs, c-cores, and c-trusses, for example (see Corollary 5).

Finally, we exhibit three graph properties Π for which Theorem 1 and Corollary 5 already yield hardness, but where we can achieve stronger results by closer inspection. First, we consider the property that includes all graphs that admit a perfect matching. While finding a vertex subset of size k that induces subgraphs with a perfect matching in two layers is polynomial-time solvable, it becomes NP-hard and W[1]-hard with respect to the number of vertices to select k in three layers (Theorem 2). For a generalization of matchings, so-called c-factors, the subgraph detection problem already becomes NP-hard and W[1]-hard with respect to the number of vertices to select k in two layers for all $c \geq 2$ (Theorem 3). Furthermore, we consider the property that includes all graphs that admit a Hamiltonian path. While finding an order-k subgraph containing a Hamiltonian path is fixed-parameter tractable with respect to the number k of vertices to select in one layer (Monien, 1985), it becomes W[1]-hard in two layers (Theorem 4).

Apart from providing a broad overview over the complexity of Π-ML-SUBGRAPH, the main technical contributions are revealing conditions on Π that make Π-ML-SUBGRAPH hard (Theorem 1) and understanding the transition from tractability to hardness for perfectly matchable subgraphs (Theorem 2) and Hamiltonian subgraphs (Theorem 4).

1.2 Related work

As mentioned in the beginning, despite the numerous practical studies related to multilayer networks, systematic work pertaining to the computational complexity of Π-ML-SUBGRAPH is not well-developed. The following special cases were studied from this viewpoint. Gai et al. (2003) and Bui-Xuan et al. (2008) studied the case where the graph property Π is the set of connected graphs and $t = \ell = 2$; that is, they studied the problem of finding connected subgraphs of size at least k in two layers. They showed that the resulting problem is polynomial-time solvable. In contrast, Cai & Ye (2014) studied a modified version of this problem, where the desired vertex subset shall be of size *exactly* k instead of at least k.[1] They showed NP-hardness and W[1]-hardness with respect to the number k of vertices to select and with respect to $|V| - k$. Agrawal et al. (2018b) gave a $2^{3^{tk}} \cdot \text{poly}(n, t)$-time algorithm for the case where Π is the set of acyclic graphs and $t = \ell$.

In terms of general graph properties, Cai & Ye (2014) proved a trichotomy for hereditary graph properties similar to the one we give in Proposition 1 in the setting where the input consists only of two layers whose edge sets are disjoint and one wants to satisfy two possibly distinct graph properties in the corresponding layers (see their Theorem 6). They also showed that Π-ML-SUBGRAPH is fixed-parameter tractable parameterized by $|V| - k$ in the following modified setting: Each layer i has a specific graph property Π_i that is characterized by a finite set of forbidden induced subgraphs and the vertex sets of the layers may differ (see their Theorem 7). Our Proposition 2 is strongly related to this result and when focusing on the plain Π-ML-SUBGRAPH problem, it can be seen as a generalization of their result and additionally provides a polynomial-size problem kernel.

In mathematical terms, multilayer graphs are equivalent to edge-colored multigraphs. These have been studied from an algorithmic viewpoint; an overview can be found in the surveys of Bang-Jensen & Gutin (2009), Chapter 26, and Kano & Li (2008). Most of the algorithmic results presented there pertain to paths and cycles that do not contain two consecutive edges in the same layer and to related questions like connectedness and Hamiltonicity using this notion of paths or cycles.

If a multilayer graph is additionally equipped with a linear ordering of the layers, then the model is mathematically equivalent to *temporal graphs* (Holme & Saramäki, 2012; Holme, 2015;

https://doi.org/10.1017/nws.2019.13 Published online by Cambridge University Press

Michail, 2016; Latapy et al., 2018). In this model, each layer models the state of the data set at a different point in time. There is a huge body of research in this area and we refer to the aforementioned surveys for an overview.

1.3 Organization

Section 2 contains some basic definitions and notation. In Section 3, we give general results for hereditary graph properties Π. In Section 4, we give general results for large classes of graph properties Π that are not necessarily hereditary. In the two succeeding sections, we take a closer look at selected graph properties where a more in-depth inspection reveals stronger results compared to the general result of Section 4. In Section 5, we investigate the parameterized computational complexity of Matching-ML-SUBGRAPH and the generalization c-Factor-ML-SUBGRAPH. In Section 6, we analyze the parameterized computational complexity of Hamiltonian-ML-SUBGRAPH. We give a conclusion and directions for future work in Section 7.

2. Preliminaries

2.1 Parameterized complexity

A *parameterized problem* is a language $L \subseteq \Sigma^* \times \mathbb{N}$, where the second component in an instance $(I, k) \in \Sigma^* \times \mathbb{N}$ is called the *parameter*. In the case of combined parameters, we write a tuple, for example, (k_1, k_2). This is just notation for a parameter $k = k_1 + k_2$. A parameterized problem L is *fixed-parameter tractable* if there is an algorithm that for each instance $(I, k) \in \Sigma^* \times \mathbb{N}$ decides whether $(I, k) \in L$ in $f(k) \cdot |I|^{O(1)}$ time, where f is a computable function and $|I|$ is the encoding length of the input I. We also say that $|I|$ is the *instance size*. The class of fixed-parameter tractable problems is FPT. A parameterized problem L is in the class XP if there is an algorithm that decides for each instance $(I, k) \in \Sigma^* \times \mathbb{N}$ whether $(I, k) \in L$ in $|I|^{f(k)}$ time, where f is a computable function. W[1]-hard parameterized problems are generally assumed not to be fixed-parameter tractable. W[1]-hardness can be shown by a parameterized reduction from another W[1]-hard problem such as INDEPENDENT SET (given a graph G and an integer k, decide whether there is a k-vertex subset in G that does not contain any edge). A *parameterized reduction* from a parameterized problem Q to a parameterized problem L is an algorithm that maps an instance (I, k) of Q to an instance (I', k') of Q in $f(k) \cdot |I|^{O(1)}$ time such that $(I, k) \in Q$ if and only $(I', k') \in L$ and $k' \leq g(k)$, where f and g are arbitrary computable functions.

Furthermore, parameters allow us to mathematically rigorously study efficient data reduction. Formally, given a parameterized problem L, a *kernelization algorithm* is a polynomial-time algorithm that maps instances (I, k) of L to instances (I', k') (called a *problem kernel*) of L such that the size of I' is upper-bounded by a function of the parameter p and (I', k') is a yes-instance if and only if (I, k) is a yes-instance. If the kernel size can be upper-bounded by a polynomial in the parameter, we call it a *polynomial kernel*. For more context and methodology, we refer to the literature (Cygan et al., 2015; Downey & Fellows, 2013; Flum & Grohe, 2006; Niedermeier, 2006).

2.2 Graphs

All graphs in this work are undirected and without self loops or parallel edges. We use standard graph notation (Diestel, 2017). A graph property Π is *hereditary* if removing any vertex from a graph in Π results again in a graph in Π. We consider the following graph properties. A graph is a *c-core* if each vertex has degree at least c (Seidman, 1983). A graph is a *c-truss* if it is connected and each edge is contained in at least $c - 2$ triangles (Cohen, 2008). We say that a graph is *Hamiltonian* if it contains a simple path that comprises all vertices in the graph. The *length of a path* is the

https://doi.org/10.1017/nws.2019.13 Published online by Cambridge University Press

number of its edges. A *c-factor* in a graph is a subset of the edges such that each vertex is incident with exactly c edges. In sans serif font face, we often denote graph properties. For example, c-Truss is the set of all c-trusses. By Matching, we refer to the set of all graphs containing a perfect matching; note that then finding a maximum matching is equivalent to finding a maximum sized Matching-subgraph. By c-Factor, we refer to the set of all graphs containing a c-factor. For a list of definitions of all graph properties mentioned in this article, see Appendix.

3. Hereditary graph properties

In this section, we study the (parameterized) computational complexity of Π-ML-SUBGRAPH with respect to hereditary graph properties Π. Many natural graph properties fall into this category, for example, being planar or being a forest. We give a trichotomy of the complexity, classifying each problem either as polynomial-time solvable or as NP-hard, and further classifying the parameterized complexity of the NP-hard cases with respect to the parameters number k of vertices to select and number ℓ of layers to select. In addition, we observe fixed-parameter tractability for the deletion parameters $|V| - k$ and $t - \ell$.

The single-layer case has been studied by Lewis & Yannakakis (1980) as well as Khot & Raman (2002); the latter studied the parameterized complexity of the subgraph detection problem for hereditary properties. Cai & Ye (2014) studied this problem on two layers with disjoint edge sets (see their Theorem 6) and for multiple layers when layers cannot be deleted (in their Theorem 7). We generalize in the following the mentioned results to the multilayer case where layers can be deleted. This allows us to classify all hereditary graph properties Π by the parameterized complexity of the corresponding Π-ML-SUBGRAPH problem.

Proposition 1 (Complete classification of hereditary graph properties)**.** *If* Π *is a hereditary graph property, then the following statements are true:*

(1) If Π *excludes at least one complete graph and at least one edgeless graph, then* Π-ML-SUBGRAPH *is solvable in polynomial time.*
(2) If Π *includes all complete graphs and all edgeless graphs, then* Π-ML-SUBGRAPH *is NP-hard and FPT when parameterized by the combined parameter number k of vertices to select and number ℓ of layers to select.*
(3) If Π *includes either all complete graphs or all edgeless graphs (but not both), then* Π-ML-SUBGRAPH *is NP-hard and W[1]-hard when parameterized by the number k of vertices to select for all numbers ℓ of layers to select.*

Proof. We utilize the concept of *Ramsey numbers*. The Ramsey number $R(p, q)$ is the minimum number x such that every graph with x vertices has either a clique of size p or an independent set of size q. It is well known that $R(p, q) \leq \binom{p+q-2}{q-1}$ (Jukna, 2011). We give separate proofs for all three statements in the theorem. Note that for the second and third statements, NP-hardness even in the single-layer case was shown by Lewis & Yannakakis (1980).

Statement 1. Let p, q be the sizes of the smallest excluded complete and edgeless graph, respectively. Note that any graph on at least $R(p, q)$ vertices contains either a clique of size p or an independent set of size q and hence is not included in Π. Hence, there are only finitely many graphs that have property Π. Furthermore, if $k \geq R(p, q)$, then we face a no-instance.

If $k < R(p, q)$, then we consider every order-k vertex subset X and check whether $G[X] \in \Pi$ in at least ℓ layers. If this is the case for some X, then we output X; otherwise, we output that the instance is a no-instance. The running time for this algorithm is $O(t\binom{n}{k}f(k))$, where $f(k)$ is the time to check membership of Π. Note that k is constant since $k < R(p, q)$ and p and q are constants only depending on Π. Hence, the overall running time is polynomial.

https://doi.org/10.1017/nws.2019.13 Published online by Cambridge University Press

Statement 2. The NP-hardness follows from the NP-hardness of the single-layer case. For the proof of fixed-parameter tractability, we introduce nested Ramsey numbers as follows:

$$R^{(1)}(p,q) = R(p,q)$$
$$R^{(i)}(p,q) = R(R^{(i-1)}(p,q), R^{(i-1)}(p,q))$$

We show that if $|V| \geq R^{(\ell)}(k,k)$, then we face a yes-instance. Indeed, we show the more general statement that, for each set of ℓ layers G_i, $i = 1, \ldots, \ell$, on vertex set V with $|V| \geq R^{(\ell)}(k,k)$, there is a vertex subset $X \subseteq V$ with $X \geq k$ such that $G_i[X] \in \Pi$ for each $i \in 1, \ldots, \ell$. We prove this by induction on ℓ.

For $\ell = 1$, we have $|V| \geq R(k,k)$. Hence, each graph on vertex set V contains either a clique of size k or an independent set of size k, proving the statement. Assume that $\ell > 1$ and that the statement holds for each $\ell' < \ell$. Since $|V| \geq R^{(\ell)}(k,k)$, each layer has either a clique of size $R^{(\ell-1)}(k,k)$ or an independent set of size $R^{(\ell-1)}(k,k)$. Let $X' \subseteq V$ with $|X'| = R^{(\ell-1)}(k,k)$ be either a clique or an independent set in layer 1. By the induction hypothesis, there is a vertex set $X \subseteq X'$ with $|X| \geq k$ such that $G_i[X] \in \Pi$ for all i such that $2 \leq i \leq \ell$. Since $X \subseteq X'$, we also have $G_1[X] \in \Pi$. Hence, $G_i[X] \in \Pi$ on each of the ℓ layers G_i, as required.

The algorithm is now as follows. If $|V| \geq R^{(\ell)}(k,k)$, then accept immediately. By the above, each subset of ℓ layers of the input multilayer graph contains a solution. Otherwise, if $|V| < R^{(\ell)}(k,k)$, then find a solution by brute force, if it exists: Simply try all possible vertex subsets X of size k and check whether $G_i[X] \in \Pi$ for at least ℓ layers i. By heredity, if there is a solution, then there is one of size k and thus the algorithm is correct. If $g(k)$ denotes the time needed to check whether $G_i[X] \in \Pi$ for some fixed i, then the running time for the brute-force step is at most $|V|^{k+O(1)} \cdot t \cdot g(k) \leq (R^{(\ell)}(k,k))^{k+O(1)} \cdot g(k) \cdot t = f(k,\ell) \cdot t$ for some function f, showing that the problem is FPT with respect to k and ℓ combined.

Statement 3. Khot & Raman (2002) showed that for hereditary properties Π including either all complete graphs or all edgeless graphs, Π-SUBGRAPH is W[1]-hard when parameterized by k. This directly translates to the multilayer case, as does NP-hardness. □

Note that every hereditary graph property falls into one of the three cases of Proposition 1. Properties that fall into the first case are exactly those containing only a finite number of graphs. In the following corollary, we give a number of hereditary properties Π that fall in the second and third cases and give the corresponding complexity results for Π-ML-SUBGRAPH implied by Proposition 1. For their definitions, we refer to the literature (Brandstädt et al., 1999; Golumbic, 2004) or to Appendix: Definitions.

Corollary 1.

(1) Π-ML-SUBGRAPH *is NP-hard and FPT when parameterized by the combined parameter number k of vertices to select and number ℓ of layers to select for* $\Pi \in$ {*Asteroidal Triple Free Graph, Chordal Graph, Comparability Graph, Interval Graph, Perfect Graph, Permutation Graph, Split Graph*}.

(2) Π-ML-SUBGRAPH *is NP-hard and W[1]-hard when parameterized by the number k of vertices to select for all numbers ℓ of layers to select for* $\Pi \in$ {*c-Colorable Graph, Complete Graph, Complete Multipartite Graph, Edgeless Graph, Forest, Planar Graph*}.

Next, we consider properties Π whose complements are hereditary or, equivalently, Π is closed under the operation of adding a new vertex v and connecting v arbitrarily to the rest of the graph. For these, we can observe that polynomial-time solvability transfers to the multilayer case.

https://doi.org/10.1017/nws.2019.13 Published online by Cambridge University Press

Observation 1. *Let* Π *be a graph property such that whenever* $G \in \Pi$ *for some graph* G*, then we have that for all graphs* $H = (V, E)$ *if there is a vertex set* $X \subseteq V$ *such that* $H[X]$ *is isomorphic to* G*, then* $H \in \Pi$*. If* Π *can be decided in* $T(|V|)$ *time for some function* T*, then* Π-ML-SUBGRAPH *can be decided in* $O(t \cdot T(|V|))$ *time for all numbers* k *of vertices to select and all numbers* ℓ *of layers to select.*

Proof. Let Π be a graph property such that if $G \in \Pi$ for some graph G and $H[X] = G$ for some graph H and vertex set X, then $H \in \Pi$. Observe that if $G \notin \Pi$, then no induced subgraph of G can be in Π. Let $(G_1, \ldots, G_t, k, \ell)$ be an instance of Π-ML-SUBGRAPH. We decide for each graph $G_1, \ldots, G_t$ whether it satisfies property Π. We face a yes-instance if and only if there are at least ℓ graphs that have property Π: We can set $X = V$, and hence $|X| \geq k$, for any $k \leq n$. □

In the following corollary, we give two examples of properties Π for which by Observation 1 Π-ML-SUBGRAPH is solvable in polynomial time.

Corollary 2. Π-ML-SUBGRAPH *is solvable in polynomial time for*

- $\Pi =$ *"The graph has maximum degree at least x."*
- $\Pi =$ *"The graph contains a triangle."*
- $\Pi =$ *"The graph has an h-index*[2] *of at least x."*

Finally, we consider the dual parameterizations for hereditary graph properties characterized by a finite number of forbidden subgraphs. In the single-layer case, this problem has been studied by Lewis & Yannakakis (1980) as well as Cai (1996). A result of Cai & Ye (2014) (Theorem 7) implies a fixed-parameter algorithm for Π-ML-SUBGRAPH if $t - \ell = 0$; that is, no layers can be deleted. In the following, we give a more general fixed-parameter algorithm for the case where $t - \ell \geq 0$ and we furthermore present a polynomial kernel.

Proposition 2. *Let* Π *be a hereditary graph property that is characterized by finitely many forbidden induced subgraphs. Then* Π-ML-SUBGRAPH *is NP-hard and FPT when parameterized by the combined parameter number* $t - \ell$ *of layers to delete and number* $|V| - k$ *of vertices to delete. It also admits a polynomial-size problem kernel with respect to this combined parameter.*

Proof. To see the fixed-parameter tractability, consider the search-tree algorithm that recursively searches for a forbidden induced subgraph G' in one of the layers, and branches, for each vertex v in G', into the branch of deleting v and, additionally, into the branch of deleting the layer of G'. Hence, in each branch we either delete a layer or a vertex, so the depth of the search-tree is upper-bounded by $(t - \ell) + (|V| - k)$. Finding G' takes polynomial time because there is only a constant number of different forbidden subgraphs and each one has constant size. Furthermore, each node in the resulting search tree has a constant number of children. Hence, the search-tree algorithm has a running time of $c^{t-\ell+|V|-k} \cdot \text{poly}(I)$, where c is a constant, and I is the instance size, as required. This procedure is correct since each forbidden subgraph is either destroyed by deleting one of its vertices or by deleting the layer it exists in. Hence each of the remaining layers does not contain a forbidden subgraph and therefore has property Π.

To see that Π-ML-SUBGRAPH admits a polynomial kernel with respect to the combined parameter $(t - \ell, |V| - k)$, we use a reduction to 2-COLOR HITTING SET, a variant of HITTING SET, and then apply a (basically folklore) kernelization for 2-COLOR HITTING SET. Herein, we are given two disjoint ground sets B and W, a family $\mathcal{F}$ of subsets of $B \cup W$, and two integers b, w. We are to decide whether there is a *hitting set* $S \subseteq B \cup W$; that is, each subset $F \in \mathcal{F}$ has $F \cap S \neq \emptyset$, containing b elements from B and w elements from W. Clearly, 2-COLOR HITTING SET is contained in NP.

https://doi.org/10.1017/nws.2019.13 Published online by Cambridge University Press

The reduction works as follows. Given an instance of Π-ML-SUBGRAPH, we put the ground set $B := V$ and put a distinct new vertex into W for each layer. For each layer, we enumerate all forbidden induced subgraphs. This takes polynomial time, as the maximum size of these subgraphs is a constant. To define $\mathcal{F}$, for each forbidden induced subgraph G' we add its vertex set V' plus the vertex $v \in W$ corresponding to the layer in which G' is contained as a set $V' \cup \{v\}$ to $\mathcal{F}$. Integer b is set to $|V| - k$ and integer w to $t - \ell$. As mentioned, the reduction works in polynomial time. Since we have to "hit" each forbidden induced subgraph by either deleting a vertex from it, or deleting its layer completely, it is not hard to verify that the reduction is correct.

We now apply the so-called sunflower kernelization procedure (Moser, 2009; Kratsch, 2012; van Bevern, 2014) to the resulting 2-COLOR HITTING SET instance. A *sunflower* in $\mathcal{F}$ is a subfamily $\mathcal{F}' \subseteq \mathcal{F}$ such that there is a set $C \subseteq B \cup W$ with the property that each pair $F, F' \in \mathcal{F}'$ has $F \cap F' = C$. The *size* of a sunflower is $|\mathcal{F}'|$. If there is a sunflower of size $b + w + 1$ in $\mathcal{F}$, then every hitting set contains at least one element of C. Hence, we can safely remove one set out of every sunflower of size at least $b + w + 2$. This can be done exhaustively in polynomial time (Moser, 2009; Kratsch, 2012; van Bevern, 2014). After this procedure has been carried out, Erdős and Rado's Sunflower Lemma (Erdös & Rado, 1960) guarantees that the remaining set family $\mathcal{F}$ has size $O((b + w)^c)$, where c is the size of the largest set in $\mathcal{F}$. This is a polynomial because the sets in $\mathcal{F}$ have constant size. By removing elements of $B \cup W$ that are not contained in any set in $\mathcal{F}$, we obtain an overall size bound on the resulting instance of 2-COLOR HITTING SET that is polynomial in $b = |V| - k$ and $w = t - \ell$.

Finally, we transfer the instance of 2-COLOR HITTING SET created in this way to an equivalent instance of Π-ML-SUBGRAPH by using a polynomial-time many-one reduction. Such a reduction exists because 2-COLOR HITTING SET is in NP and Π-ML-SUBGRAPH is NP-hard for every graph property Π that is characterizable by a finite number of forbidden induced subgraphs (Lewis & Yannakakis, 1980). □

In the following corollary, we give several hereditary properties Π that are characterized by a finite number of forbidden subgraphs. For these, Π-ML-SUBGRAPH is fixed-parameter tractable with respect to the number $t - \ell$ of layers to delete and the number $|V| - k$ of vertices to delete combined. For their definitions, see Appendix: Definitions.

Corollary 3. *Let* $\Pi \in \{$*Cluster Graph, Cograph, Line Graph, Quasi-Threshold Graph, Split Graph*$\}$. *Then,* Π-ML-SUBGRAPH *is NP-hard. When parameterized by the combined parameter number* $t - \ell$ *of layers to delete and number* $|V| - k$ *of vertices to delete, then* Π-ML-SUBGRAPH *is FPT and admits a polynomial kernel.*

While this collection of graph properties might seem motivated only from a graph-theoretic point of view, we note that quasi-threshold graphs are a model for familial groups in social networks (Nastos & Gao, 2013) and split graphs are a model of communities with core/periphery structures (Borgatti & Everett, 2000).

4. Nonhereditary graph properties

In contrast to the previous section, we now give two results related to graph properties that are not necessarily hereditary. Two examples of such nonhereditary properties that are prominent in network analysis are Connectivity and c-Core (recall that graph is a c-core if each vertex has degree at least c). First, we consider graph properties in which each graph admits a certain nice vertex partitioning. This family of graph properties includes Connectivity and c-Core, among others. We give an FPT algorithm for the parameter total number t of layers; this algorithm is also an XP algorithm with respect to the number ℓ of subgraph layers. Second, we provide a general sufficient condition for graph properties Π for which Π-ML-SUBGRAPH is W[1]-hard for

https://doi.org/10.1017/nws.2019.13 Published online by Cambridge University Press

the combined parameter (k, ℓ). This general sufficient condition captures many prominent graph properties such as Connectivity, c-Core, c-Truss, and Matching.

4.1 Vertex-partitionable graph properties

We start with investigating graph properties Π that allow for efficiently computable partitions of a graph into maximal components that each satisfies Π. For such properties Π it turns out that finding large Π-subgraphs in all layers of the input graph is tractable. This can be seen as a generalization of the component-detection algorithm in two layers by Gai et al. (2003).

First, we define the type of partition that the graph property shall allow.

Definition 1. *Let Π be a graph property and let $G = (V, E)$ be a graph. A partition $\mathcal{P} := \{X_1, \ldots, X_x\}$ of V is a* Π-partition *if*

(i) $G[X_i] \in \Pi$ for all $X_i \in \mathcal{P}$,

(ii) for all $X \subseteq V$ such that $G[X] \in \Pi$, we have $X \subseteq X_i$ for some $X_i \in \mathcal{P}$.

Informally, the existence of a Π-partition means that there are maximal components containing all induced subgraphs that fulfill the property Π. In the case of Connectivity, these are exactly the connected components of the graph. We now show that if Π-partitions can be computed efficiently, then we can solve Π-ML-Subgraph efficiently if t, ℓ, or $t - \ell$ are small.

Proposition 3. *Let Π be a graph property such that every graph $G = (V, E)$ has a Π-partition that can be computed in $T(|V|, |E|)$ time where T is non-decreasing in both arguments. Then,* Π-ML-Subgraph *can be solved in $\binom{t}{\ell} \cdot O(|V| \cdot t) \cdot \max_{1 \leq i \leq t} (|E_i| + T(|V|, |E_i|))$ time.*

Proof. We describe an algorithm that works when $\ell = t$, that is, when we aim for satisfying property Π in all layers. To apply this algorithm for all other cases, we iterate through all possibilities to select ℓ layers where property Π shall be satisfied and then apply the algorithm for this selection. This gives an additional factor of $\binom{t}{\ell}$ to the overall running time. In each application, our algorithm outputs all maximal sets $X \subseteq V$ such that $G_i[X] \in \Pi$ for all selected input graphs G_i. We refer to these sets as *solutions* in the following. The algorithm maintains a partition $\mathcal{P}$ of V where, initially, $\mathcal{P} = \{V\}$.

The algorithm checks whether there is a $Y \in \mathcal{P}$ such that $G_i[Y] \notin \Pi$ for some input graph G_i. If there is such a vertex set $Y \in \mathcal{P}$, then it computes in $T(|V|, |E_i|)$ time a Π-partition $\mathcal{P}_Y$ of $G_i[Y]$. The partition $\mathcal{P}$ is replaced by $(\mathcal{P} \setminus \{Y\}) \cup \mathcal{P}_Y$. If there is no such vertex set, then the algorithm outputs all $Y \in \mathcal{P}$. It accepts if one of the outputs has size at least k and rejects otherwise.

To see the correctness of the algorithm, first observe that, for each output Y, we have that $G_i[Y] \in \Pi$ for all selected input graphs G_i. To show maximality of each Y, we show that the algorithm maintains the invariant that each solution X is a subset of some $Y \in \mathcal{P}$. This invariant is trivially fulfilled for the initial partition $\{V\}$. Now consider a set Y that is further partitioned by the algorithm. By the invariant, any solution X that has non-empty intersection with Y is a subset of Y. Since $\mathcal{P}_Y$ is a Π-partition of $G_i[Y]$, by Property (ii) of Π-partitions, there is no solution X that contains vertices of two distinct sets Y_1, Y_2 of $\mathcal{P}_Y$. Thus, each solution that is a subset of Y is also a subset of some $Y' \in \mathcal{P}_Y$. Hence, each output set X is a solution as it is an element of the final partition $\mathcal{P}$ and all solutions are subsets of elements of $\mathcal{P}$.

To upper-bound the running time, observe that for each $Y \in \mathcal{P}$, we can test in $O(t \cdot \max_{1 \leq i \leq t} T(|V|, |E_i|))$ time whether it needs to be partitioned further. At most $|V|$ partitioning steps are performed and if a set $Y \in \mathcal{P}$ does not need to be partitioned further, then it can be discarded for the remainder of the algorithm. Thus, in $O(|V|)$ applications of the "maximality test" the result is that Y is a solution and in $O(|V|)$ applications of the maximality test, Y is further

https://doi.org/10.1017/nws.2019.13 Published online by Cambridge University Press

partitioned. Hence, the overall number of sets Y that are elements of $\mathcal{P}$ at some point is $O(|V|)$. The overall running time now follows from the assumptions on T and from the fact that the induced subgraphs for all G_i can be computed in $O(t \cdot \max_{1 \leq i \leq t} |E_i|)$ time for each Y. □

Examples of graph properties covered by Proposition 3 are Connectivity and c-Edge-Connectivity. If we assume that graphs on one vertex are considered as (trivial) c-cores, then the c-Core property is also covered: The nontrivial c-core of a graph is uniquely determined (it is the subgraph remaining after deleting any vertex with degree less than c). Similarly, the c-Truss property is covered by Proposition 3 if we allow one-vertex graphs to be considered as c-trusses. Observe that we can also require the c-cores and c-trusses to be connected. For definitions of the graph properties mentioned above and in the following corollary, see Appendix: Definitions.

If T is a polynomial function, which holds for all examples described above, then Π-ML-Subgraph is fixed-parameter tractable with respect to t and polynomial time solvable if ℓ or $t - \ell$ are constants.

Corollary 4. *Let $\Pi \in \{$c-Core, c-Edge-Connectivity, Connectivity, c-Truss$\}$. Π-ML-Subgraph is FPT when parameterized by the total number t of layers and polynomial-time solvable if the number ℓ of layers to select or the number $t - \ell$ of layers to delete is constant.*

4.2 A general hardness reduction

We now describe a large class of graph properties Π for which Π-ML-Subgraph is NP-hard and W[1]-hard when parameterized by the combined parameter number k of vertices to select and number ℓ of layers to select. We call these graph properties *staggered* (Definition 2). The definition is somewhat technical but covers many natural graph properties Π that are not hereditary, such as Connectivity. In the main theorem of this paragraph (Theorem 1), we show that for staggered graph properties Π, Π-ML-Subgraph is NP-hard and W[1]-hard when parameterized by the combined parameter number k of vertices to select and number ℓ of layers to select. Furthermore, since all graph properties from Corollary 4 are staggered, it shows that for those properties, Π-ML-Subgraph becomes intractable when parameterized by ℓ instead of t. We list some graph properties Π for which Theorem 1 can be applied in Corollary 5 and explain how the theorem yields the results in said cases.

Intuitively, Theorem 1 covers graph properties Π for which it is possible to construct graphs G with the following properties:

- The vertices are partitionable in three sets: a fixed number of *obligatory* vertices, a variable number of *optional* vertices, and a variable number of *forbidden* vertices.
- A sufficiently large subgraph of G has property Π if and only if it contains all obligatory vertices and no forbidden vertices.

Formally, we characterize these graph properties Π as follows.

Definition 2. *Let Π be a graph property and f, g:$\mathbb{N} \to \mathbb{N}$ two polynomial-time computable functions. We say that Π is* (f, g)-staggered *if there is an algorithm A_Π and a constant c_Π depending only on Π such that A_Π takes as input a set W, a subset $W' \subseteq W$, and an integer $\alpha \geq c_\Pi$, runs in $(|W| + \alpha)^{O(1)}$ time, and computes a graph $G = (V, E)$ fulfilling the following conditions:*

(1) For each $w \in W$, there is a vertex set $X_w \subseteq V$ with $|X_w| = f(\alpha)$;
(2) $\{X_w \mid w \in W\} \cup \{Y\}$ is a partition of V for some Y with $|Y| = g(\alpha)$;
(3) for all $X \subseteq V$ with $|X| \geq \alpha \cdot f(\alpha) + g(\alpha)$, we have that

$$G[X] \in \Pi \Leftrightarrow \exists W'' \subseteq W' \text{ such that } X = \bigcup_{w \in W''} X_w \cup Y$$

The output of Algorithm A_Π is the graph $G = (V, E)$ as well as the partition of its vertex set $\{X_w \mid w \in W\} \cup \{Y\}$.

https://doi.org/10.1017/nws.2019.13 Published online by Cambridge University Press

The intuition is that each set X_w corresponds to one vertex $w \in W$ and every sufficiently large set X such that $G[X] \in \Pi$ either fully contains X_w or not. Furthermore, Y is the set of obligatory vertices that have to be included in X in order to have that $G[X] \in \Pi$. Finally, the sets X_w that correspond to vertices $w \in W \setminus W'$ are forbidden; that is, they have to be fully excluded from X in order to have that $G[X] \in \Pi$. For the proof, we reduce from BICLIQUE, which is NP-hard (Johnson, 1987) and W[1]-hard when parameterized by the size h of the biclique (Lin, 2015).

Theorem 1. *Let Π be an (f, g)-staggered graph property. Then, Π-ML-SUBGRAPH is NP-hard and W[1]-hard when parameterized by the combined parameter number k of vertices to select and number ℓ of layers to select.*

Proof. We give a parameterized reduction from BICLIQUE that, given an undirected graph H and a positive integer h, asks whether H contains a $2h$-vertex *biclique*, a complete bipartite subgraph $K_{h,h}$ in which both partite sets have size h.

Reduction idea. The main idea is to construct a layer for each vertex v of the input graph using Algorithm A_Π from Definition 2 such that the optional vertices correspond to the open neighborhood of v in the input graph. Then we can show that selecting layers corresponds to selecting one half of the biclique and the optional vertices that are part of the selected subgraph form the second half of the biclique.

Let $(H = (U, F), h)$ be an instance of BICLIQUE and assume $h \geq 2$ without loss of generality. Let A_Π, f, g be the algorithm and functions promised by the definition of being (f, g)-staggered. We construct an instance of Π-ML-SUBGRAPH in the following way.

For all $v \in U$, let $N_H(v)$ be the neighborhood of v with respect to H. Run Algorithm A_Π (see Definition 2) on input $(U, N_H(v), h)$ to create graphs G_v for each $v \in U$. A graph G_v created that way has a vertex set $V_v = \bigcup_{u \in U} X_u \cup Y$. For all created graphs, we identify the vertices of the sets X_u in an arbitrary but fixed fashion, the same for vertices in Y. This allows us to say that all graphs G_v are defined over the same vertex set. Set $k := h \cdot f(h) + g(h)$ and $\ell := h$. Note that the parameters k and ℓ only depend on the solution size h of BICLIQUE. By Definition 2, this procedure runs in polynomial time. Now we show that $(\{G_v \mid v \in U\}, k, \ell)$ is a yes-instance of Π-ML-SUBGRAPH if and only if (H, h) is a yes-instance of BICLIQUE.

$(\Rightarrow)$: Assume that (H, h) is a yes-instance of BICLIQUE and let (C, D) with $C, D \subseteq U$ and $|C| = |D| = h$ represent a biclique. Then we set $X := \bigcup_{v \in C} X_v \cup Y$, where Y and the X_v are the vertex sets promised by Conditions 1 and 2 of Definition 2. Note that $|X| = h \cdot f(h) + g(h) = k$. Furthermore, for all $v' \in D$ and all $v \in C$ such that $X_v \subset X$, it holds that $v \in N_H(v')$. Hence, by Condition 3 of Definition 2, $G_{v'}[X] \in \Pi$. It follows that the number of layers G_v with $G_v[X] \in \Pi$ is at least $h = \ell$. Consequently, (X, D) is a solution of Π-ML-SUBGRAPH.

$(\Leftarrow)$: Assume that $(\{G_v \mid v \in U\}, k, \ell)$ is a yes-instance of Π-ML-SUBGRAPH. Then we know that there are graphs G_i, with $i \in L$, $L \subseteq U$, $|L| \geq h$, and a vertex set $X \subseteq V$ with $|X| \geq k$, such that $G_i[X] \in \Pi$ for all $i \in L$. By the construction of G_i (Conditions 1, 2, and 3 of Definition 2), we know that $X = \bigcup_{v \in W'} X_v \cup Y$ for some $W' \subseteq U$ with $|W'| \geq h$. Furthermore, we know that if $i \in L$ then for all $j \in W'$ (that is $X_j \subset X$) we have that i is a neighbor of j. Lastly, $i \in L$ implies that $X_i \not\subset X$ and hence $i \notin W'$. Hence, we have that (L, W') is a biclique in H with $|L| \geq h$ and $|W'| \geq h$. □

In the following corollary, we present several graph properties Π for which the single-layer case Π-SUBGRAPH is polynomial-time solvable but by application of Theorem 1, Π-ML-SUBGRAPH is NP-hard and W[1]-hard when parameterized by the combined parameter number k of vertices to select and number ℓ of subgraph layers. The proof of Corollary 5 simply consists of the description of Algorithm A_Π of Theorem 1 for those graph properties. For the definitions of the graph properties appearing in Corollary 5, see Appendix: Definitions.

https://doi.org/10.1017/nws.2019.13 Published online by Cambridge University Press

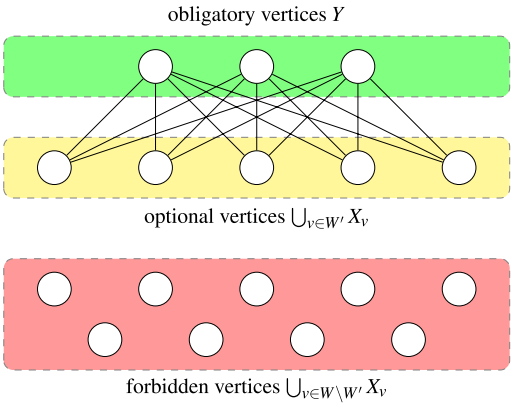


Figure 1. Visualization of the graph constructed by Algorithm A_Π for $\Pi \in \{c\text{-Core}, c\text{-Connectivity}\}$ with $c = 3$.

Corollary 5. Π-ML-SUBGRAPH *is NP-hard and W[1]-hard when parameterized by the combined parameter number k of vertices to select and number ℓ of layers to select for* $\Pi \in \{c\text{-Connectivity}, c\text{-Core}, c\text{-Factor}, \text{Connectivity}, c\text{-Truss}, \text{Hamiltonian}, \text{Matching}, \text{Star}, \text{Tree}\}$.

Proof. For each of the listed properties Π, we show that Π is (f, g)-staggered for some functions f and g. To this end, we describe the polynomial-time algorithm A_Π with inputs W, $W' \subseteq W$, and α. Theorem 1 then yields NP-hardness and W[1]-hardness of Π-ML-SUBGRAPH when parameterized by the combined parameter number k of vertices to select and number ℓ of subgraph layers. Figure 1 shows the graph constructed by Algorithm A_Π for $\Pi \in \{c\text{-Core}, c\text{-Connectivity}\}$ with $c = 3$.

Algorithm A_Π for $\Pi \in \{$Connectivity, Tree, Star, 1-Core$\}$:

We construct the graph $G = (V, E)$ as follows. Let $X_w := \{w\}$ for all $w \in W$ and $Y := \{u\}$ and hence $V = W \cup \{u\}$. Add an edge $\{u, w\}$ to the edge set E for each vertex $w \in W'$. This clearly fulfills Conditions 1 and 2 from Definition 2 for $f(\alpha) = g(\alpha) = 1$. Note that the graph G contains a star with u as center and all vertices $w \in W'$ as leaves and all other vertices are isolated. Furthermore, it is easy to see that no graph that satisfies one of the properties Connectivity, Tree, Star, and 1-Core can contain an isolated vertex. Hence, each subgraph of G that satisfies one of those properties is a subtree of the star in G and therefore has to contain the center u. It follows that Condition 3 from Definition 2 is fulfilled.

Algorithm A_Π for $\Pi \in \{c$-Core, c-Connectivity$\}$ with $c > 1$:

We construct the graph $G = (V, E)$ as follows. Let $X_w := \{w\}$ for all $w \in W$ and $Y := \{u_1, \ldots, u_c\}$ and hence $V = W \cup \{u_1, \ldots, u_c\}$. Add edges $\{u, w\}$ to E for all vertices $u \in Y$ and $w \in W'$. This clearly fulfills Conditions 1 and 2 from Definition 2 for $f(\alpha) = 1$ and $g(\alpha) = c$. Note that the graph G contains a complete bipartite subgraph where one part is Y and the other is W' and all other vertices are isolated. Furthermore, it is easy to see that no graph that satisfies one of the properties c-Core and c-Connectivity can contain an isolated vertex. It follows that every subgraph of G that satisfies one of those properties has to be a subgraph of the complete bipartite subgraph in G. In order to be a c-core or c-connected, a complete bipartite graph has to have at least c vertices in each part. Since we have that $|Y| = c$, all vertices of Y have to be contained in each subgraph of G satisfying one of the properties c-Core and c-Connectivity and we get that Condition 3 from Definition 2 is fulfilled. The resulting graph is visualized in Figure 1.

Algorithm A_Π for $\Pi = c$-Truss:

We construct the graph $G = (V, E)$ as follows. Let $X_w := \{w\}$ for all $w \in W$ and $Y := \{u_1, \ldots, u_{c-1}\}$ and hence $V = W \cup \{u_1, \ldots, u_{c-1}\}$. Add edges $\{u, w\}$ to E for all vertices $u \in Y$ and $w \in W' \cup Y$ with $u \neq w$. This clearly fulfills Conditions 1 and 2 from Definition 2 for $f(\alpha) = 1$ and $g(\alpha) = c - 1$. Note that the vertices in Y form a clique of size $c - 1$. Furthermore, every vertex from W' is connected exactly to all vertices in Y and hence every triangle in G contains at least two

https://doi.org/10.1017/nws.2019.13 Published online by Cambridge University Press

vertices from Y and for every $w \in W'$, we have that $Y \cup \{w\}$ forms a clique of size c and hence a c-truss. It follows that for each $W'' \subseteq W'$ with $W'' \neq \emptyset$, we have that $G[W'' \cup Y]$ is a c-truss. Since vertices in W' are not connected to each other, all vertices in Y are necessary to produce enough triangles for the edges going from W'' to Y. Also, we cannot add any isolated vertices to the subgraph since we require c-trusses to be connected. It follows that Condition 3 from Definition 2 is fulfilled.

Algorithm A_Π for $\Pi \in$ {Matching, c-Factor}:

Recall that Matching = 1-Factor. Hence, we describe the algorithm only for the more general c-Factor property. We construct the graph $G = (V, E)$ as follows. (For Matching set $c = 1$.) Let $X_w := \{w_1, \ldots, w_{c+1}\}$ for all $w \in W$ and $Y := \emptyset$. Add all edges $\{w_i, w_j\}$ with $1 \leq i < j \leq c+1$ to E for all $w \in W'$. This clearly fulfills Conditions 1 and 2 from Definition 2 for $f(\alpha) = c+1$ and $g(\alpha) = 0$. Note that for each $w \in W'$, $G[X_v]$ is a complete graph of size $c+1$ and hence a connected c-regular graph. Furthermore, for any set $X' \subset X_w$ and any set $X'' \subseteq V \setminus X_w$, we have that $G[X']$ is a connected component of $G[X' \cup X'']$ with a minimum degree strictly smaller than c. Hence, $G[X' \cup X'']$ does not have a c-factor. Together with the fact that all vertices in the sets X_w with $w \in W \setminus W'$ are isolated, it follows that if there is a vertex set X such that $G[X]$ has a c-factor, then $X = \bigcup_{w \in W''} X_v$ for some $W'' \subseteq W'$. Hence, we get that Condition 3 from Definition 2 is fulfilled.

Algorithm A_Π for Π = Hamiltonian:

We construct the graph $G = (V, E)$ as follows. Let $X_w := \{w\}$ for all $w \in W$ and $Y := \emptyset$ and hence $V = W$. Add edges $\{u, v\}$ to E for all vertices $u, v \in W'$. This clearly fulfills Conditions 1 and 2 from Definition 2 for $f(\alpha) = 1$ and $g(\alpha) = 0$. Note that the vertices W' form a clique in G and all other vertices are isolated. Furthermore, we have that any clique is also a Hamiltonian subgraph and any Hamiltonian subgraph cannot contain any isolated vertices. Hence, we get that Condition 3 from Definition 2 is fulfilled.

The results follow from the existence of the described algorithms. □

A particular consequence of Corollary 5 is that, while the polynomial-time solvability of the connected component detection algorithm for two layers by Gai et al. (2003) generalizes to any constant number of layers (Proposition 3), it does not generalize to an arbitrary, given number of layers.

5. Matching and c-Factors

In this section, we consider the graph properties Matching and its generalization c-Factor. To recall, a graph G has property Matching if it contains a perfect matching and G has property c-Factor if it has a c-regular subgraph containing all vertices of G. Finding a maximum c-factor for a given graph is polynomial-time solvable for all c (see Plummer 2007 or Nichterlein 2014, Chapter 3, for an overview on graph factors). For these properties, Theorem 1 shows that Π-ML-Subgraph is W[1]-hard with respect to the number k of vertices and the number ℓ of layers to select. Theorem 1 does not rule out fixed-parameter tractability with respect to the total number t of layers, or with respect to the number k of vertices to select if the number ℓ of layers to select is constant. In this section, however, we show that both Matching-ML-Subgraph and c-Factor-ML-Subgraph are hard even for a constant number of layers, thus strengthening the statement of Theorem 1 for these properties.

5.1 Matching

As mentioned, through Theorem 1 we get in Corollary 5 in Section 4 that Matching-ML-Subgraph is W[1]-hard when parameterized by the combined parameter number k of vertices to select and number ℓ of layers to select. Through closer inspection, we can get a stronger result. We show that Matching-ML-Subgraph is polynomial-time solvable for $\ell \leq 2$ and becomes

https://doi.org/10.1017/nws.2019.13 Published online by Cambridge University Press

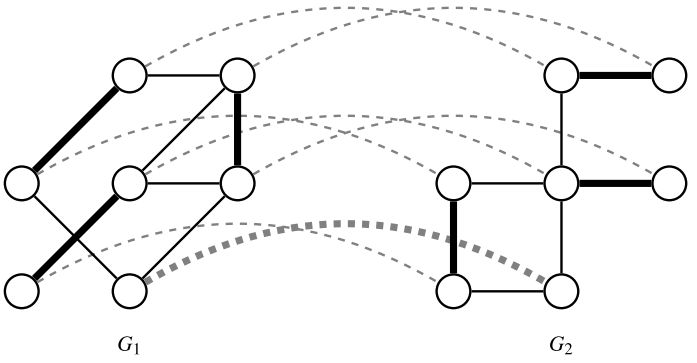


Figure 2. Construction of the graph $G' = (V', E')$ from graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_1)$. Black edges have weight $|V| + 1$ and gray dashed edges have weight $|V|$. The thick edges are a maximum-weight matching for G'.

W[1]-hard when parameterized by the number k of vertices to select already for $\ell \geq 3$. Intuitively, the reason for the computational complexity transition from two layers to three layers is as follows. By overlaying two matchings, one may create cycles and paths but without connections between them. We can cope with this by finding a maximum weighted matching in an auxiliary graph. Adding a third layer, however, allows arbitrary connections between cycles and paths, which allows the construction of gadgets to show hardness.

We now reduce Matching-ML-SUBGRAPH with $\ell = 2$ layers to MAXIMUM WEIGHT MATCHING, that is, to the problem where we are given a graph with edge weights and ask to find a matching with maximum edge weights. To this end, let $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ be the two layers of the input graph for which we would like to know whether there is an $X \subseteq V$ of size at least k such that both $G_1[X]$ and $G_2[X]$ have a perfect matching. The reduction is given in the following lemma and visualized in Figure 2.

Lemma 1. *Given two graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$, define a graph $G' = (V', E')$ as follows:*

- $V' = \{v_1, v_2 \mid v \in V\}$ *and*
- $E' = \{\{v_1, v_2\} \mid v \in V\} \cup \{\{u_i, v_i\} \mid \{u, v\} \in E_i\}$.

Define a weight function $w\colon E' \to \mathbb{N}$ as follows; let $n := |V|$:

$$w(\{u_i, v_j\}) = \begin{cases} n & \text{if } i \neq j \text{ and } u = v \\ n+1 & \text{if } i = j \text{ (and } u \neq v\text{)} \end{cases}$$

Let $k \in \mathbb{N}$. Then, there is a set $X \subseteq V$ of size at least k such that both $G_1[X]$ and $G_2[X]$ have a perfect matching if and only if the graph G' has a matching of w-weight at least $n^2 + k$.

Proof. Assume that G' has a matching $M' \subseteq E'$ with $w(M') \geq n^2 + k$. Let us first check that M' is in fact a perfect matching: If $|M'| \leq n - 1$, then $w(M') \leq (n+1)(n-1) = n^2 - 1 < n^2 + k$, which would contradict the choice of M'. Thus $|M'| \geq n$ but then M' must have exactly n edges since G' has $2n$ vertices. That is, M' is a perfect matching. Now we show how to get a vertex set $X \subseteq V$ such that both $G_1[X]$ and $G_2[X]$ have perfect matchings. Let $Y := \{v \mid v \in V \wedge \{v_1, v_2\} \in M'\}$; that is, Y is the set of vertices of V whose copies in G' are matched to each other under M'. Let $X := V \setminus Y$. It can be easily checked that both $G_1[X]$ and $G_2[X]$ have perfect matchings; we show this for $G_1[X]$: For any $v \in X$, we know that $\{v_1, v_2\} \notin M'$, or else we would have $v \in Y$ and $v \notin X$. Thus, using that M' is a perfect matching, v_1 must be matched to another vertex u_1, which is then also in X, by definition. It follows that M' induces a perfect matching on $G'[X_1]$, where $X_1 := \{v_1 \mid v \in X\}$. Since $G_1[X]$ is an isomorphic copy of $G'[X_1]$ under the canonical isomorphism $\phi\colon v \mapsto v_1$, we get that $G_1[X]$ also has a perfect matching.

https://doi.org/10.1017/nws.2019.13 Published online by Cambridge University Press

It remains to check that X has size at least k: Observe that $|X| + |Y| = n$ since every vertex of V is either in X or in Y. Each $v \in Y$ corresponds to a matching edge $\{v_1, v_2\} \in M'$ which has weight n under w. Thus, if $|X| < k$, then $|Y| > n - k$, which implies that $w(M') \leq |X| \cdot (n+1) + |Y|n < kn + k + (n-k)n = n^2 + k$, contradicting the choice of M'.

Assume now that both $G_1[X]$ and $G_2[X]$ have perfect matchings M_1 and M_2 for some X of size at least k. Clearly, the size of X must be even. Define a matching M' of G' by

$$M' := \{\{v_1, v_2\} \mid v \in V \setminus X\} \cup \{\{u_i, v_i\} \mid \{u, v\} \in M_i\}$$

In other words, M' is obtained by copying M_1 and M_2 to G' in the obvious way and matching all leftover vertices by the edges between the copies of the same vertex.

Clearly, for each vertex $v \in V \setminus X$ this adds an edge $\{v_1, v_2\}$ of weight n to M'. From M_1 and M_2, we copied $\frac{|X|}{2}$ edges each, which results in exactly $|X| \geq k$ edges of weight $n+1$ in M'. Thus, the total weight of M' is $n^2 + k$, as claimed. □

To show that Matching-ML-SUBGRAPH remains W[1]-hard for any $\ell \geq 3$ when parameterized by k, we reduce from MULTICOLORED CLIQUE that is known to be W[1]-hard when parameterized by the solution size (Fellows et al., 2009).

Theorem 2. *Matching-ML-SUBGRAPH can be solved in polynomial time if $\ell \leq 2$. It is NP-hard and W[1]-hard when parameterized by the number of vertices to select k for all $\ell \geq 3$ and total numbers of layers $t \geq \ell$.*

Proof. We get the polynomial-time solvability of Matching-ML-SUBGRAPH for the case $\ell \leq 2$ from Lemma 1. For the case of $\ell \geq 3$, we give a parameterized reduction from MULTICOLORED CLIQUE. In MULTICOLORED CLIQUE, we are given an h-partite graph $H = (U_1 \uplus \ldots \uplus U_h, F)$ and need to determine whether it contains a clique of size h. Note that such a clique necessarily contains exactly one vertex from each set U_i, and cliques of more than h vertices are impossible. We say that vertices from U_i have *color i*.

Reduction idea. The main idea is to create the first two layers in such a way that selecting a subgraph that admits a perfect matching in both layers corresponds to selecting exactly one vertex of each color. The third layer is constructed in a way that each subgraph that admits perfect matchings in the first two layers admits also a perfect matching in this layer if all selected vertices are pairwise connected and hence form a multicolored clique.

Without loss of generality, we assume that the number h of colors is even. We construct an instance of Matching-ML-SUBGRAPH for $t = \ell = 3$ as follows and then argue that the construction is easily generalizable to larger numbers of layers.

Vertices. First, create $h-1$ vertices for each vertex in graph H (one vertex for each color other than its own color). Formally, for each color $1 \leq j \leq h$ and each $u_i \in U_j$, create the vertex set V_i consisting of the vertices $v_{(i,j')}, j' \in (\{1, \ldots, h\} \setminus \{j\})$. Second, create one *color vertex* w_j for each color $j \in \{1, \ldots, h\}$. We denote the set of color vertices as $W := \bigcup_{1 \leq j \leq h} \{w_j\}$.

Vertex selection gadget by graphs G_1 and G_2. The vertex selection gadget is intended to make sure the vertices selected in any valid solution of the constructed Matching-ML-SUBGRAPH instance correspond to pairwise differently colored vertices of H. The vertex selection gadget is visualized in Figure 3. We construct it as follows. For each color $1 \leq j \leq h$, create for each $u_i \in U_j$ one cycle on $\{w_j\} \cup V_i$ in the graph $G_1 \cup G_2$ such that the edges are alternatingly from G_1 and from G_2. These $|U_j|$ cycles are all of length h and share only the color vertex w_j. To realize this, create the following edges. For each $1 \leq z \leq h-2$, create an edge in graph $G_{(z \bmod 2)+1}$ between $v_{(i,z)}$ and $v_{(i,z+1)}$ if $z < j-1$, between $v_{(i,z+1)}$ and $v_{(i,z+2)}$ if $z \geq j$, and between $v_{(i,z)}$ and $v_{(i,z+2)}$ if $z = j-1$. Create an edge between w_j and $v_{(i,1)}$ in graph G_2, between $v_{(i,h)}$ and w_j in graph G_1 if $j \neq h$, and between $v_{(i,h-1)}$ and w_j in graph G_1 if $j = h$.

https://doi.org/10.1017/nws.2019.13 Published online by Cambridge University Press

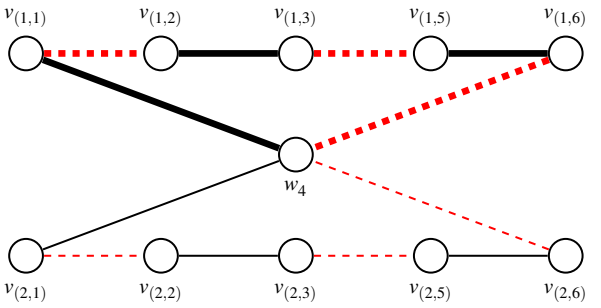


Figure 3. Parts of the vertex selection gadget G_1 and G_2 for two vertices $u_1, u_2 \in U_4$, where the number of colors is $h = 6$. Black edges belong to E_1 and red dashed edges belong to E_2. The thick and the thin edges both create cycles that have edges alternating between Layers 1 and 2. Since the color vertex w_4 is contained in both cycles, only one of these cycles can be contained in a matching subgraph.

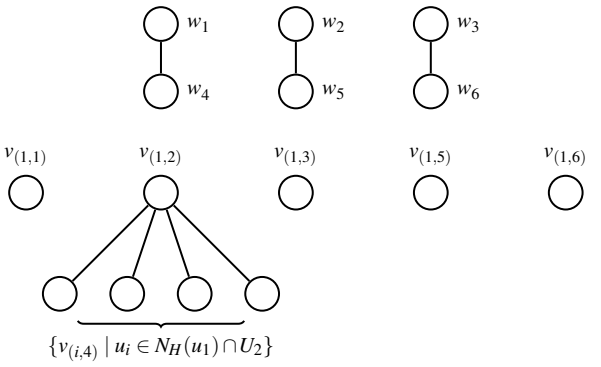


Figure 4. Parts of the validation gadget G_3 for $h = 6$ and a vertex $u_1 \in U_4$. The color vertices w_1 to w_6 form a matching and each vertex $v_{(1,j)}$ is connected to all vertices $v_{(i,4)}$ with $u_i \in N_H(u_1) \cap U_j$, as exemplarily visualized for $v_{(1,2)}$.

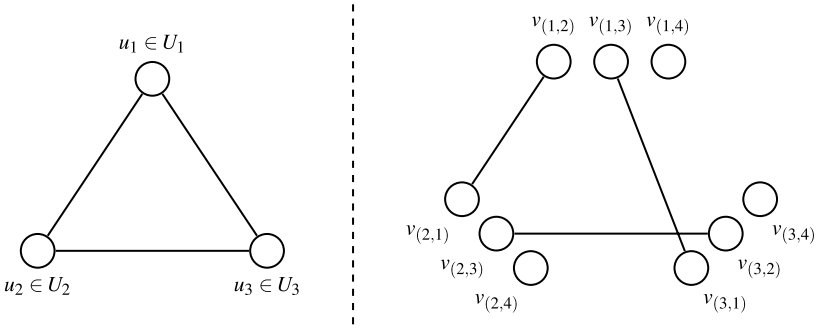


Figure 5. Parts of the validation gadget G_3 for $h = 4$ and a triangle consisting of $u_1 \in U_1$, $u_2 \in U_2$, and $u_3 \in U_3$. The triangle (to be interpreted as a part of the original graph) is depicted on the left, and the corresponding part of the gadget is depicted on the right. The color vertices w_1 to w_4 are omitted.

Validation gadget by graph G_3. The validation gadget is intended to make sure the vertices selected in any valid solution of the constructed Matching-ML-SUBGRAPH instance correspond to a clique in H. The validation gadget is visualized in Figures 4 and 5. It is constructed as follows. For each vertex pair u_i, $u_{i'}$ with $u_i \in U_j$ and $u_{i'} \in U_{j'}$ such that u_i and $u_{i'}$ are adjacent in H, we create an edge between $v_{(i,j')}$ and $v_{(i',j)}$ in G_3. Furthermore, create the edge $\{w_j, w_{j+h/2}\}$ for each $1 \leq j \leq h/2$.

Finally, by setting $k = h^2$ and $t = \ell = 3$, we complete the construction, which can clearly be performed in polynomial time and the new parameter k solely depends on h.

Correctness. It remains to show that graph H has a clique that contains each color exactly once if and only if there is a vertex set $X \subseteq V$ with $|X| \geq k$ such that graph $G_z[X]$ contains a perfect matching for each $1 \leq z \leq 3$.

https://doi.org/10.1017/nws.2019.13 Published online by Cambridge University Press

($\Rightarrow$): Assume that graph H has a clique $K := \{u_1, u_2, \ldots, u_h\}$ and, without loss of generality, $u_i \in U_i$ for all $1 \leq i \leq h$. We show that $X := W \cup V_1 \cup V_2 \cup \cdots \cup V_h$ is a solution for our Matching-ML-SUBGRAPH instance. By construction, X is of size $h + h \cdot (h-1) = h^2 = k$. It remains to show that graph $G_z[X]$ has a perfect matching for each $1 \leq z \leq 3$. Recall that we created for each color $1 \leq j \leq h$ and for each vertex $u_i \in U_j$ one cycle on $\{w_j\} \cup V_i$ in the graph $G_1 \cup G_2$ such that the edges alternatingly are from G_1 and G_2. Since X only contains one of these cycles for each color, a perfect matching is easy to find for graphs $G_1[X]$ and for $G_2[X]$. For graph $G_3[X]$, we can find the matching $\{\{v_{(i,j)}, v_{(j,i)}\} \mid (1 \leq i, j \leq h) \wedge (i \neq j)\} \cup \{\{w_j, w_{j+h/2}\} \mid 1 \leq j \leq h/2\}$, since, by construction, $v_{(i,j)}$ is adjacent to $v_{(j,i)}$ if u_i is adjacent to u_j, and $u_i \in U_i$ and $u_j \in U_j$ (which is the case since K is a clique).

($\Leftarrow$): Assume that there is a vertex set $X \subseteq V$ with $|X| \geq k$ such that graph $G_z[X]$ contains a perfect matching for each $1 \leq z \leq 3$. First, consider the graph $G_1 \cup G_2$ and some pair of vertices $\{x_1, x_2\} \subseteq X$ that is matched in G_1 or in G_2. Then, these two vertices must be from the same cycle $\{w_j\} \cup V_i$ for some $1 \leq j \leq h$ and $u_i \in U_j$, since otherwise there is no edge between them in any of the two graphs. Furthermore, if two vertices from $\{w_j\} \cup V_i$ are in X, then all vertices from $\{w_j\} \cup V_i$ must be in X because otherwise neither G_1 nor G_2 has a perfect matching: Every vertex except w_j has exactly one neighbor in G_1 and one neighbor in G_2, which are both enforced to be also contained in X—this enforces the whole cycle $\{w_j\} \cup V_i$ to be contained in X. However, X contains the vertices from $\{w_j\} \cup V_i$ for at most one i for every color $1 \leq j \leq h$, because w_j can only be matched to one vertex in G_1 and to one vertex in G_2. This implies that X contains for each $1 \leq j \leq h$ all vertices from $\{w_j\} \cup V_i$ for exactly one i, since $|X| \geq k = h^2$ and $|\{w_j\} \cup V_i| = h$. Without loss of generality, let $V_i \subseteq X$ for all $1 \leq i \leq h$ and let u_i be the vertex in graph H corresponding to V_i. We show that $K = \{u_1, u_2, \ldots, u_h\}$ is a clique in H. In graph $G_3[X]$, each color vertex w_j must be matched to its only neighbor: $w_{j+h/2}$ if $j \leq h/2$ and $w_{j-h/2}$ if $j > h/2$ (and cannot be matched to $v_{(i,j)}$-vertices). Let $\{u_i, u_{i'}\} \subseteq K$, with $u_i \in U_j$ and $u_{i'} \in U_{j'}$. Then, note that $v_{(i,j')}$ and $v_{(i',j)}$ must be matched since, by construction of G_3, vertex $v_{(i,j')}$ is only adjacent to vertex $v_{(i',j)}$. Moreover, if $v_{(i,j')}$ is adjacent to $v_{(i',j)}$ in G_3, then u_i is adjacent to $u_{i'}$ in H. Thus overall, K is a h-vertex clique in H, as required.

To make this reduction work for any $t \geq \ell > 3$, we insert $\ell - 3$ additional layers of complete graphs and $t - \ell$ layers of edgeless graphs. □

5.2 c-Factors

We now show that c-Factor-ML-SUBGRAPH with $c \geq 2$ is W[1]-hard when parameterized by k for $\ell \geq 2$. We reduce from MULTICOLORED CLIQUE that is known to be W[1]-hard when parameterized by the solution size (Fellows et al., 2009). The hardness reduction is similar to the one we use in the proof of Theorem 2. Intuitively, since there are connected c-regular graphs for $c \geq 2$, we only need one layer to build a vertex selection gadget, whereas in the 1-factor (matching) case we need two layers.

Theorem 3. *For $c \geq 2$, c-Factor-ML-SUBGRAPH is NP-hard and W[1]-hard when parameterized by the number k of layers to select for all $\ell \geq 2$ and all $t \geq \ell$.*

Proof. In the following, we prove that, for $c \geq 2$, c-Factor-ML-SUBGRAPH is W[1]-hard when parameterized by k for $t = \ell = 2$ and then argue that the construction is easily generalizable. As in the proof of Theorem 2, we give a parameterized reduction from MULTICOLORED CLIQUE: We are given an h-partite graph $H = (U_1 \uplus \ldots \uplus U_h, F)$ and need to determine whether it contains a clique of size h.

Reduction idea. The idea of the reduction is similar to the one for Matching in the proof of Theorem 2 but already works for $\ell = 2$ in the c-factor case ($c \geq 2$). The reason that we only need

https://doi.org/10.1017/nws.2019.13 Published online by Cambridge University Press

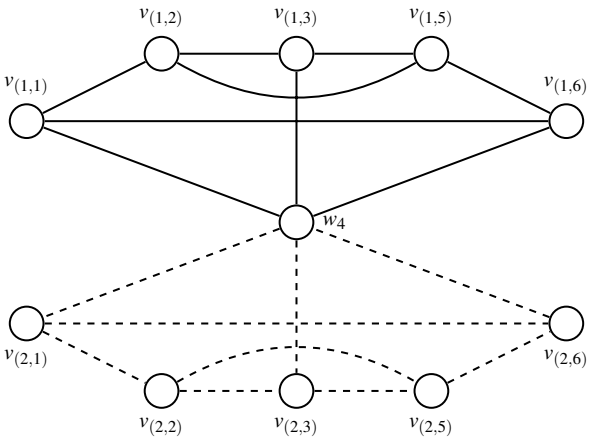


Figure 6. Parts of the vertex selection gadget G_1 for two vertices $u_1, u_2 \in U_4$, where the number of colors is $h = 6$ and $c = 3$. The normal and the dashed edges both create c-regular subgraphs. Since the color vertex w_4 is contained in both subgraphs, only one of these subgraphs can be selected. The complete subgraph on V_F is not depicted.

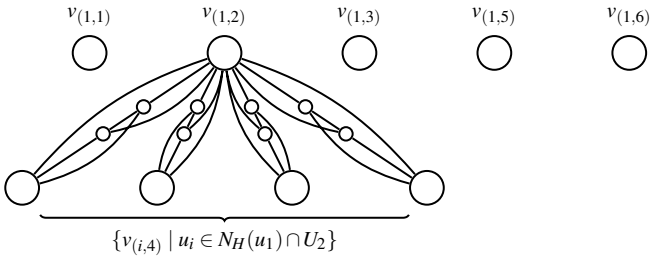


Figure 7. Parts of the validation gadget G_2 for $c = 3$, $h = 6$, and a vertex $u_1 \in U_4$. Each vertex $v_{(1,j)}$ and each vertex $v_{(i,4)}$ with $u_i \in N_H(u_1) \cap U_j$ together with the vertices in $V_{\{u_1,u_i\}}$ (depicted as the small vertices) forms a complete subgraph of size c, as exemplarily visualized for $v_{(1,2)}$. The color vertices w_1 to w_6 form a complete subgraph and are not depicted.

two layers is that we can construct connected c-regular graphs for $c \geq 2$ of almost arbitrary size and this allows us to construct a vertex selection gadget with only one layer.

We say that vertices from U_i have *color i*. Without loss of generality, assume that $h \geq c + 1$, that h is even, and that $|F| \geq 3$.

Vertices. For each color j, $1 \leq j \leq h$, we do the following: We create a color vertex w_j and for each vertex u_i in U_j, and we create a set of $h - 1$ vertices $V_i = \{v_{(i,j')} \mid 1 \leq j' \leq h \text{ and } j' \neq j\}$. Note that we have one vertex in V_i for each color except the color of u_i. Let $W = \{w_j \mid 1 \leq j \leq h\}$. Furthermore, for each edge $f \in F$, we create a set of vertices V_f with $|V_f| = c - 1$. Let $V_F = \bigcup_{f \in F} V_f$ and $V = \bigcup_i V_i \cup W \cup V_F$.

Vertex selection gadget by graph G_1. The vertex selection gadget is intended to make sure the vertices selected in any valid solution of the constructed Matching-ML-SUBGRAPH instance correspond to pairwise differently colored vertices of H. The vertex selection gadget is visualized in Figure 6. It is constructed as follows. For every $1 \leq j \leq h$ and every u_i in U_j, we do the following: We create a connected c-regular graph on the vertex set $V_i \cup \{w_j\}$. Note that this can be done as follows: We order the vertices in $V_i \cup \{w_j\}$ arbitrarily and connect each vertex to the $\lfloor c/2 \rfloor$ subsequent vertices, wrapping around at the end. If c is odd, then we additionally connect each vertex v with the vertex at position $(x + h/2) \bmod h$ in the ordering, where x is the position of vertex v. Furthermore, we create a complete graph on the vertices in V_F.

Validation gadget by graph G_2. The validation gadget is intended to make sure the vertices selected in any valid solution of the constructed Matching-ML-SUBGRAPH instance correspond to vertices of H that form a clique in H. The validation gadget is visualized in Figure 7. It is constructed as follows. For each edge $f \in F$, we do the following: Let u_i and $u_{i'}$ be the endpoints of f and $u_i \in U_j$ and $u_{i'} \in U_{j'}$. We create a complete graph on the vertices in $V_f \cup \{v_{(i,j')}, v_{(i',j)}\}$. Note that this complete graph has order $c + 1$ and hence is a c-regular graph. Furthermore, we create a complete graph on all vertices in W.

https://doi.org/10.1017/nws.2019.13 Published online by Cambridge University Press

By setting $k = h^2 + \frac{1}{2}h(h-1)\cdot(c-1)$, we complete the construction, which can clearly be performed in polynomial time and the new parameter k solely depends on h.

Correctness. ($\Rightarrow$): Assume that graph H has an h-colored clique K and without loss of generality $K = \{u_1, u_2, \ldots, u_h\}$ and $u_i \in U_i$ for each $i \in \{1, \ldots, h\}$. Let F_K denote the set of all edges in the clique K. Furthermore, let $V_K = \bigcup_{1\leq i\leq h} V_i$ and $V_{F_K} = \bigcup_{f\in F_K} V_f$. We show that $X = W \cup V_K \cup V_{F_K}$ is a solution for c-FACTOR-ML-SUBGRAPH. Note that $|X| = h^2 + \frac{1}{2}h(h-1)\cdot(c-1)$ by construction. It remains to show that $G_1[X]$ and $G_2[X]$ each have c-factors. Observe that for any graph $G = (V, E)$ and any partition $\mathcal{P} = \{P_1, P_2, \ldots, P_p\}$ of V, we have that, if $G[P_i]$ has a c-factor for all i, $1 \leq i \leq p$, then G has a c-factor as well.

(1) Note that $G_1[V_{F_K}]$ is a complete graph of order $\frac{1}{2}h(h-1)\cdot(c-1) \geq 2c+1$ and hence has a c-factor. Furthermore, let $u_i \in K$ and assume that $u_i \in U_j$. Then we have that $G_1[V_i \cup \{w_j\}]$ is by construction a c-regular graph and hence also has a c-factor. Since K is h-colored, $\{V_i \cup \{w_j\} \mid u_i \in K \text{ and } u_i \in U_j\}$ is a partition of $V_K \cup W$.

(2) Note that $G_2[W]$ is a complete graph of size $h \geq c+1$ and hence has a c-factor. Let $f \in F_K$ be the edge connecting u_i and $u_{i'}$, and let $u_i \in U_j$ and $u_{i'} \in U_{j'}$. Then we have that $G_2[V_f \cup \{v_{(i,j')}, v_{(i',j)}\}]$ is by construction a complete graph of order $c+1$ and hence also has a c-factor. Since K is an h-colored clique, we have that $\{V_f \cup \{v_{(i,j')}, v_{(i',j)}\} \mid f \in F_K \text{ and } f = \{u_i, u_{i'}\}\}$ is a partition of $V_{F_K} \cup V_K$. Notably, we also have that $\mathcal{P}_K = \{\{v_{(i,j')}, v_{(i',j)}\} \mid \{u_i, u_{i'}\} = f \text{ for some } f \in F_K\}$ is a partition of V_K, since any u_i is connected to $h-1$ other vertices with a different color each and therefore we have that $\{v_{(i,j')} \mid \{v_{(i,j')}, v_{(i',j)}\} \in \mathcal{P}_K\} = V_i$.

($\Leftarrow$): Assume that there is a vertex set $X \subseteq V$ such that $|X| \geq k$ and $G_i[X]$ has a c-factor for ℓ different layers i. By construction of the layers, we have that $G_1[X]$ and $G_2[X]$ both have c-factors. Furthermore, we show the following facts:

Fact 1: If $v_{(i,j')} \in X$ and $u_i \in U_j$, then $V_i \subseteq X$ and $w_j \in X$: By construction $G_1[V_i \cup \{w_j\}]$ is a connected c-regular graph and each $v_{(i,j')} \in V_i$ is *only* connected to other vertices in $V_i \cup \{w_j\}$ in G_1. Note that any proper subgraph of a connected c-regular graph is not c-regular and does not have a c-factor. It follows that as soon as any $v_{(i,j')} \in V_i$ is included in X, all other vertices in V_i have to be included, as well as w_j, the vertex corresponding to the color of u_i.

Fact 2: If $V_i \subseteq X$ and $V_{i'} \subseteq X$ and $u_i \in U_j$, then $u_{i'} \notin U_j$: Note that $G_1[V_{i'}]$ does not have a c-factor. By construction, we get a connected c-regular graph by adding the vertex corresponding to the color of $u_{i'}$, hence $G_1[V_{i'}]$ is a proper subgraph of a connected c-regular graph. Furthermore, we have that w_j is already part of the c-regular spanning graph of $G_1[V_i \cup \{w_j\}]$; therefore, $u_{i'}$ cannot have color j, that is, $u_{i'} \notin U_j$.

Fact 3: If $X \cap V_f \neq \emptyset$ for some $f \in F$, then $V_f \cup \{v_{(i,j')}, v_{(i',j)}\} \subseteq X$, where $u_i \in U_j$ and $u_{i'} \in U_{j'}$ are the endpoints of f: By construction, $G_2[V_f \cup \{v_{(i,j')}, v_{(i',j)}\}]$ is a clique of size $c+1$ and hence a connected c-regular graph. Furthermore, this clique is disconnected from the rest of G_2 and hence as soon as one of its vertices is included in X, all of them are included.

Now we show that $K = \{u_i \mid V_i \subseteq X\}$ is an h-colored clique in H. First, we show that there must be some $v_{(i,j)} \in X$: Since $|X| \geq h^2 + \frac{1}{2}h(h-1)\cdot(c-1)$, we have that $X \neq W$ and from Fact 3 we get that $X \not\subseteq W \cup V_F$.

By Fact 1, we know that any V_i is either a subset of X or $X \cap V_i = \emptyset$. Furthermore, if $V_i \subseteq X$ and $u_i \in U_j$, so is w_j. Fact 2 yields that we cannot have two vertices of the same color in K. This implies

$$\Big|\bigcup_{V_i\subseteq X} V_i \cup W\Big| \leq h^2 \tag{1}$$

Note that $|X| \geq h^2 + \frac{1}{2}h(h-1)\cdot(c-1)$ and Inequality (1) imply that $X \cap V_F \neq \emptyset$. By Fact 3, X can only include vertices corresponding to edges between vertices u_i and $u_{i'}$ if $V_i \cup V_{i'} \subseteq X$. Therefore,

$$|X \cap V_F| \leq \frac{1}{2}h(h-1)\cdot(c-1) \tag{2}$$

https://doi.org/10.1017/nws.2019.13 Published online by Cambridge University Press

Note that $|X| = h^2 + \frac{1}{2}h(h-1)\cdot(c-1)$ if and only if both Inequalities (1) and (2) are equalities. This implies that $|K| = h$, all vertices in K have different colors and all colors are present (Inequality 1), and that K is a clique in H (Inequality (2)).

To make this reduction work for any $t \geq \ell > 2$, we insert $\ell - 2$ additional layers of complete graphs and $t - \ell$ layers of edgeless graphs. □

6. Hamiltonian paths

In this section, we investigate the problem of finding Hamiltonian subgraphs, that is, subgraphs that have a simple path visiting all vertices. Corollary 5 in Section 4 states that Hamiltonian-ML-SUBGRAPH is W[1]-hard when parameterized by the combined parameter k and ℓ. Through closer inspection, we can get a stronger result. Hamiltonian-SUBGRAPH is known to be NP-hard and FPT when parameterized by the size of the subgraph k (Monien, 1985). For the multilayer case, we can show that it is already W[1]-hard for any constant $\ell \geq 2$.

Theorem 4. *Hamiltonian-ML-SUBGRAPH is NP-hard and W[1]-hard when parameterized by the number k of vertices to select for all $\ell \geq 2$ and $t \geq \ell$.*

Proof. We reduce from the MULTICOLORED BICLIQUE problem. In MULTICOLORED BICLIQUE, we are given a bipartite graph $H = (U \cup W, F)$ and a partitioning that partitions vertices in U and W into h parts each, that is, $U = U_1 \uplus \ldots \uplus U_h$ and $W = W_1 \uplus \ldots \uplus W_h$. We call these $2h$ parts *colors*, that is, each vertex of H is of exactly one color in $\{U_1, \ldots, U_h, W_1, \ldots, W_h\}$. We need to determine whether H contains a biclique of size $2h$ that contains one vertex of each color. A simple parameterized reduction from CLIQUE shows that MULTICOLORED BICLIQUE is W[1]-hard (Dell & Marx, 2012). Observe that the vertex coloring implies that any solution contains h vertices from U and h vertices from W. We will call the vertices from U *low*, and the vertices from W *high*, imagining that the vertices from W are on top of the vertices from U. Similarly, we call colors U_i *low* and colors W_i *high*. Note that all colors are different. Given an instance H of MULTICOLORED BICLIQUE, we construct an instance of Hamiltonian-ML-SUBGRAPH for $t = \ell = 2$ as follows and then argue that the construction is easily generalizable.

Reduction idea. The main idea is to create new vertices for both the vertices of the MULTICOLORED BICLIQUE instance as well as the edges. We create two layers where in the first one any maximal Hamiltonian subgraph contains a path that "selects" one vertex from each color and one edge of each combination of a high and low color. The second layer is constructed in such a way that each maximal Hamiltonian subgraph of the first layer is also Hamiltonian in the second layer if and only if the selected edges indeed connect the selected vertices of the respective colors, implying that the selected vertices form a multicolored biclique.

Vertices. The vertex set V consists of the following subsets:

- all vertices $U \cup W$ of H,
- $\{s_1, s_2\}$, where we assume that s_1 and s_2 are not vertices of H,
- $A_{i,j}$, $1 \leq i \leq h$, $1 \leq j \leq h$, where $A_{i,j} := \{\alpha_{\{u,w\}} \mid \{u, w\} \in F \wedge u \in U_i \wedge w \in W_j\}$, and
- $D_{i,j}$, $1 \leq i \leq h$, $1 \leq j \leq h$, where $D_{i,j} := \{\delta_{\{w,u\}} \mid \{w, u\} \in F \wedge w \in W_i \wedge u \in U_j\}$.

Informally, the latter two sets are constructed by adding two vertices for each edge of H, each corresponding to one orientation of the undirected edge. The vertices are then assigned to the vertex sets according to the colors of their endpoints and the orientations. Oriented edges from U to W (and their corresponding vertices in V) are called *ascending*, and oriented edges from W to U (and their corresponding vertices) are called *descending*. Note that all these sets are non-empty unless we face a no-instance.

Now we describe how to construct a vertex and edge selection gadget by graph G_1 and a validation gadget by graph G_2. We organize the vertices of both graphs in *levels*. Each graph

https://doi.org/10.1017/nws.2019.13 Published online by Cambridge University Press

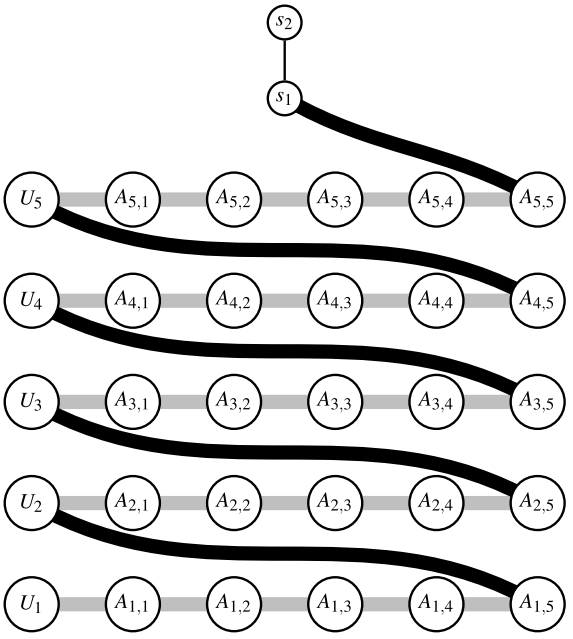


Figure 8. The lower half of the vertex selection gadget G_1 for $h = 5$. The circles annotated with U_i or $A_{i,j}$ correspond to the respective vertex set from the construction, and s_1 and s_2 are single vertices connected by an edge. The thick gray edges represent the edges between the two vertex sets they connect. The thick black edges indicate that all edges between the two vertex sets are present. The upper half of the gadget is symmetric and not depicted.

has $2h^2 + 2h + 2$ levels of vertices with edges only between neighboring levels; each U_i, W_i, $A_{i,j}$ and each $D_{i,j}$ forms one level, and the two remaining levels contain s_1 and s_2, respectively.

Vertex and edge selection gadget by graph G_1. Informally, graph G_1 is constructed by putting all low vertices and their incident ascending edges into low levels, then adding two levels for s_1 and s_2, and then putting all high vertices and their incident descending edges into high levels. More precisely, U_1 is the first level of G and $A_{1,1}$ is the second level. Then edges are added from each $u \in U_1$ to all vertices of $A_{1,1}$ that correspond to an edge incident with u. Then, the ascending vertices "incident" with vertices from U_1 are added for increasing colors of the high endpoints. Afterwards, the vertices from U_2 are added and then the vertices corresponding to their incident edges, and so on. The special vertices s_1 and s_2 are added in two middle levels, separating the levels containing low vertices from those containing high vertices. Formally, the graph G_1 is constructed as follows:

- For each $u \in U_i$, $1 \leq i \leq h$, add an edge to each $\alpha_{\{u,w\}} \in A_{i,1}$;
- for each $\alpha_{\{u,w\}} \in A_{i,j}$, $1 < j < h$, add an edge to each $\alpha_{\{u,w'\}} \in A_{i,j+1}$;
- for each $\alpha_{\{u,w\}} \in A_{i,h}$, $1 \leq i < h$, add an edge to each $u' \in U_{i+1}$;
- for each $\alpha_{\{u,w\}} \in A_{h,h}$, add an edge to s_1;
- add the edge $\{s_1, s_2\}$;
- for each $w \in W_1$, add the edge $\{s_2, w\}$;
- for each $w \in W_i$, $1 < i \leq h$, add an edge to each $\delta_{\{w,u\}} \in D_{i,1}$;
- for each $\delta_{\{w,u\}} \in D_{i,j}$, $1 \leq j < h$, add an edge to each $\delta_{\{w,u'\}} \in D_{i,j+1}$; and
- for each $\delta_{\{w,u\}} \in D_{i,h}$, $1 \leq i < h$, add an edge to each $w' \in W_{i+1}$.

The idea behind the construction is that any path from the first level to the last level corresponds to a selection of $2h$ vertices and of $2h^2$ edges incident with these vertices. The vertex selection gadget is visualized in Figure 8.

Validation gadget by graph G_2. With the second graph G_2, we enforce that the selected ascending and descending edges between each color pair U_i and W_j correspond to the same edge in H and that any path of length $2h + 2h^2 + 1$ passes through each level of G_1 and each level of G_2.

https://doi.org/10.1017/nws.2019.13 Published online by Cambridge University Press

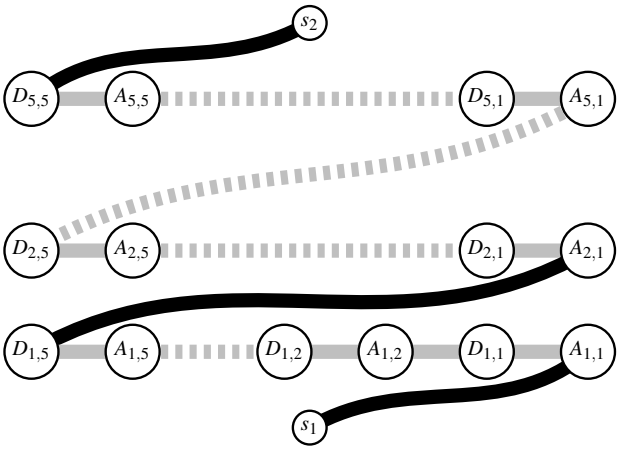


Figure 9. The lower half of the levels of the validation gadget G_2 for $h = 5$. The circles annotated with $A_{i,j}$ or $D_{i,j}$ correspond to the respective vertex set from the construction, and s_1 and s_2 are single vertices. The thick gray edges represent the edges between the two vertex sets they represent. The thick black edges indicate that all possible edges between the respective vertex sets are present. Dashed edges indicate that some vertex sets are not visualized. The higher levels of the gadget are not depicted.

Formally, G_2 is constructed as follows. Herein, assume an arbitrary but fixed ordering on pairs of low and high colors.

- Level 1 contains s_1;
- level $2i$, $1 \leq i \leq h^2$, contains all ascending vertices of the ith color pair;
- level $2i + 1$, $1 \leq i \leq h^2$, contains all descending vertices of the ith color pair;
- level $2h^2 + 2$ contains s_2;
- level $2h^2 + 2 + i$, $1 \leq i \leq h$, contains all vertices from U_i; and
- level $2h^2 + h + 2 + i$, $1 \leq i \leq h$, contains all vertices from W_i.

All edges between consecutive levels are added except for the levels $2i$ and $2i + 1$, $1 \leq i \leq h^2$: Here, we add only an edge between vertices that correspond to the same edge; that is, we add the edge set $\{\alpha_{\{u,w\}}, \delta_{\{w,u\}} \mid \{u, w\} \in F\}$. The validation gadget is visualized in Figure 9.

To finish the construction, we set $k := 2h + 2h^2 + 2$. The reduction clearly runs in polynomial time and the new parameter k depends only on the parameter h of the MULTICOLORED BICLIQUE instance.

Correctness. Thus it remains to show equivalence of the instances.

($\Rightarrow$): Let $K = \{u_1, \ldots, u_h, w_1, \ldots, w_h\}$ be a multicolored biclique in H. Let X be the vertex set containing K, s_1, and s_2, and for each edge e of $H[K]$, the ascending and the descending vertex corresponding to e. We show that $G_1[X]$ and $G_2[X]$ have a Hamiltonian path. In G_1, this path starts at u_1, then visits α_{u_1,w_1}, and then α_{u_1,w_2} until α_{u_1,w_h}. Then it visits u_2 and the ascending vertices corresponding to edges incident with u_2 in the same fashion, that is, first α_{u_2,w_1}, then α_{u_2,w_2}, and so on. This is continued until α_{u_h,w_h} is visited. Then, the path visits s_1 and s_2. Then it visits the high vertices and the descending vertices for their incident edges in the same fashion. By construction, all necessary edges are present: Neighboring edge vertices correspond to edges that share one endpoint, and after a vertex α_i, the next visited edge vertex is incident with α_i.

In G_2, the path visits each level exactly once, going from level 1 through level k. The necessary edges are present since the only neighboring levels that are not complete bipartite graphs are those that contain ascending and descending vertices of the same color pair. Since for each color pair the ascending and descending vertices in S correspond to the same edge in H, they are adjacent in G_2.

($\Leftarrow$): Observe that any vertex set consisting only of vertices from $\{s_1, s_2\} \cup \bigcup_{1 \leq i \leq h} (U_i \cup W_i)$ is disconnected either in G_1 or in G_2. Thus, the set X contains at least one ascending or descending vertex. In either case, it must also contain an edge vertex of the other type: In G_2, the levels containing ascending and descending vertices alternate and any set containing either only vertices of the first two levels of G_2 or only of vertices of the last $2h + 2$ levels is disconnected in G_1. Now, since X contains an ascending and a descending vertex and since $G_1[X]$ is connected, the vertices s_1 and s_2

https://doi.org/10.1017/nws.2019.13 Published online by Cambridge University Press

are contained in X. Thus, the vertex set X contains a vertex from the first and the last levels of $2h + 2h^2 + 2$ levels in G_2. This implies that it contains exactly one vertex of each level of G_2. Thus, the vertex set X contains exactly h low vertices and h high vertices with different colors and it contains for each color pair an ascending and a descending vertex. By construction of G_2 and since $G_2[X]$ has a Hamiltonian path visiting each level exactly once, these two vertices are the same; that is, the $2h^2$ ascending and descending vertices correspond to h^2 edges in H. By construction of G_1 and since $G_1[X]$ has a Hamiltonian path visiting each level of G_1 exactly once, these edges are incident only with vertices of $(U \cup W) \cap X$. Thus, $H[(U \cup W) \cap X]$ is a multicolored biclique.

To make this reduction work for any $t \geq \ell \geq 2$, we can insert $\ell - 2$ additional layers of complete graphs and $t - \ell$ layers of edgeless graphs. □

7. Conclusion

We performed a systematic study of the (parameterized) computational complexity of subgraph detection problems in multilayer networks. In particular, we encountered several computational hardness results for multilayer subgraph detection problems that are solvable in polynomial time in the single-layer case. In the following, we list some possibilities for future research with the goal to obtain positive algorithmic results.

First, the case of two-layer graphs should continue to receive special attention. We showed that Matching-ML-SUBGRAPH is solvable in polynomial time in the two-layer case, whereas it is W[1]-hard when parameterized by the number of vertices to select k for three or more layers. Considering acyclic subgraphs, Agrawal et al. (2018b) also showed specialized algorithms for the two-layer case. Also Cai & Ye (2014) focused mostly on the two-layer case. It would be interesting to systematically explore which subgraph detection problems are tractable in the two-layer case and to identify more problems that behave differently for two and three layers.

Second, in many applications the input graphs are directed. One of our hardness results transfers directly to this case: The construction in the reduction from MULTICOLORED BICLIQUE to Hamiltonian-ML-SUBGRAPH (proof of Theorem 4) can be easily adapted to yield directed acyclic graphs by orienting all edges from lower levels to higher levels, implying that Hamiltonian-ML-SUBGRAPH is NP-hard and W[1]-hard when parameterized by k for all $\ell \geq 2$ and $t \geq \ell$ if every layer is a directed acyclic graph. Hence, for directed acyclic graphs the complexity gap between the cases with one and two layers is even bigger because we can find a longest path in a directed acyclic graph in polynomial-time (as opposed to being NP-hard and FPT with respect to subgraph order k in the undirected single-layer case). As also already mentioned by Cai & Ye (2014), finding positive algorithmic results for multilayer subgraph problems in directed graphs is a challenging open direction.

Finally, in some applications, one is interested in different subgraph properties for each layer (Cai & Ye, 2014; Agrawal et al., 2018a). For example, in the SUPPORTED PATH problem, one layer is undirected and one layer is directed, and one aims to find an induced subgraph that is connected in the undirected layer and Hamiltonian in the directed layer (Fertin et al., 2015). There is only limited systematic investigation of the complexity of such mixed multilayer subgraph problems (Cai & Ye, 2014; Agrawal et al., 2018a).

Acknowledgments. HM was partially supported by the DFG, projects DAPA (NI 369/12) and MATE (NI 369/17). MS was supported by the DFG, project DAPA (NI 369/12), the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme (FP7/2007-2013) under REA grant agreement number 631163.11, by the Israel Science Foundation (grant no. 551145/14), and by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program under grant agreement number 714704. Parts of MS work were done while with TU Berlin, Germany, and Ben-Gurion University of the Negev, Beer Sheva, Israel. CK was supported by the DFG, project MAGZ (KO 3669/4-1). RB was partially supported by the DFG, fellowship BR 5207/2. This work was initiated at the research retreat of the TU Berlin Algorithmics and Computational Complexity research group held in Darlingerode, Harz mountains, Germany, April 2014. The authors would like to thank Sepp Hartung for initial discussions.



https://doi.org/10.1017/nws.2019.13 Published online by Cambridge University Press

Conflict of interest. All authors have nothing to disclose.

Notes

1 Note that for hereditary graph properties requiring vertex sets of size exactly k or at least k is equivalent in terms of computational complexity. However, for connectivity, which is not a hereditary property, it can make a difference.
2 The *h-index* of a graph is the largest integer h such that the graph contains at least h vertices with degree at least h (Eppstein & Spiro, 2012).

References

Agrawal, A., Krithika, R., Lokshtanov, D., Mouawad, A., & Ramanujan, M. S. (2018a). On the parameterized complexity of simultaneous deletion problems. *Proceedings of the 37th IARCS annual conference on foundations of software technology and theoretical computer science (FSTTCS '17)* (pp. 9:1–9:14). Leibniz International Proceedings in Informatics (LIPIcs), vol. 93. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

Agrawal, A., Lokshtanov, D., Mouawad, A., & Saurabh, S. (2018b). Simultaneous feedback vertex set: A parameterized perspective. *ACM Transactions on Computation Theory*, *10*(4), 18:1–18:25.

Bang-Jensen, J., & Gutin, G. (2009). *Digraphs: theory, algorithms and applications* (2nd ed.). Springer.

Berlingerio, M., Coscia, M., Giannotti, F., Monreale, A., & Pedreschi, D. (2013). Multidimensional networks: foundations of structural analysis. *Proceedings of the 22nd International World Wide Web Conference (WWW '13)*, *16*(5–6), 567–593.

van Bevern, R. (2014). Towards optimal and expressive kernelization for d-hitting set. *Algorithmica*, *70*(1), 129–147.

Boccaletti, S., Bianconi, G., Criado, R., del Genio, C. I., Gómez-Gardeñes, J., Romance, M., … Zanin, M. (2014). The structure and dynamics of multilayer networks. *Physics Reports*, *544*(1), 1–122.

Boden, B., Günnemann, S., Hoffmann, H., & Seidl, T. (2017). Mimag: mining coherent subgraphs in multi-layer graphs with edge labels. *Knowledge and Information Systems*, *50*(2), 417–446.

Borgatti, S. P., & Everett, M. G. (2000). Models of core/periphery structures. *Social Networks*, *21*(4), 375–395.

Brandstädt, A., Le, V. B., & Spinrad, J. P. (1999). *Graph classes: A survey*. SIAM Monographs on Discrete Mathematics and Applications, vol. 3. SIAM.

Bredereck, R., Komusiewicz, C., Kratsch, S., Molter, H., Niedermeier, R., & Sorge, M. (2017). Assessing the computational complexity of multi-layer subgraph detection. *Proceedings of the 10th international conference on algorithms and complexity (CIAC '17)* (vol. 10236, pp. 128–139). LNCS. Springer.

Bui-Xuan, B.-M., Habib, M., & Paul, C. (2008). Competitive graph searches. *Theoretical Computer Science*, *393*(1–3), 72–80.

Cai, L. (1996). Fixed-parameter tractability of graph modification problems for hereditary properties. *Information Processing Letters*, *58*(4), 171–176.

Cai, L., & Ye, J. (2014). Dual connectedness of edge-bicolored graphs and beyond. *Proceedings of the 39th international symposium on mathematical foundations of computer science (MFCS '14)* (vol. 8635., pp. 141–152). LNCS, Springer.

Chen, J., Molter, H., Sorge, M., & Suchý, O. (2018a). Cluster editing in multi-layer and temporal graphs. *Proceedings of the 29th international symposium on algorithms and computation (ISAAC '18)* (vol. 123, pp. 24:1–24:13). LIPIcs, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Chen, J., Niedermeier, R., & Skowron, P. (2018b). Stable marriage with multi-modal preferences. *Proceedings of the 2018 ACM conference on economics and computation (EC '18)* (pp. 269–286). ACM.

Cohen, J. (2008). Trusses: Cohesive subgraphs for social network analysis. *National Security Agency Technical Report*, 16.

Cygan, M., Fomin, F. V., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., … Saurabh, S. (2015). *Parameterized algorithms*. Springer.

Dell, H., & Marx, D. (2012). Kernelization of packing problems. *Proceedings of the 23rd annual ACM-SIAM symposium on discrete algorithms (SODA '12)* (pp. 68–81). SIAM.

Diestel, R. (2017). *Graph theory* (vol. 173, 5th ed.). Graduate Texts in Mathematics. Springer.

Dong, X., Frossard, P., Vandergheynst, P., & Nefedov, N. (2012). Clustering with multi-layer graphs: A spectral perspective. *IEEE Transactions on Signal Processing*, *60*(11), 5820–5831.

Dong, X., Frossard, P., Vandergheynst, P., & Nefedov, N. (2014). Clustering on multi-layer graphs via subspace analysis on Grassmann manifolds. *IEEE Transactions on Signal Processing*, *62*(4), 905–918.

Downey, R. G., & Fellows, M. R. (2013). *Fundamentals of parameterized complexity*. Springer.

Eppstein, D., & Spiro, E. S. (2012). The h-index of a graph and its application to dynamic subgraph statistics. *Journal of Graph Algorithms and Applications*, *16*(2), 543–567.

Erdös, P., & Rado, R. (1960). Intersection theorems for systems of sets. *Journal of the London Mathematical Society*, *35*(1), 85–90.

Fellows, M. R., Hermelin, D., Rosamond, F. A., & Vialette, S. (2009). On the parameterized complexity of multiple-interval graph problems. *Theoretical Computer Science*, *410*(1), 53–61.

Fertin, G., Komusiewicz, C., Mohamed-Babou, H., & Rusu, I. (2015). Finding supported paths in heterogeneous networks. *Algorithms*, *8*(4), 810–831.

https://doi.org/10.1017/nws.2019.13 Published online by Cambridge University Press

Flum, J., & Grohe, M. (2006). *Parameterized complexity theory*. Springer.
Gai, A.-T., Habib, M., Paul, C., & Raffinot, M. (2003). *Identifying common connected components of graphs*. Techical Report, RR-LIRMM-03016, LIRMM, Université de Montpellier II.
Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. Freeman.
Golumbic, M. C. (2004). *Algorithmic graph theory and perfect graphs* (vol. 57, 2nd ed.). Annals of Discrete Mathematics. Elsevier B. V.
Holme, P. (2015). Modern temporal network theory: a colloquium. *The European Physical Journal B*, *88*(9), 234.
Holme, P., & Saramäki, J. (2012). Temporal networks. *Physics Reports*, *519*(3), 97–125.
Jiang, D., & Pei, J. (2009). Mining frequent cross-graph quasi-cliques. *ACM Transactions on Knowledge Discovery from Data*, *2*(4).
Johnson, D. S. (1987). The NP-completeness column: An ongoing guide. *Journal of Algorithms*, *8*(3), 438–448.
Jukna, Stasys. (2011). *Extremal Combinatorics—with applications in computer science*. Texts in Theoretical Computer Science. An EATCS Series. Springer.
Kano, M., & Li, X. (2008). Monochromatic and heterochromatic subgraphs in edge-colored graphs—A survey. *Graphs and Combinatorics*, *24*(4), 237–263.
Khot, S., & Raman, V. (2002). Parameterized complexity of finding subgraphs with hereditary properties. *Theoretical Computer Science*, *289*(2), 997–1008.
Kim, J., & Lee, J. (2015). Community detection in multi-layer graphs: A survey. *SIGMOD Record*, *44*(3), 37–48.
Kivelä, M., Arenas, A., Barthelemy, M., Gleeson, J. P., Moreno, Y., & Porter, M. A. (2014). Multilayer networks. *Journal of Complex Networks*, *2*(3), 203–271.
Komusiewicz, C., & Niedermeier, R. (2012). New races in parameterized algorithmics. *Proceedings of the 37th international symposium on mathematical foundations of computer science (MFCS '12)* (vol. 7464). LNCS. Springer.
Kratsch, S. (2012). Polynomial kernelizations for MIN f+Π1 and MAX NP. *Algorithmica 63*(1–2), 532–550.
Latapy, M., Viard, T., & Magnien, C. (2018). Stream graphs and link streams for the modeling of interactions over time. *Social Network Analysis and Mining*, *8*(1), 61.
Lewis, J. M., & Yannakakis, M. (1980). The node-deletion problem for hereditary properties is NP-complete. *Journal of Computer and System Sciences*, *20*(2), 219–230.
Lin, B. (2015). The parameterized complexity of *k*-biclique. *Proceedings of the 26th annual ACM-SIAM symposium on discrete algorithms (SODA '15)* (pp. 605–615). SIAM.
Magnani, M., & Rossi, L. (2011). The ML-model for multi-layer social networks. *Proceedings of the international conference on advances in social networks analysis and mining (ASONAM '11)* (pp. 5–12). IEEE Computer Society.
Michail, O. (2016). An introduction to temporal graphs: An algorithmic perspective. *Internet Mathematics*, *12*(4), 239–280.
Monien, B. (1985). How to find long paths efficiently. *Annals of Discrete Mathematics*, *25*, 239–254.
Moser, H. (2009). *Finding optimal solutions for covering and matching problems*. Ph.D. thesis, Institut für Informatik, Friedrich-Schiller Universität Jena.
Mucha, P. J., Richardson, T., Macon, K., Porter, M. A., & Onnela, J.-P. (2010). Community structure in time-dependent, multiscale, and multiplex networks. *Science*, *328*(5980), 876–878.
Nastos, J., & Gao, Y. (2013). Familial groups in social networks. *Social Networks*, *35*(3), 439–450.
Nichterlein, A. (2014). *Degree-constrained editing of small-degree graphs*. Ph.d. thesis, TU Berlin.
Niedermeier, R. (2006). *Invitation to fixed-parameter algorithms*. Oxford University Press.
Niedermeier, R. (2010). Reflections on multivariate algorithmics and problem parameterization. *Proceedings of the 27th international symposium on theoretical aspects of computer science (STACS '10)* (vol. 5, pp. 17–32). Germany: LIPIcs IBFI Dagstuhl.
Plummer, M. D. (2007). Graph factors and factorization: 1985-2003: A survey. *Discrete Mathematics*, *307*(7–8), 791–821.
Rossi, L., Musolesi, M., & Torsello, A. (2015). On the *k*-anonymization of time-varying and multi-layer social graphs. *Proceedings of the 9th international conference on web and social media (ICWSM '15)* (pp. 377–386). AAAI Press.
Seidman, S. B. (1983). Network structure and minimum degree. *Social Networks*, *5*(3), 269–287.
Zeng, Z., Wang, J., Zhou, L., & Karypis, G. (2007). Out-of-core coherent closed quasi-clique mining from large dense graph databases. *ACM Transactions on Database Systems*, *32*(2), 13.

Appendix: Definitions

Asteroidal Triple-Free Graph An independent set of size three where each pair of vertices is joined by a path that avoids the neighborhood of the third is called an *asteroidal triple*. A graph is *asteroidal triple-free* if it does not contain asteroidal triples.

***c*-Colorable Graph** A graph is *c-colorable* if there is a way of coloring the vertices with at most c different colors such that no two adjacent vertices share the same color.

***c*-Connectivity** A graph is called *c-connected* if it contains at least $c + 1$ vertices, but does not contain a set of $c - 1$ vertices whose removal disconnects the graph.

***c*-Core** A graph is called a *c-core* if each vertex has degree at least c.

https://doi.org/10.1017/nws.2019.13 Published online by Cambridge University Press

c-**Edge-Connectivity** A graph is called *c-edge-connected* if it does not contain a set of $c-1$ edges whose removal disconnects the graph.

c-**Factor** A graph has a *c-factor* if it has a *c*-regular spanning graph.

c-**Regular Graph** A graph is called *c-regular* if every vertex has degree *c*.

c-**Truss** A graph is called a *c-truss* if it is connected and each edge is contained in at least $c-2$ triangles.

Chordal Graph A graph is called *chordal* if each induced cycle has at most three vertices.

Cluster Graph A graph is called a *cluster graph* if it is a collection of disjoint cliques.

Cograph A graph is called a *cograph* if it does not contain any induced path of length four.

Comparability Graph A graph is called a *comparability graph* if there is a partial order over the vertices such that each pair of vertices is adjacent if and only if it is comparable.

Complete Multipartite Graph A graph is called *complete multipartite* if the vertex set can be partitioned such that each vertex pair is adjacent if and only if the two vertices are in different partitions.

Edgeless Graph A graph is called *edgeless* is it does not contain any edges.

Forest A graph is called a *forest* if it is a collection of trees.

Hamiltonian Graph A graph is called *Hamiltonian* if it contains a *Hamiltonian path*, that is, a simple path that visits each vertex exactly once.

h-**Index** The *h-index* of a graph is the largest integer *h* such that the graph contains at least *h* vertices with degree at least *h*.

Interval Graph A graph is called an *interval graph* if an interval of the real numbers can be assigned to each vertex such that two vertices are adjacent if and only if the intervals overlap.

Line Graph A graph is called a *line graph* if there is another graph such that each vertex of the line graph corresponds to an edge of the other graph and two vertices in the line graph are adjacent if the corresponding edges in the other graph share a common endpoint.

Matching A graph has a *matching* if it has a 1-factor.

Perfect Graph A graph is called *perfect* if the chromatic number of every induced subgraph equals the size of the largest clique of that subgraph. The *chromatic number* of a graph is the smallest *c* such that the graph is *c*-colorable.

Permutation Graph A graph is called a *permutation graph* if the vertices represent elements in a permutation such that two vertices are adjacent if and only if the respective pair of elements is reversed by the permutation.

Planar Graph A graph is called *planar* if it can be embedded in the plane without edge-crossings; that is, it can be drawn on the plane such that its edges only intersect at their endpoints.

Quasi-Threshold Graph A graph is called a *quasi-threshold graph* if it does not contain any induced path of length four and any induced cycle of length four.

Split Graph A graph is called a *split graph* if its vertices can be partitioned into a clique and an independent set.

Star A graph is called a *star* if it is a tree with only one internal node.

Cite this article: Bredereck R., Komusiewicz S., Kratsch S., Molter H., Niedermeier R., and Sorge M. (2019). Assessing the computational complexity of multilayer subgraph detection. *Network Science* **7**, 215–241. https://doi.org/10.1017/nws.2019.13

https://doi.org/10.1017/nws.2019.13 Published online by Cambridge University Press