Technische Universität Berlin
Electrical Engineering and Computer Science
Institute of Software Engineering and Theoretical Computer Science
Algorithmics and Computational Complexity (AKT)

Vertex Cover Under Time Constraints

Bachelor Thesis

by Valentin Rohm

for the degree „Bachelor of Science“ (B. Sc.)
in the course of studies „Informatik“

Supervisor and Primary Reviewer:	Prof. Dr. Rolf Niedermeier
Secondary Reviewer:	Prof. Dr. Markus Brill
Co-Supervisors:	Till Fluschnik, Philipp Zschoche

November 2, 2018

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.
Die selbstständige und eigenhändige Ausfertigung versichert an Eides statt

Ort, Datum	Unterschrift

Zusammenfassung

Das Ziel dieser Arbeit ist es, die, insbesondere parametrisierte, Komplexität einer Verallgemeinerung des bekannten VERTEX COVER-Problems zu untersuchen: MULTISTAGE VERTEX COVER (MSVC). Ersteres fragt nach einer Teilmenge $V' \subseteq V$ (einer Knotenüberdeckung, engl. vertex cover), wobei $G = (V, E)$ ein ungerichteter Graph ist, sodass für jede Kante $e = \{u, v\} \in E$ mindestens einer der beiden Knoten u und v in V' enthalten ist. MULTISTAGE VERTEX COVER nimmt als Eingabe einen temporalen statt einem statischen Graphen entgegen. Ein temporaler Graph unterscheidet sich insofern von einem statischen Graphen, als dass er zwar eine feste Knotenmenge hat, seine Kantenmenge sich jedoch mit der Zeit ändert. Ein Graph der durch die Knotenmenge eines temporalen Graphen und die Kantenmenge zu einem bestimmten Zeitpunkt gegeben ist, wird als eine Stufe (engl. layer) des temporalen Graphen bezeichnet. MULTISTAGE VERTEX COVER fragt nach einer Knotenüberdeckung für jede Stufe, wobei bestimmte Beschränkungen beachtet werden müssen. Da sich die Topologie realer Netzwerke ebenfalls mit der Zeit ändert, könnten temporale Graphen und das MSVC-Problem ein sinnvoller Ansatz sein, solche Netzwerke und manche ihrer Anwendungen zu modellieren.

Wir zeigen, dass MSVC teilweise selbst unter solchen Beschränkungen NP-vollständig bleibt, die für das klassische VERTEX COVER eine effiziente Lösung in Linearzeit erlauben, wie z.B. Bäume als Eingabegraphen bzw. Stufen. Außerdem untersuchen wir die Möglichkeit einer Problemkernreduktion, wobei wir je nach Problemvariante sowohl nach Problemkernen mit polynomiell beschränkter Größe suchen, als auch untere Schranken für die Größe der Problemkerne angeben. Zu diesem Zweck greifen wir auf das Konzept der cross-composition zurück. Wir entwickeln einen FPT-Algorithmus für den Parameter k mit Laufzeit $\mathcal{O}(\tau \cdot (2^{2k^2+5k} \cdot k^2) + |V|^3)$, sowie Varianten des Algorithmus, die entweder die Berechnung einer konkreten Lösung erlauben, oder die Speicherkomplexität des Algorithmus senken. Kurz behandeln wir auch eine Variante von MSVC, MSVC-Sum. Der einzige Unterschied zwischen den beiden Problemvarianten ist, dass bei MSVC-Sum die Summe der symmetrischen Differenzen $S_1 \triangle S_2, ..., S_{\tau-1} \triangle S_\tau$ durch ℓ nach oben beschränkt wird, statt den einzelnen Differenzen. Wir zeigen, dass MSVC und MSVC-Sum meist ähnliche Eigenschaften bezüglich ihrer Komplexität haben.

Abstract

The aim of this thesis is to study the computational and in particular parameterized complexity of a generalization of the well known Vertex Cover problem, Multistage Vertex Cover (MSVC). Vertex Cover asks to find a subset (a vertex cover) $V' \subseteq V$ where $G = (V, E)$ is an undirected graph, such that for every edge $e = \{u, v\} \in E$, u or v is contained in V'. For MSVC, the input is a temporal instead of a static graph. A temporal graph is different from a static graph in the sense that while the vertex set is fixed, the edge set varies with time. The graph given by the vertex set and the edge set for each time step is called a layer. The Multistage Vertex Cover problem asks to find a vertex cover for each layer while meeting certain constraints. Because the topology of real-world networks also varies with time, temporal graphs and the MSVC problem could be a good way to model certain real-world applications.

We prove that MSVC remains NP-complete even under constraints such as the layers being trees. Notably, classic Vertex Cover is trivially linear-time solvable on trees. Further, we provide polynomial kernels and lower bounds for kernelizations. For this purpose, the cross-composition framework is employed. For the parameter k, we develop an FPT algorithm running in $\mathcal{O}(\tau \cdot (2^{2k^2+5k} \cdot k^2) + |V|^3)$ time, including optimizations for space complexity and the ability to extract a specific solution. A variant of MSVC, MSVC-Sum, is covered briefly. The only difference between MSVC and MSVC-Sum is that the sum of the symmetric differences $S_1 \triangle S_2, ..., S_{\tau-1} \triangle S_\tau$ is upper-bounded in ℓ, rather than the individual symmetric differences. We show that MSVC-Sum has similar characteristics with respect to computational complexity in most cases.

Contents

1 Introduction

VERTEX COVER is a well-known NP-complete graph problem. It asks to find a subset $V' \subseteq V$ where $G = (V, E)$ is an undirected graph, such that for every edge $e = (u, v) \in E$, u or v is contained in V'. The VERTEX COVER problem has many real-world applications, including but not limited to computational biology [Hua] and network security [Fil+07]. Filiol et al. [Fil+07] have studied worm propagation in computer networks. They describe a scenario where after an initial infection, the attacker continues to communicate with the worm in order to update it. To reduce the communication overhead, the attacker only sends the updates to a few "privileged" nodes, which are able to efficiently communicate with all other nodes, and distribute the update. The defender too has an interest in finding such "privileged" nodes, because knowing the potential weak points of the network may help identify a worm attack more quickly. The "privileged" nodes can be modeled as a vertex cover set of the network graph where each edge represents a communication link. However, some modern worms such as the famous Stuxnet are known to remain dormant in the network for several weeks between attacks [**mueller2012stuxnet**], and the topology of real-world networks is always subject to change over time. For this reason, we are required to introduce a notion of time to both graphs and the VERTEX COVER problem: In this thesis, we will study a variation of VERTEX COVER, MULTISTAGE VERTEX COVER (MSVC). MSVC asks to maintain a vertex cover of a temporal graph (formally defined in Definition 2.2) in every time step, while only limited adjustments between time steps are allowed. We will study the computational and parameterized complexity of MSVC under various constraints such as the graph being a tree in every layer, or every layer containing just a single edge (see Table 1.1). We will also study the possibility of polynomial kernelizations (see Table 1.2) and develop a parameterized algorithm (Algorithm 3.42) which allows us to solve the problem efficiently for low values of the parameter k, the size of the desired vertex cover set.

1.1 Related Work

As VERTEX COVER is a classic NP-complete problem, lots of research has been done regarding its complexity and that of its variations. For instance, Chen, Kanj, and Xia [CKX06] provide a parameterized polynomial-space algorithm that decides an instance of VERTEX COVER in $\mathcal{O}((1.2738^k + kn))$, which is currently the fastest known algorithm for this problem. Buss and Goldsmith [BG93], Chen, Kanj, and Jia [CKJ01] and Downey, Fellows, and Stege [DFS99] provide us with studies on polynomial-time data reduction rules and polynomial kernels, where the lowest known upper bound is $\mathcal{O}(k^2)$. Other

Table 1.1: Complexity classes and best known running times for MSVC, MSVC-Sum and their variants.

	parameters / constraints	*MSVC*	*MSVC-Sum*
Arbitrary layers	k	FPT (Thm. 3.38)	FPT (Conjecture 4.16)
	τ	para-NP-hard (Thm. 3.8)	para-NP-hard (Thm. 4.3)
	τ or ℓ are fixed constants	NP-complete (Thm. 3.1)	NP-complete(Thm. 4.1)
	$\ell \geq 2k$	FPT(Thm. 3.35)	N/A
	$\ell \geq 2k \cdot (\tau - 1)$	N/A	FPT(Thm. 4.14)
	$k \geq \|V\|$	yes-instance (Lemma 3.19)	yes-instance (??)
Every layer is a tree	k	FPT (Thm. 3.38)	FPT (Conjecture 4.16)
	τ	para-NP-hard (Thm. 3.8)	para-NP-hard(Thm. 4.3)
	no constraint	NP-complete (Thm. 3.1)	NP-complete(Thm. 4.1)
	$\ell = 0$	NP-complete (Thm. 3.1)	NP-complete(Thm. 4.1)
	$\tau = 1$	Polynomial (Thm. 3.20)	Polynomial (Thm. 4.2)
	$\ell \geq 2k$	Polynomial (Thm. 3.20)	N/A
	$\ell \geq 2k \cdot (\tau - 1)$	N/A	Polynomial (Thm. 4.2)
$\|E_i\| = 1$ for all layers G_i	k	FPT (Thm. 3.38)	FPT (Conjecture 4.16)
	τ	FPT (Thm. 3.34)	FPT (Thm. 4.15)
	no constraint	NP-complete (Thm. 3.1)	NP-complete (Thm. 4.1)
	$k \geq \tau$	yes-instance (Thm. 3.17)	yes-instance (Thm. 4.4)
	$\ell \leq 1$	NP-complete (Thm. 3.1)	N/A
	$\ell \geq 2$	yes-instance (Thm. 3.17)	N/A
	$\ell \geq 2 \cdot (\tau - 1)$	N/A	yes-instance (Thm. 4.4)

Table 1.2: Smallest known kernel sizes for MSVC, MSVC-Sum and their variants (PK stands for polynomial kernel).

	parameters / constraints	*MSVC*	*MSVC-Sum*
k	$\ell = 0$	$\mathcal{O}(k^2)$** (Thm. 3.24)	$\mathcal{O}(k^2)$** (Thm. 4.5)
	$\ell > 0$	no PK* (Thm. 3.25)	N/A
	$2k \cdot (\tau - 1) > \ell > 0$	N/A	no PK* (Observation 4.9)
	$\ell \geq 2k \cdot (\tau - 1)$	N/A	no PK*(Thm. 4.6)
	$\|E_i\| = 1$ for all layers G_i and $\ell = 0$	no PK* (Thm. 3.25)	N/A
$k + \tau$	no constraint	$3k^2 \cdot \tau$ (Thm. 3.30)	$3k^2 \cdot \tau$ (Thm. 4.10)
	$\ell \geq 2k$	$\mathcal{O}(k^2 \cdot \tau)$****(Thm. 3.32)	N/A
	$\ell \geq 2k \cdot (\tau - 1)$	N/A	$\mathcal{O}(k^2 \cdot \tau)$****(Thm. 4.11)
τ	Arbitrary layers	no kernelization** (Lemma 3.33)	no kernelization** (Lemma 4.13)
	$\|E_i\| = 1$ for all layers G_i	$5\tau + 1$ (Thm. 3.34)	$7\tau - 2$ (Thm. 4.12)

*Assuming that NP $\not\subseteq$ coNP / poly.
**Assuming that $P \neq NP$
'***Bikernel
****Turing kernelization

studies on kernelization of Vertex Cover include Abu-Khzam et al. [AK+04], who introduce the technique of *crown reduction*. Guo, Niedermeier, and Wernicke [GNW05] have studied problem generalizations such as Connected Vertex Cover, Capacitated Vertex Cover, and Maximum Partial Vertex Cover with respect to fixed-parameter tractability. Very little research has been done on variations of Vertex Cover on temporal graphs, such as MSVC. A notable exception is Akrida et al. [Akr+18], who studied a problem called Temporal Vertex Cover (TVC). The fundamental difference to MSVC is that Temporal Vertex Cover asks for every edge being covered only once throughout the lifetime of the temporal graph, or at least once in every Δ consecutive time steps, where Δ is a natural number, in case of the sliding window variation. Akrida et al. [Akr+18] provide hardness results, approximations and algorithms for TVC.

Michail [Mic16] give an introduction to temporal graphs in general. There are some studies on temporal generalizations of graph theory problems: Wu et al. [Wu+14] have studied path problems on temporal graphs. The temporal notion allows for new problem definitions such as Earliest-Arrival Path, Latest-Departure Path and Fastest Path. Other examples of (NP-hard) problems on temporal graphs are given by Michail and Spirakis [MS16], who studied problems such as Matching and Traveling Salesman on temporal graphs, and **zschoche2017computational**, who studied the problem of finding separators in temporal graphs. A separator is a vertex set such that if it is removed, all temporal paths between two designated terminal vertices are eliminated. There is plenty of work available on fixed-parameter tractability and kernelization in general, for example the studies by Downey and Fellows [DF95] and Bodlaender, Jansen, and Kratsch [BJK14], the latter of which introduce the cross-composition framework, which can be used to establish lower bounds for the size of kernels.

2 Preliminaries

The following chapter includes the most important basic definitions which are used throughout this thesis. In some obvious cases, such as edges and vertices, we will adhere to the well-known definitions given by textbooks such as [VLL90].

2.1 Problem Definitions

Definition 2.1. (parameterized problem) A *parameterized problem* is a language $L \subseteq \Sigma^\star \times \mathbb{N}$ where Σ is a finite alphabet (we assume that $0 \in \mathbb{N}$). The second element is called the *parameter* of the problem.

Vertex Cover
Input: An undirected graph $G = (V, E)$ and an integer $k \in \mathbb{N}$.
Question: Is there a subset $V' \subseteq V$ with $|V'| \leq k$ (also referred to as a vertex cover of size (at most) k) such that for every edge $e = \{u, v\} \in E \Rightarrow u \in V' \vee v \in V'$?

If such a subset V' exists, then it is referred to as a *solution* to the given Vertex Cover instance.

Unlike static graphs, temporal graphs do not have a static edge set, but each edge of a temporal graph has a time stamp. As a prerequisite to MSVC, we provide formal definitions for temporal graphs and some related terms.

Definition 2.2. (temporal edge, temporal graph and layers) Let V be a vertex set and $\tau \in \mathbb{N}$, then a *temporal edge* on V and τ is a pair $e = (\{v, w\}, t)$ where $v, w \in V$ and $1 \leq t \leq \tau \in \mathbb{N}$. A *temporal graph* is a triple $(V, \mathcal{E}, \tau)$ where V is a vertex set, $\mathcal{E}$ is a set of temporal edges on V and $\tau \in \mathbb{N}$. Let $\mathcal{G} = (V, \mathcal{E}, \tau)$ be a temporal graph and $E_i = \{\{v, w\} \mid (\{v, w\}, i) \in \mathcal{E}\}$ for all $i \in \{1, ..., \tau\}$, then $G_1 = (V, E_1), ..., G_\tau = (V, E_\tau)$ are the *layers* of $\mathcal{G}$.

As stated in the introduction, MSVC asks to find a vertex cover set in every time step, where only limited adjustments between time steps are possible. As a notion of such adjustments, we use the *symmetric difference*. The *symmetric difference* $S \triangle T = (S \cup T) \setminus (S \cap T)$ of two sets S and T is the set of elements that are contained in either S or T but not both.

Multistage Vertex Cover (MSVC)
Input: A temporal graph $\mathcal{G} = (V, \mathcal{E}, \tau)$ and two integers k, $\ell \in \mathbb{N}$.
Question: Is there a sequence $\mathcal{S} = (S_1, ..., S_\tau)$ with $S_i \subseteq V$ for every $i \in \{1, ..., \tau\}$ such that

- S_i is a vertex cover of size at most k of G_i for every $i \in \{1, ..., \tau\}$, and
- $|S_i \triangle S_{i+1}| \leq \ell$ for every $i \in \{1, ..., \tau - 1\}$?

If such a sequence $\mathcal{S}$ exists, then it is referred to as a *solution* to the given MSVC instance.

Next, we introduce a variation of MSVC, MSVC-Sum. The difference lies in the function of the parameter ℓ: In MSVC, it is an upper bound to the cardinality of the symmetric difference between two consecutive vertex covers in a solution. In MSVC-Sum, it is an upper bound to the sum of all such cardinalities.

Multistage Vertex Cover Sum (MSVC-Sum)
Input: A temporal graph $\mathcal{G} = (V, \mathcal{E}, \tau)$ and two integers k, $\ell \in \mathbb{N}$.
Question: Is there a sequence $\mathcal{S} = (S_1, ..., S_\tau)$ with $S_i \subseteq V$ for every $i \in \{1, ..., \tau\}$ such that

- S_i is a vertex cover of size at most k of G_i for every $i \in \{1, ..., \tau\}$, and
- $\sum_{i=1}^{\tau-1} |S_i \triangle S_{i+1}| \leq \ell$?

If such a sequence $\mathcal{S}$ exists, then it is referred to as a *solution* to the given MSVC-Sum instance.

Definition 2.3. (yes-instance and no-instance) A tuple containing all necessary inputs as specified in the definition of a decision problem is an *instance* of the respective problem. If for an instance I the question of a decision problem can be answered with “yes”, then I is a *yes-instance* of the problem, otherwise it is a *no-instance*.

Definition 2.4. (fixed-parameter tractability) A parameterized problem P with parameter k is *fixed-parameter tractable* if there is an algorithm that can decide whether an instance I of P is a yes-instance or a no-instance in time $f(k) \cdot |I|^{\mathcal{O}(1)}$ where f is an arbitrary computable function depending only on k.

2.2 Auxiliary Definitions

For later reference we provide auxiliary definitions, which are specific to this thesis and the MSVC problem.

Definition 2.5. (covered edges) Let V be a set of vertices and let E be a set of undirected edges on V. An edge $e = \{u, v\}$ is *covered* by a subset $V' \subseteq V$ if u or v is contained in V'. If $V = \{v\}$, then e is covered by v. A temporal edge $e_t = (e, t)$ is covered by a sequence $\mathcal{S} = (S_1, ..., S_\tau)$ of vertex sets if e is covered by S_t.

Definition 2.6. (delete and add actions) Let $(\mathcal{G}, k, \ell)$ be an instance of MSVC, and let $\mathcal{S} = (S_1, ..., S_\tau)$ be a solution. If $v \in S_i$ and $v \notin S_{i+1}$ for $v \in V$ and $1 \leq i \leq \tau - 1$, then v has been *deleted* in layer G_{i+1} (a delete action has been performed on v in solution $\mathcal{S}$ at layer G_{i+1}). If $v \notin S_i$ and $v \in S_{i+1}$ for $v \in V$ and $1 \leq i \leq \tau - 1$, then v has been *added* in layer G_{i+1} (an add action has been performed on v in solution $\mathcal{S}$ at layer G_{i+1}).

Definition 2.7. (optimal actions) Let $I = (\mathcal{G}, k, \ell)$ be an instance of MSVC. An action is optimal for I if I is a no-instance, or if there is a solution to I in which the action is performed.

Definition 2.8. (initial configuration) Let $(\mathcal{G}, k, \ell)$ be an instance of MSVC, and let $\mathcal{S} = (S_1, ..., S_\tau)$ be a solution. Then S_1 is the *initial configuration* of $\mathcal{S}$.

2.3 Kernelization and Cross-Composition

It is often possible to preprocess an instance of a parameterized problem in order to obtain an equivalent, but smaller instance, which subsequently saves running time when solving it. Next, two approaches to this problem, kernels and bikernels, are formally defined. In a later section, we will provide both polynomial kernels and lower bounds for kernelization.

Definition 2.9. (kernel and kernelization) A *kernelization* of a parameterized (decision) problem with input x and parameter k is a polynomial-time algorithm that takes an instance $I = (x, k)$ as its only argument and returns an instance $K = (x', k')$ of the same parameterized problem with parameter k' such that K is a yes-instance if and only if I is a yes-instance and $|x'| + k' \leq f(k)$ for a computable function $f : \mathbb{N} \to \mathbb{N}$. The instance K is called a *kernel* of I, and $f(k)$ is its *size*. If $f(k) \leq k^c$ for a non-negative integer c, then I has a *polynomial kernel.*

In the case where K is an instance of a different parameterized problem than I, we refer to K as a *bikernel.*

The next two definitions are part of the Cross-Composition framework by Bodlaender, Jansen, and Kratsch [BJK14]. It is used to rule out polynomial kernels for a parameterized problem.

Definition 2.10. (polynomial equivalence relation — based on [BJK14]) An equivalence relation $\mathcal{R}$ on $\Sigma^\star$ is called a *polynomial equivalence relation* if the following two conditions hold:

- There is an algorithm that, given $x, y \in S$, decides whether x and y belong to the same equivalence class in time polynomial in $|x| + |y|$.

- For any finite set $S \subseteq \Sigma^\star$ the equivalence relation $\mathcal{R}$ partitions the elements of S into a number of classes that is polynomially upper-bounded in the size of the largest element of S.

Definition 2.11. (AND-cross-composition — based on [BJK14]) Let $L \subseteq \Sigma^\star$ be a language, let $\mathcal{R}$ be a polynomial equivalence relation on $\Sigma^\star$, and let $Q \subseteq \Sigma^\star \times \mathbb{N}$ be a parameterized problem. An *AND-cross-composition* of L into Q (with respect to $\mathcal{R}$) is an algorithm that takes t instances $I_1, I_2, ..., I_t \in \Sigma^\star$ of L which belong to the same equivalence class of $\mathcal{R}$ as its input, takes time polynomial in $\sum_{i=1}^{t} |I_i|$, and returns an instance $I = (y, k) \in \Sigma^\star \times \mathbb{N}$ of Q such that the following two conditions hold:

- The parameter k is polynomially bounded in $\max_i |I_i| + \log t$
- The instance I is a yes-instance of Q if and only if I_i is a yes-instance of Q for every $i \in \{1, ..., t\}$.

3 Multistage Vertex Cover (MSVC)

3.1 Hardness

In this section, we investigate MSVC with respect to hardness under various constraints. In particular, we will treat the case where every layer is a tree, and the case where every layer contains just a single edge. In some cases, we can prove NP-hardness for a constant value of a parameter, which implies para-NP-hardness.

3.1.1 NP-hard cases

Theorem 3.1. *MSVC is NP-complete even if*

- $\tau \geq 1$ *is any fixed constant,*
- $\ell \geq 0$ *is any fixed constant,*
- *every layer* G_i *is a tree and* $\ell = 0$,
- *or* $|E_i| = 1$ *for every layer* $G_i = (V, E_i)$ *and* $\ell \leq 1$.

In order to prove Theorem 3.1, we first have to prove that MSVC is contained in NP. Next, we provide polynomial-time many-one reductions to prove NP-hardness for MSVC under different constraints.

Lemma 3.2. *MSVC is contained in NP.*

Proof. Given an MSVC instance I and a solution $\mathcal{S} = (S_1, ..., S_\tau)$, we can verify if $\mathcal{S}$ is a solution to I in polynomial time. We have to prove that:

- S_i is a vertex cover of size at most k of G_i for every $i \in \{1, ..., \tau\}$;
- $|S_i \triangle S_{i+1}| \leq \ell$ for every $i \in \{1, ..., \tau - 1\}$.

We can verify that S_i is a vertex cover of size at most k of G_i for every $i \in \{1, ..., \tau\}$ in polynomial time because Vertex Cover is contained in NP. Moreover, $|S_i \triangle S_{i+1}| \leq \ell$ for every $i \in \{1, ..., \tau - 1\}$ can be verified in polynomial time because each symmetric difference can be computed in polynomial time, and a total of $\tau - 1$ symmetric differences have to be computed. □

Lemma 3.3. *There is a polynomial-time many-one reduction from* Vertex Cover *to MSVC where* ℓ *is any fixed constant.*

Proof. Let $I = (G = (V, E), k)$ be an instance of Vertex Cover and let $c \in \mathbb{N}$ be an arbitrary but fixed constant. We construct the MSVC instance as follows: Set $V' = V$, $\mathcal{E} = \{(\{v, w\}, 1) \mid \{v, w\} \in E\}, \mathcal{G}' = (V, \mathcal{E}, 1)$, $k' = k$ and $\ell' = c$. All of these operations can be performed in polynomial time. Now $I' = (\mathcal{G}', k', \ell')$ is an instance of MSVC. We have to prove that I is a yes-instance of Vertex Cover if and only if I' is a yes-instance of MSVC. Assume that I is a yes-instance of Vertex Cover, then there is a subset $V' \subseteq V$ with $|V'| \leq k$ so that V' is a vertex cover of size at most $k = k'$. Set $S_1 = V'$. We do not have to verify the second condition because $\mathcal{G}'$ has only one layer. Assume I' is a yes-instance of Vertex Cover, then the only layer G_1 has a vertex cover S_1 of size at most $k' = k$. Because $G_1 = G$, S_1 is also a vertex cover of size at most k of G. □

Lemma 3.4. *There is a polynomial-time many-one reduction from* Vertex Cover *to MSVC where τ is any fixed constant.*

Proof. Let $I = (G = (V, E), k)$ be an instance of vertex cover. We construct the MSVC instance as follows: Set $V' = V$, $\tau = c$ (can be an arbitrary constant), $\mathcal{E} = \{(\{v, w\}, 1) \mid \{v, w\} \in E\} \cup \{(\{v, w\}, 2) \mid \{v, w\} \in E\} \cup ... \cup \{(\{v, w\}, \tau) \mid \{v, w\} \in E\}$, $\mathcal{G}' = (V, \mathcal{E}, \tau)$, $k' = k$ and $\ell' = 0$. All of these operations can be performed in polynomial time. Now $I' = (\mathcal{G}', k', \ell')$ is an instance of MSVC. We have to prove that I is a yes-instance of Vertex Cover if and only if I' is a yes-instance of MSVC. Assume that I is a yes-instance of Vertex Cover, then there is a subset $V' \subseteq V$ with $|V'| \leq k$ so that V' is a vertex cover of size at most $k = k'$. Set $S_1 = S_2 = ... = S_\tau = V'$. By doing this we also get $|S_i \triangle S_{i+1}| = 0 \leq \ell$ for every $i \in \{1, ..., \tau - 1\}$. Assume that I' is a yes-instance of Vertex Cover, then the set S_1 (and S_2, ..., S_τ) is a vertex cover of size at most $k' = k$ of G_1. Because $G_1 = G$, S_1 is also a vertex cover of size at most k of G. □

Lemma 3.5. *There is a polynomial-time many-one reduction from* Vertex Cover *to MSVC with $|E_i| = 1$ for all layers G_i and $\ell = 0$.*

Proof. Let $I = (G, k)$ be an instance of vertex cover. We construct the MSVC instance $I' = (\mathcal{G}, k', \ell)$ as follows: I and I' share the same vertex set V. For each edge in the Vertex Cover instance, the MSVC instance receives a layer containing only the said edge, so $\tau = |E|$. We set $k' = k$, and $\ell = 0$. All of these operations can be performed in polynomial time.

If the vertex cover instance is a yes-instance, a set S of at most k vertices exists that covers all edges of G. Then $\mathcal{S} = (S_1, ..., S_\tau)$ where $S_i = S$ for every $i \in \{1, ..., \tau\}$ is a solution to the MSVC instance, because by covering each edge of the original graph G, S also covers the corresponding layers of I'.

If I' is a yes-instance, a set of at most k vertices S and a solution $\mathcal{S} = (S_1, ..., S_\tau)$ exists such that $S_1 = \ldots = S_\tau = S$ because $\ell = 0$. As S is a solution to I', it covers all of its layers. Then S is also a solution to the Vertex Cover instance, because by covering each layer of I' it also covers the corresponding edges of G. □

Lemma 3.6. *There is a polynomial-time many-one reduction from MSVC with $|E_i| = 1$ for all layers G_i and $\ell = 0$ to MSVC $|E_i| = 1$ for all layers G_i and $\ell = 1$.*

Proof. Let $I = (\mathcal{G}, k, l)$ be an instance of MSVC with $|E_i| = 1$ for all layers G_i and $\ell = 0$. We construct the instance $I' = (\mathcal{G}' = (V', \mathcal{E}', \tau'), k', \ell')$ as follows: We set $V' = V \cup \{v'_1, ..., v'_{2\tau}\}$ and $\mathcal{E}' = \{(e, \tau) \mid (e, \tau/2) \in \mathcal{E}\} \cup F$ with $F = \{(\{v'_i, v'_{i+\tau}\}, 1 + (2(i-1))\}$ for every $i \in \{1, ..., \tau\}$, $\tau' = 2\tau$, $k' = k + 1$ and $\ell = 1$. Intuitively speaking, by doing this we add 2τ additional vertices and τ additional temporal edges such that every new vertex is connected to exactly one other new vertex and none of the old ones. Each of the new edges is featured in exactly one layer. The layers featuring the new edges all have odd indexes, the original layers have even indexes. The total amount of layers is increased to 2τ. All of these operations can be performed in polynomial time.

If I is a yes-instance with a solution $\mathcal{S} = (S_1, ..., S_\tau)$ (with $S_1 = \ldots = S_\tau$ because $\ell = 0$), then we can construct a solution $\mathcal{S}' = (S'_1, ..., S'_{2\tau})$ to I' as follows: Set $S'_i = S_i$ for every $i \in \{2, 4, ..., 2\tau\}$ and $S'_j = S_j \cup \{v'_{(j+1)/2} \mid j \in \{1, 3, ...2\tau - 1\}$. The sequence $\mathcal{S}$ is a solution to I' because

- $S'_k = S_k$ and $E'_k = E_k$ for every $k \in \{2, 4, ...2\tau\}$ and thus S'_k is a vertex cover of G'_k. $S'_k = S_k \cup \{v'_{(j+1)/2} \mid k \in \{1, 3, ..., 2\tau - 1\}\}$ and G'_k only contains one additional edge when compared to G_k, which is covered by $v'_{(j+1)/2}$. Thus S'_k is a vertex cover of G'_k.
- $|S'_i \triangle S'_{i+1}| \leq \ell$ for every $i \in \{1, 3, ..., \tau - 1\}$ holds because $|S_i| = k$ for layers with an even index and $|S_i| = k + 1$ for layers with an odd index.

If I' is a yes-instance, then its solution $\mathcal{S}$ uses a maximum of $k + \tau$ different vertices: To prove this, we try to construct a solution that uses as many different vertices as possible: We can select at most $k + 1$ different vertices for the initial configuration (S'_1). Using a smaller initial configuration does not allow us to use a higher amount of different vertices because one would have to add vertices to reach even $k + 1$. The next action (in layer 2) has to be a delete because we cannot add any more vertices. After that, we can add and delete vertices alternatingly. There is no way to use a higher amount of vertices than using this method because we add a vertex whenever possible and delete when we cannot add any more. By doing this we can add $(\tau' - 1)/2 = (2\tau - 1)/2 = \tau - 1$ further vertices. This leaves us with a maximum of $(k + 1) + (\tau - 1) = k + \tau$ different vertices. We know that throughout the MSVC we have to use exactly τ vertices to cover the newly added layers (with odd indices), because the τ contained edges are mutually disjoint. Also, none of these edges are connected to the original graph, so the remaining k vertices have to be sufficient to cover the original layers. This implies that the input $(\mathcal{G}, k, \ell)$ is a yes-instance. □

Lemma 3.7. *There is a polynomial time many-one reduction from* VERTEX COVER *to MSVC where every layer is a tree and* $\ell = 0$.

Proof. Let $I = (G = (V, E), k)$ be an instance of VERTEX COVER. We construct a set of spanning trees such that every edge $e \in E$ is contained in the edge set of at least one of the spanning trees using the following algorithm: Let T_1 be an arbitrary spanning tree of G (a spanning tree can be found in polynomial time using Breadth-First-Search) [VLL90]. Given $i \in \mathbb{N}$ spanning trees $T_1 = (V, E_1)$, ..., $T_i = (V, E_i)$ (in the first iteration of the algorithm we only have the first spanning tree T_1), we compute $T_{i+1} = (V, E_{i+1})$ as follows: We select an arbitrary edge $e = \{u, v\} \in E$ such that $e \notin E_j$ for every $j \in \{1, ..., i\}$. If no such e exists, then we are done, as each edge of the graph G is contained in at least one of the spanning trees $T_1, ..., T_i$. Otherwise, we add this edge to E_{i+1} and build the remaining spanning tree using Breadth-First-Search by starting the search from u and inserting v into the BFS queue first (which forces the algorithm to discover v right after u) so that $\{u, v\} = e$ is contained in the resulting spanning tree. All of these operations can be performed in polynomial time: A single spanning tree can be found in polynomial time, and at most $|E|$ spanning trees are computed: G can have at most $|E|$ edges and T_i always contains at least one edge that is not contained in any T_j for $j \in \{1, ..., i-1\}$. Therefore, after at most $|E|$ iterations, the algorithm terminates, after the final spanning $E_{t'}$ tree has been computed. An instance $I' = (\mathcal{G}, k', \ell)$ is constructed by setting $\mathcal{G} = (V', \mathcal{E}, \tau)$ with $V' = V$, $\mathcal{E} = \{(\{u, v\}, t) | \{u, v\} \in E_t, t \in \{1, ..., t'\}\}$ and $\tau = t'$, $k' = k$ and $\ell = 0$.

We now prove that the constructed instance I' is a yes-instance of MSVC if and only I is a yes-instance of MSVC: Assume that I is a yes-instance of Vertex Cover, then there is a subset $V' \subseteq V$ with $|V'| \leq k$ that covers all edges of G. Because we only added temporal edges $(\{u, v\}, t)$ to $\mathcal{G}$ such that $\{u, v\}$ is contained in E and therefore covered by V', V' is also a vertex cover of size at most k for every layer of $\mathcal{G}$. Let $\mathcal{S} = (S_1, ..., S_\tau)$ with $S_1 = \ldots = S_\tau = V'$. We get $|S_i \triangle S_{i+1}| = 0 \leq \ell$ for every $i \in \{1, ..., \tau - 1\}$, which implies that I' is a yes-instance of MSVC. Assume that I' is a yes-instance of MSVC. Because $\ell = 0$ we have $S_1 = \ldots = S_\tau = S$ for any solution $\mathcal{S} = S_1, ..., S_\tau$. Such vertex set S covers every edge of every layer, and by construction every edge of G is contained in at least one of the layers. Therefore, S is a vertex cover of G, and I is a yes-instance. □

Proof of Theorem 3.1. There are polynomial-time many-one reductions from Vertex Cover to MSVC with arbitrary values of τ (Lemma 3.4) and ℓ (Lemma 3.3), to MSVC with $|E_i| = 1$ for all layers G_i and $\ell = 0$ (Lemma 3.5), and to MSVC where each layer G_i is a tree and $\ell = 0$ (Lemma 3.7), which proves NP-hardness for these problem variants. There is another polynomial-time many-one reduction from MSVC with $|E_i| = 1$ for all layers G_i and $\ell = 0$ to MSVC with $|E_i| = 1$ for all layers G_i and $\ell = 1$ (Lemma 3.6), which proves NP-hardness for the latter. MSVC is contained in NP (Lemma 3.2), therefore all of its variants which were proven to be NP-hard are also NP-complete. □

Notably, we were able to prove NP-hardness even if every layer is a tree, or every layer contains just a single edge, even though trees and single edges are tractable cases for the classic Vertex Cover problem.

3.1.2 para-NP-hard cases

Next, we will prove that some variants of MSVC are NP-hard even for constant values of certain parameters, and thus para-NP-hard. This rules out kernelizations and fixed-parameter algorithms, which will be studied in depth later.

We know that Vertex Cover (which is equivalent to MSVC with only one layer) can be solved efficiently on trees, and MSVC generally - for arbitrary values of τ - can not. We prove that MSVC remains NP-complete in the intermediate case where $\tau = 2$.

Theorem 3.8. *MSVC is para-NP-hard for the parameter τ even if every layer is a tree.*

To prove Theorem 3.8, we provide three different polynomial-time many-one reductions, which we can compose to receive a polynomial-time many-one reduction from a variation of the NP-hard Independent Set to MSVC on trees where $\tau = 2$. As an intermediate step, we reduce from Independent Set to Hamiltonian Cycle, a formal definition of which is presented below.

Hamiltonian cycle
Input: An undirected graph $G = (V, E)$
Question: Is there a cycle $H = (V, E_H)$ with $E_H \subseteq E$ in G such that H visits every vertex of V exactly once?

If such a cycle exists, then it is referred to as a *Hamiltonian cycle* of G.

Lemma 3.9. *There is a polynomial-time many-one reduction from* Independent Set *on cubic Hamiltonian graphs to* Vertex Cover *on cubic Hamiltonian graphs.*

Proof. We reduce from the NP-hard Independent Set on cubic Hamiltonian (planar) graphs [FSS10] to Vertex Cover on cubic Hamiltonian graphs by using the well-known reduction which maps an instance $I = (G = (V, E), k)$ of Independent Set to an instance $I' = (G = (V, E), |V| - k)$ of Vertex Cover [Kar72]. This reduction does not change the structure of the graph G. Hence, Vertex Cover remains NP-complete on cubic Hamiltonian graphs. □

Fleischner, Sabidussi, and Sarvanov [FSS10] assume that, when speaking of a Hamiltonian graph $G = (V, E)$, a specific Hamiltonian cycle $H = (V, E_H)$ with $E_H \subseteq E$ is given, and prove NP-completeness even under this condition. We will maintain this assumption and use H in the next reduction.

Lemma 3.10. *There is a polynomial-time many-one reduction from* Vertex Cover *on cubic Hamiltonian graphs to MSVC where every layer is a forest, $\tau = 2$ and $\ell = 0$.*

Proof. Let $I = (G = (V, E), H, k)$ be an instance of Vertex Cover such that G is a cubic Hamiltonian graph with Hamiltonian cycle $H = (V, E_H)$ where $E_H \subseteq E$. We construct an instance $I' = (\mathcal{G}, k', \ell)$ of MSVC, where $\mathcal{G}$ has the layers G_1 and G_2, as follows: Let $e \in E_H$ be an arbitrary edge of the Hamiltonian cycle H. We set $G_1 = (V, E_H \setminus \{e\})$, $G_2 = (V, (E \setminus E_H) \cup \{e\})$. Further, we set $k' = k$ and $\ell = 0$. All of these operations can be performed in polynomial time. Now G_1 is a Hamiltonian path (which is a forest with exactly one connected component) and G_2 is the graph that is obtained when we remove all edges of G_1 from G. Removing the edges of G_1 lowers the degree of each vertex by two, except at the two endpoints of the Hamiltonian path which have their degree lowered by one. Since each vertex of G is of degree three, G_2

contains $|V| - 2$ vertices of degree one and two vertices of degree two. A cycle requires three vertices of degree two, and thus G_2 is a forest. Now $I' = (\mathcal{G}, k', \ell)$, where $\mathcal{G}$ has the layers G_1 and G_2, is an instance of MSVC on forests, and we have $\tau = 2$.

Assume that I is a yes-instance of VERTEX COVER. Then there is a vertex cover V' of size at most k for G, which is also a vertex cover of size at most k for both G_1 and G_2 as every edge of these two layers is also present in G. Also, if we choose $\mathcal{S} = (V', V')$ as solution to I', then we have $|V' \triangle V'| = 0 \leq \ell$. Thus, I' is a yes-instance of MSVC.

Conversely, assume that I' is a yes-instance of MSVC. Then there are two vertex covers each of size at most k, V_1 and V_2, that cover G_1 and G_2, respectively. Since $\ell = 0$, we have $V_1 = V_2$. Thus, V_1 covers both G_1 and G_2. Because every edge of G is present in either G_1 or G_2, V_1 is a vertex cover of size at most k of G, and I is a yes-instance of VERTEX COVER. □

Lemma 3.11. *There is a polynomial-time many-one reduction from MSVC where every layer is a forest, $\tau = 2$ and $\ell = 0$ to MSVC where every layer is a tree, $\tau = 2$ and $\ell = 0$.*

Proof. Let $I = (\mathcal{G} = (V, \mathcal{E}, \tau), k, 0)$ be an instance of MSVC and let the forests G_1 and G_2 be the only layers of $\mathcal{G}$. We construct an instance $I' = (\mathcal{G}', k', 0)$, where $\mathcal{G}' = (V', \mathcal{E}', 2)$ has the layers G_1 and G_2, as follows: We set $V' = V \cup \{u, v\}$, add the edge $e = \{u, v\}$ to both G_1 and G_2 and set $k' = k + 1$. The graph $T = (\{u, v\}, \{e\})$ is a tree and thus both G_1 and G_2 retain the properties of a forest. We want to connect exactly one vertex of each connected component (except T) of both layers to the vertex v.

Note that the connected components of a given graph G can be found in polynomial time [VLL90]. A *bridge* is an edge $e = \{u, v\}$ such that if e is removed, then there is no path from u to v. If two acyclic subgraphs of a given graph G are connected by only a bridge, then G is acyclic as well.

We repeat the following steps for G_1 and G_2: Assume that G_i consists of $m + 1$ connected components with vertex sets $C_1, ..., C_m, C_{m+1} = \{u, v\}$. Let $v_1, ..., v_m$ be arbitrary vertices with $v_i \in C_i$ for $i \in \{1, ..., m\}$. We add m edges $e_1, ..., e_m$ where $e_j = \{v_j, v\}$ for $j \in \{1, ..., m\}$ to G_i. All of these operations can be performed in polynomial time. Because we connected C_{m+1} to all other connected components, G_1 and G_2 are now connected graphs. However, because the e_j are bridges for $j \in \{1, ..., m\}$, the resulting graph remains acyclic and we get an acyclic connected graph, a tree. Now $I' = (\mathcal{G}', k', 0)$ is an instance of MSVC where every layer is a tree, $\tau = 2$ and $\ell = 0$.

Assume that I is a yes-instance of MSVC. Then there are two vertex covers each of size at most k, V_1 and V_2, that cover G_1 and G_2, respectively. Since $\ell = 0$ we have $V_1 = V_2$. Thus, V_1 covers both G_1 and G_2. The reduction only adds edges which are adjacent to v, which implies that $\mathcal{S} = (S_1 = V_1 \cup \{v\}, S_2 = V_1 \cup \{v\})$ is a solution to I'. Thus we have $|S_1| = |S_2| \leq k + 1 = k'$ and $|S_1 \triangle S_2| = 0$, and I' is a yes-instance of MSVC.

Conversely, assume that I' is a yes-instance of MSVC, then there are two vertex covers each of size at most k', V_1' and V_2', that cover G_1' and G_2', respectively. Since $\ell = 0$ we have $V_1' = V_2'$. Thus, V_1' covers both G_1' and G_2'. Because u is a leaf in both layers, every solution that includes u can be modified such that it includes its only neighbor v instead. We set $V_1 = V_2 = (V' \setminus \{u\}) \cup \{v\}$, where V_1 still covers both G_1' and G_2' and $|V| \leq k + 1$.

Also, G_1 and G_2 are covered by V_1, because all of their edges are also present in G'_1 and G'_2. However, v does not exist in G_1 and G_2, and we can remove v from V to get a vertex cover of size at most k for G_1 and G_2 and the solution $\mathcal{S} = (V_1, V_2)$. Therefore, I is a yes-instance of MSVC. □

Proof of Theorem 3.8. We know the following three polynomial-time many-one reductions:

- The reduction r_1 from the NP-hard Independent Set on cubic Hamiltonian graphs to Vertex Cover on cubic Hamiltonian graphs (Lemma 3.9)
- The reduction r_2 from Vertex Cover on cubic Hamiltonian graphs to MSVC where every layer is a forest, $\tau = 2$ and $\ell = 0$. (Lemma 3.10)
- The reduction r_3 from MSVC where every layer is a forest, $\tau = 2$ and $\ell = 0$ to MSVC where every layer is a tree, $\tau = 2$ and $\ell = 0$. (Lemma 3.11)

Assume that I is any instance of the NP-hard Independent Set on cubic Hamiltonian graphs, then we can compute the composition $r(I) = r_3(r_2(r_1(I)))$ of the three reductions and map I to an instance I' of MSVC where every layer is a tree, $\tau = 2$ and $\ell = 0$ in polynomial time. Thus, r is a polynomial-time many-one reduction from an NP-hard problem to MSVC where every layer is a tree and τ is a constant, which proves that MSVC remains para-NP-hard for the parameter τ even if every layer is a tree. □

We have proven that MSVC remains NP-hard if every layer is a tree and $\tau = 2$. Next, we want to extend this result to arbitrary values of τ. To do this, we need to provide two additional lemmata first.

Theorem 3.12. *MSVC remains para-NP-hard for the parameter τ for any constant value of $\ell \in \mathbb{N}$ if every layer is a tree.*

Lemma 3.13. *Let $I = (\mathcal{G} = (V, \mathcal{E}, \tau), k, \ell)$ be an instance of MSVC with $k \leq |V|$. If I has a solution $\mathcal{S} = (S_1, ..., S_\tau)$ with $|S_1| < k$, then I also has a solution $S' = (S'_1, ..., S'_\tau)$ with $|S'_1| = |S_1| + 1$.*

Proof. Let S_m be the first layer in which a vertex is added in $\mathcal{S}$, and let v be the respective vertex. If no such v exists, then let $v \in S_1$ be an arbitrary vertex (we know that such v exists because we have $k \leq |V|$). Now set $S'_1 = S_1 \cup \{v\}, ..., S'_{m-1} = S_{m-1} \cup \{v\}$ (if we chose an arbitrary vertex v, we set $m = \tau + 1$ and add v to every S_i with $i \in \{1, ..., \tau\}$). Then $S' = (S'_1, ..., S'_\tau)$ is a solution to I: We have $S_i \subseteq S'_i$ for $i \in \{1, ..., \tau\}$ because we have $S'_i = S_i$ for $i \geq m$ and $S'_i = S_i \cup \{v\}$ for $i < m$. Moreover, we have $|S'_i \triangle S'_{i+1}| \leq \ell$ for $i \in \{1, ..., \tau\}$: If $i \geq m$, we have $S'_i = S_i$ and $S'_{i+1} = S_{i+1}$ and thus $|S'_i \triangle S'_{i+1}| = |S_i \triangle S_{i+1}| \leq \ell$. If $i < m - 1$, we have $S'_i = S_i \cup \{v\}$ and $S'_{i+1} = S_{i+1}\{v\}$ and thus $|S'_i \triangle S'_{i+1}| = |S_i \triangle S_{i+1}| \leq \ell$. Finally, if $i = m - 1$, we know that $v \notin S_{m-1}$, $v \in S'_m = S_m, S'_{m-1} = S_{m-1} \cup \{v\}$ and thus $|S'_{m-1} \triangle S'_m| = |S_{m-1} \triangle S_m| - 1 \leq \ell$. □

Lemma 3.14. *Let $I = (\mathcal{G}, k, \ell)$ be an instance of MSVC. If I has a solution $\mathcal{S} = (S_1, ..., S_\tau)$ and $k \leq |V|$, then I also has a solution $S' = (S'_1, ..., S'_\tau)$ with $|S_1| = k$.*

Proof. If $|S_1| = k$, then we are done. If $|S_1| < k$, then we can apply Lemma 3.13 iteratively until $|S_1| = k$. □

Proof of Theorem 3.12. First, we prove by induction that MSVC is para-NP-hard for the parameter τ and every *even* natural number ℓ if every layer is a tree.

We have already seen that MSVC is para-NP-hard (and NP-hard for $\tau = 2$) for $\ell = 0$ if every layer is a tree (Theorem 3.8). Now, assume that MSVC on trees is para-NP-hard (and NP-hard for $\tau = 2$) if $\ell = c$ where c is an arbitrary even number. Let $I = (\mathcal{G} = (V, \mathcal{E}, 2), k, c)$ be an instance of MSVC on trees where $\mathcal{G}$ has the layers G_1 and G_2. We prove that MSVC on trees is NP-hard for $c+2$ by constructing a polynomial-time many-one reduction that maps I to an instance $I' = (\mathcal{G}' = (V', \mathcal{E}', \tau), k', c+2)$ where $\mathcal{G}'$ has the layers G_1' and G_2', each being a tree.

Let $v \in V$ be an arbitrary vertex. We set $V' = V \cup \{v_1, v_2, v_3, v_4\}$ and $\mathcal{E}' = \mathcal{E} \cup \{(\{v_1, v_2\}, 1), (\{v_3, v_4\}, 2), (\{v_1, v\}, 1), (\{v_3, v\}, 2)\}$. Effectively, this adds four vertices to the temporal graph $\mathcal{G}$, and two disjoint edges (one in each layer) between them. Another edge connects either endpoint of the newly added edge to the rest of the graph. We also set $k' = k + 1$. All of these operations can be performed in polynomial time.

Assume that I is a yes-instance of MSVC, then there are two vertex covers each of size at most k, S_1 and S_2, that cover G_1 and G_2, respectively. Also, we have $|S_1 \triangle S_2| \leq c$. Now set $S_1' = V_1 \cup \{v_1\}$ and $S_2' = V_2 \cup \{v_3\}$. Then (S_1', S_2') is a solution to I', because S_1' and S_2' are vertex covers of size at most $k' = k + 1$ of G_1' and G_2', and $|S_1' \triangle S_2'| \leq c + 2$. Assume that I' is a yes-instance of MSVC, then there are two vertex covers each of size at most k', S_1' and S_2', that cover G_1' and G_2', respectively. Also, because v_1 and v_2 are incident to the edge $e = \{v_1, v_2\}$ connecting them in G_1, either v_1 or v_2 must be contained in V_1'. Analogous to this, either v_3 or v_4 must be contained in V_2'. Without loss of generality, we assume that $v_1 \in V_1'$ and $v_3 \in V_2'$, because by doing so we also cover the edges $(\{v_1, v\}, 1)$ and $(\{v_3, v\}$: If a there is a solution that includes v_2 (v_4) in G_1 (G_2), then there is also a solution with v_3 (v_4) in the respective layer. Because every edge of G_1 (G_2) is also present in G_1' (G_2'), and S_1' and S_2' are vertex covers of size at most $k' = k+1$ of G_1' and G_2', they are also vertex covers G_1 and G_2. Further, because we have $v_1 \notin V$ and $v_3 \notin V$, we can set $V_1 = V_1' \setminus \{v_1\}$ and $V_2 = V_2' \setminus \{v_3\}$ and get two vertex covers of size at most $k' - 1 = k$ of G_1 and G_2. We have $|V_1 \triangle V_2| = |V_1' \triangle V_2'| - 2 = c + 2 - 2 = c$, and thus I' is a yes-instance of MSVC.

Next, we prove that MSVC is para-NP-hard for the parameter τ and every uneven natural number ℓ if every layer is a tree.

Let $I = (\mathcal{G} = (V, \mathcal{E}, 2), k, c)$ be an instance of MSVC on trees where $\mathcal{G}$ has the layers G_1 and G_2. Assume that MSVC on trees is para-NP-hard (and NP-hard for $\tau = 2$) for $\ell = c$ where c is an arbitrary even number. We prove that MSVC on trees is para-NP-hard for $\ell = c + 1$ by proving that I is a yes-instance of MSVC if and only if the instance $I' = (\mathcal{G}, k, c + 1)$ is a yes-instance of MSVC (technically, we are performing a polynomial-time many-one reduction here, which changes nothing but the value of ℓ). Assume that $I = (\mathcal{G}, k, c)$ is a yes-instance of MSVC and its layers G_1 and G_2 are trees. Then there is a solution $\mathcal{S} = (S_1, S_2)$ with $|S_1 \triangle S_2| \leq c \leq c + 1$ and thus I remains a yes-instance of MSVC if we increment c by one.

Now assume that $I' = (\mathcal{G}, k, c+1)$ is a yes-instance of MSVC and its layers G_1' and G_2' are trees. Either we have $|V| \leq k$, in which case I is a yes-instance due to Lemma 3.19, or because of Lemma 3.14 there is a solution $\mathcal{S}' = (S_1', S_2')$ such that $|S_1'| = k$. Further, we can assume that no more than $\frac{c}{2}$ vertices are deleted in S_2': If exactly $\frac{c}{2}$ vertices are deleted in S_2', at most $\ell - \frac{c}{2} = c + 1 - \frac{c}{2} = \frac{c}{2} + 1$ vertices can be added in S_2'. If $d > \frac{c}{2}$ vertices are deleted in S_2', then only $c + 1 - d < \frac{c}{2} + 1$ vertices can be added. As soon as we exceed $\frac{c}{2}$ delete actions in S_2', it does not allow us to perform any more add actions. Thus, if there is a solution that deletes more than deleting $\frac{c}{2}$ vertices is in G_2, then there is another one with no more than $\frac{c}{2}$ delete actions in G_2. If only $\frac{c}{2}$ vertices are deleted and $|S_1'| = k$, then we can only add $\frac{c}{2}$ vertices in S_2' and we have $|S_1' \triangle S_2'| \leq 2 \cdot \frac{c}{2} = c$, which implies that I is a yes-instance of MSVC. □

3.2 Tractable cases

In some cases, we can solve the MSVC problem in polynomial time, or even return "yes" instantly. In most of the situations discussed in this section, either the parameters k and ℓ are high enough to allow us to choose every relevant vertex in every layer, or we prove that a variant of MSVC is equivalent to a variant of Vertex Cover which is known to be solvable in polynomial time.

Lemma 3.15. *If for an instance of MSVC or MSVC-Sum with $|E_i| = 1$ for all layers G_i one has $k \geq \tau$, then it is a yes-instance.*

Proof. Let $e_i = \{v_i, u_i\} \in E_i$ be the only edge of layer G_i. Because $k \geq \tau$ we can choose $S_1 = S_2 = ... = S_\tau = \{v_1, v_2, ..., v_\tau\}$. Now $(S_1, ..., S_\tau)$ is a solution to the MSVC instance because

- $v_i \in e_i$ for every $i \in \{1, ..., \tau\}$ and thus S_i is a vertex cover of size at most $\tau \leq k$.
- $|S_i \triangle S_{i+1}| \leq \ell$ for every $i \in \{1, ..., \tau-1\}$ because $S_i = S_{i+1}$ for every $i \in \{1, ..., \tau-1\}$ and thus $|S_i \triangle S_{i+1}| = 0$ for every $i \in \{1, ..., \tau-1\}$.

□

Lemma 3.16. *If for an instance of MSVC with $|E_i| = 1$ for all layers G_i one has $\ell \geq 2$, then it is a yes-instance.*

Proof. Let $e_i = \{v_i, u_i\} \in E_i$ be the only edge of layer G_i. We choose $S_i = \{v_i\}$ for every $i \in \{1, ..., \tau\}$. Now $(S_1, ..., S_\tau)$ is a solution to the MSVC instance because

- $v_i \in e_i$ for $i \in \{1, ..., \tau\}$ and as v_i covers the only edge in E_i, S_i is a vertex cover of size at most 1.
- $|S_i \triangle S_{i+1}| \leq \ell$ for every $i \in \{1, ..., \tau-1\}$ because $|S_i \triangle S_{i+1}| = 2$ for every $i \in \{1, ..., \tau-1\}$ and $\ell \geq 2$.

□

Theorem 3.17. *If for an instance of MSVC with $|E_i| = 1$ for all layers G_i one has $k \geq \tau$ or $\ell \geq 2$, then it is a yes-instance.*

Proof. See Lemma 3.15 and Lemma 3.16. □

Lemma 3.18. *If for an instance of MSVC-Sum with $|E_i| = 1|$ for all layers G_i one has $\ell \geq 2\tau - 2$, then it is a yes-instance.*

Proof. Let $e_i = \{v_i, u_i\} \in E_i$ be the only edge of layer G_i. We choose $S_i = \{v_i\}$ for every $i \in \{1, ..., \tau\}$. Now $(S_1, ..., S_\tau)$ is a solution to the MSVC instance because

- $v_i \in e_i$ for $i \in \{1, ..., \tau\}$ and as v_i covers the only edge in E_i, S_i is a vertex cover of size at most 1.
- $\sum_{i=1}^{\tau-1} |S_i \triangle S_{i+1}| \leq \ell$ because $|S_i \triangle S_{i+1}| = 2$ for every $i \in \{1, ..., \tau - 1\}$ and thus $\sum_{i=1}^{\tau-1} |S_i \triangle S_{i+1}| = 2\tau - 2 \leq \ell$.

□

Lemma 3.19. *Let $I = (\mathcal{G} = (V, \mathcal{E}, \tau), k, \ell)$ be an instance of MSVC. If $k \geq |V|$, then I is a yes-instance of MSVC.*

Proof. Let $\mathcal{S} = (S_1, ..., S_\tau)$ with $S_i = V$ for every $i \in \{1, ..., \tau\}$. Then $\mathcal{S}$ is a solution to I, because V covers every (temporal) edge of $\mathcal{G}$ and thus all of its layers, and $|S_i \triangle S_{i+1}| = |V \triangle V| = 0 \leq \ell$ for all $i \in \{1, ..., \tau - 1\}$. □

In Theorem 3.1 we have proven that MSVC is NP-hard even if every layer is a tree. However, if there is only a single layer, or ℓ is too high to limit the selection of vertices for a solution, then the given MSVC instance effectively degrades to τ independent instances of Vertex Cover, which can be solved in polynomial time. The following theorem will prove this.

Theorem 3.20. *MSVC can be solved in polynomial time if every layer is a tree and $\tau = 1$ or $\ell \geq 2k$*

To prove Theorem 3.20, we construct two polynomial-time reductions. They can be used to transform an instance of MSVC where every layer is a tree and $\tau = 1$ or $\ell \geq 2k$ to an instance of Vertex Cover on trees, which is known to be solvable in polynomial time.

Lemma 3.21. *There is a polynomial-time many-one reduction from MSVC with $\tau = 1$ where the only layer is a tree to* Vertex Cover *on a tree*

Proof. Let $I = (\mathcal{G}, k, \ell)$ be an instance of MSVC with $\tau = 1$ such that the only layer G_1 is a tree. We construct an instance $I' = (G', k')$ of vertex cover by setting $G' = G_1$ (which implies that the input graph to the Vertex Cover instance is also a tree) and $k' = k$ (obviously this can be performed in polynomial time). Assume that I is a yes-instance of MSVC, then a vertex cover of size at most k exists for G_1, which is also a vertex cover of size at most $k' = k$ for $G' = G_1$. This implies that I' is a yes-instance of VERTEX COVER. Assume I' is a yes-instance of VERTEX COVER, then a vertex cover of size at most k' exists for G', which is also a vertex cover of size at most $k = k'$ for $G_1 = G'$. We also have $|S_i \triangle S_{i+1}| \leq \ell$ for every $i \in \{1, ..., \tau - 1\}$ in any solution $\mathcal{S} = (S_1, ..., S_\tau)$ because $\tau = 1$. This implies that I is a yes-instance of MSVC. □

Lemma 3.22. *There is a polynomial-time Turing reduction from MSVC where every layer is a tree and $\ell \geq 2k$ to VERTEX COVER on trees*

Proof. Let $I = (\mathcal{G}, k, \ell)$ be an instance of MSVC with $\ell \geq 2k$ such that every layer is a tree, and G_1, ..., G_τ its layers. We construct τ VERTEX COVER instances I'_1, ..., I'_τ by setting $I'_i = (G_i, k)$ for every $i \in \{1, ..., \tau\}$ (obviously this can be performed in polynomial time). This implies that the input graphs for each VERTEX COVER instance are also trees. Assume that I is a yes-instance of MSVC, then a vertex cover of size at most k exists for every layer G_i, and the constructed VERTEX COVER instances I'_i are yes-instances. Assume that I'_i is a yes-instance for every $i \in \{1, ..., \tau\}$, then a vertex cover of size at most k exists for every G_i. We also have $|S_i \triangle S_{i+1}| \leq \ell$ for every $i \in \{1, ..., \tau - 1\}$ because regardless of the specific solution $\mathcal{S} = (S_1, ..., S_\tau)$ for any two sets S_i and S_{i+1} with $|S_i| \leq k$ and $|S_{i+1}| \leq k$ the maximum symmetric difference is $2k \leq \ell$. This implies that I is a yes-instance of MSVC.

Note that this is equivalent to a polynomial-time Turing reduction: We create τ instances of VERTEX COVER in polynomial time, which allows us to solve the MSVC instance efficiently if the solutions to the VERTEX COVER instances are known. □

Proof of Theorem 3.20. Due to Lemma 3.21, there is a polynomial-time many-one reduction from any instance of the MSVC problem with $\tau = 1$ where the only layer is a tree to VERTEX COVER on trees. It is folklore that VERTEX COVER on trees is polynomial-time solvable, which implies that MSVC with $\tau = 1$ where the only layer is a tree is also polynomial-time solvable by applying the reduction and solving the returned instance of VERTEX COVER. Due to Lemma 3.22, there is a polynomial-time Turing reduction from MSVC where every layer is a tree and $\ell \geq 2k$ to VERTEX COVER on trees, which is contained in P. This implies that the problem itself is polynomial-time solvable. □

3.3 Kernelizations

In this section, we study the possibility of polynomial kernels for MSVC and MSVC-Sum. In many cases, we can provide polynomial kernels or bikernels by using data reduction rules or reductions to VERTEX COVER, for which kernelizations are known. In other cases, we rule out polynomial kernels using the cross-composition framework by Bodlaender, Jansen, and Kratsch [BJK14].

First, we study the case where $\ell = 0$. Intuitively, $\ell = 0$ means that we can consider the MSVC instance as an instance of VERTEX COVER where all layers are combined to a single graph. This is because if $\ell = 0$, we have to cover all layers with the same vertex set. Consequently, the case where $\ell = 0$ has similar properties to VERTEX COVER, and admits a kernel of size $\mathcal{O}(k^2)$. Next, we will formally prove this.

Lemma 3.23. *There are polynomial-time many-one reductions that map an instance $I = (\mathcal{G}, k, \ell)$ with $\ell = 0$ of either MSVC or MSVC-Sum to an instance $I' = (G, k)$ of* Vertex Cover*.*

Proof. Let $I = (\mathcal{G} = (V, \mathcal{E}, \tau), k, 0)$ be an instance of either MSVC or MSVC-Sum and $G_1 = (V, E_1), ..., G_\tau = (V, E_\tau)$ the layers of $\mathcal{G}$. We set $G = (V, E = E_1 \cup \ldots \cup E_\tau$, which can be performed in polynomial time, and $I' = (G, k)$. Assume that I is a yes-instance of either MSVC or MSVC-Sum, then there is a solution $(S_1, ..., S_\tau)$ with $S_1 = \ldots = S_\tau \subseteq V$ such that S_1 is a vertex cover of size at most k of all layers of $\mathcal{G}$, and thus S_1 covers all edges of every layer. Therefore, I' is a yes-instance of Vertex Cover. Assume that I' is a yes-instance of Vertex Cover, then G has a vertex cover S_1 of size at most k. Because $G = (V, E)$, and E includes all edges of every layer of $\mathcal{G}$, S_1 is also a vertex cover of size at most k of G_i for every $i \in \{1, ..., \tau\}$, and $(S_1, ..., S_\tau)$ with $S_i = S_1$ for every $i \in \{1, ..., \tau\}$ is a solution to I: We have $|S_i \triangle S_{i+1}| \leq \ell$ for every $i \in \{1, ..., \tau - 1\}$ (if I is an instance of MSVC) and $\sum_{i=1}^{\tau-1} |S_i \triangle S_{i+1}| \leq \ell$ (if I is an instance of MSVC-Sum), because we have $S_1 = \ldots = S_\tau$. Therefore, I is a yes-instance of either MSVC or MSVC-Sum. □

Theorem 3.24. *Any instance $I = (\mathcal{G}, k, \ell)$ of MSVC where $\ell = 0$ has a bikernel of size at most $\mathcal{O}(k^2)$.*

Proof. We know that there is a polynomial-time reduction from MSVC to Vertex Cover (Lemma 3.23), and it is known that Vertex Cover has a kernel of size at most $\mathcal{O}(k^2)$ [CKJ01]. Therefore, we can obtain a bikernel of size $\mathcal{O}(k^2)$ for MSVC by performing the reduction to Vertex Cover first, then the known kernelization for Vertex Cover. Both of these steps require polynomial time, and the retrieved bikernel K is a yes-instance if and only if I is a yes-instance because

- The intermediate instance J of Vertex Cover obtained through a polynomial-time reduction from I is a yes-instance if and only if I is a yes-instance, as this is a general property of reductions.
- The retrieved kernel K is a yes-instance if and only if the instance J, which serves as the input for the kernelization algorithm of Vertex Cover, is a yes-instance, as this is a general property of kernelizations.

□

For arbitrary values of ℓ, we can rule out kernels of size polynomial in k using the technique of AND-cross-composition.

Theorem 3.25. *MSVC admits no kernel of size polynomial in k, even if every layer contains exactly one edge and $\ell = 1$.*

Proof. We prove that MSVC with $|E_i| = 1$ for all layers G_i and $\ell = 1$ AND-cross-composes into itself: Let $\mathcal{R}$ be a relation on the set which contains all possible valid inputs for MSVC with $|E_i| = 1$ for all layers G_i and $\ell = 1$ such that for two instances

$I = (\mathcal{G}_i, k_i, \ell)$ and $J = (\mathcal{G}_j, k_j, \ell)$ with $\mathcal{G}_i = (V_i, \mathcal{E}_i, \tau_i)$ and $\mathcal{G}_j = (V_j, \mathcal{E}_j, \tau_j)$ one has $(I, J) \in \mathcal{R}$ if and only if $|V_i| = |V_j|$ and $k_i = k_j$. Now $\mathcal{R}$ is a polynomial equivalence relation because:

- It is obvious that $\mathcal{R}$ is an equivalence relation.
- Given two MSVC instances $I = (\mathcal{G}_i, k_i, \ell_i)$ and $J = (\mathcal{G}_j, k_j, \ell_j)$, we can decide whether they belong to the same equivalence class in polynomial time by comparing V_i to V_j and k_i to k_j.
- For any finite set S of Vertex Cover instances, $\mathcal{R}$ partitions its elements into a number of classes that is polynomially bounded in the size of the largest element of S: Let $I_1 = (G_1 = (V_1, E_1), k_1)$ be the element of S with the highest number of vertices in its graph, and let $I_2 = (G_2 = (V_2, E_2), k_2)$ be the element of S with the highest parameter k. Then the number of equivalence classes is at most $|V_1| \cdot k_2$.

Our AND-cross-composition now takes t instances $I_1 = (\mathcal{G}_1, k, \ell), ..., I_t = (\mathcal{G}_t, k, \ell)$ of MSVC with $|E_i| = 1$ for all layers G_i and $\ell = 1$ which belong to the same equivalence class of $\mathcal{R}$ as its input and creates an instance $I = (\mathcal{G} = (V, \mathcal{E}, \tau), k, \ell)$ of MSVC with $\ell = 0$ as follows: Let $G_{i,1}, ..., G_{i,\tau_i}$ be the layers of $\mathcal{G}_i$ for every $i \in \{1, ..., t\}$. We combine all of these layers for every $\mathcal{G}_i$ with $i \in \{1, ..., t\}$ to a single sequence of layers by performing the following three steps:

- Start with an empty sequence of layers and $\mathcal{E} = \emptyset$. Now, for every $i \in \{1, ..., t\}$ add the layers $G_{i,1}, ..., G_{i,\tau_i}$ in that order to our temporal graph $\mathcal{G}$. We receive the following sequence of layers in $\mathcal{G}$: $G_{1,1}, ..., G_{1,\tau_1}, ..., G_{t,1}, ..., G_{t,\tau_t} = G'_1, ..., G'_\theta$.
- After every layer G_{i,τ_i} with $i \in \{1, ..., t-1\}$, add $2k-2$ layers which are identical to G_{i,τ_i}.
- Before every layer $G_{i,1}$ with $i \in \{2, ..., t\}$, add a layer identical to $G_{i,1}$.

The parameter k is polynomially bounded in $\max_i |x_i| + \log t$ as I uses the same k as the I_i.

Assume that $I_i = (\mathcal{G}_i, k, 1)$ is a yes-instance of MSVC for every $i \in \{1, ..., t\}$. Then there is a vertex cover V_j of size at most k for any layer G_j of $\mathcal{G}_i$ and $j \in \{1, ..., \tau_i\}$, and thus, there are such vertex covers for all layers of $\mathcal{G}$. We can construct a solution $\mathcal{S} = (S_{1,1}, ..., S_{t,\tau_t}) = (S_1, ..., S_\theta)$ to $\mathcal{G}$ as follows: For every layer G_j that has been added to $\mathcal{G}$ at position $p \in \mathbb{N}$ in the first step, we set $S_p = V_j$. Now, consider a sequence of $2k-2$ layers added after a layer G_{q,τ_q} for $q \in \{1, ..., t\}$ in the second step, which along with the layers added in the third step we will refer to as duplicate layers. For any such sequence, we start with the first of its layers G_m with $m \in \{1, ..., \theta\}$ and set the corresponding vertex set S_m in our solution $\mathcal{S}$ to $S_{m-1} \setminus \{v\}$ where v is an arbitrary vertex that is not needed to cover $G_m = \ldots = G_{m+k-2}$. If no such v exists and we have $|S_m| = 1$, then we do nothing. We repeat these steps with $G_m, \ldots, G_{m+k-2}$, $k-1$ times in total. Because we have $|E_i| = 1$ for all layers of all temporal graphs in all of the input instances and thus also in our constructed instance I, each layer only requires one vertex to be covered. Thus, after repeating the steps $k-1$ times, starting with a set S_m with $|S_m| \leq k$, we are left with a set S_{m+k-2} containing only a single vertex w which covers the only edge of the corresponding layer G_{m+k-2}. In each of the next $k-1$ duplicate layers, we add vertices that we need to cover the next sequence of non-duplicate layers: If the next non-duplicate layer G_a with $a \in \{1, ..., \theta\}$ originally belonged to a temporal graph with solution $\mathcal{G}_f = (S_{s,1}, ..., S_{s,\tau_s})$ and $s \in \{1, ..., t\}$, we want to have $S_a = S_{s,1}$. To achieve this, we first add a vertex u that covers G_a. Next, we proceed to add one vertex of $S_{s,1}$ in each layer until we reach a layer G_r with $r \in \{1, ..., \theta\}$ where for the corresponding vertex set S_r, we have $|S_r| = k$.

Note that here we assume that $|S_{s,1}| = k$, which is possible due to Lemma 3.14.

If we have $w \notin S_{s,1}$, we delete it in the final, $(2k-1)$-st, duplicate layer, and add the k-th vertex of $S_{s,1} = S_r$ next. Intuitively speaking, the additional $2k-1$ duplicate layers allow us to replace every vertex we used to cover the layers of one of the original t MSVC instances with a new initial configuration for layers of the next original MSVC instance.

We know that every S_i is either a vertex cover taken from a solution to one of the original t MSVC instances where it covers the same layer as in I, or was created through a sequence of add and delete actions which do not affect the vertex cover property (vertices were only deleted if it was unaffected, and adding vertices never affects it unless we exceed k vertices, which doesn't happen here). Thus, every S_i in $\mathcal{S}$ remains a vertex cover of size at most k for every $i \in \{1, \ldots, \tau_1 + \ldots + \tau_t + (2k-1) \cdot (t-1)\}$. Moreover, we have $|S_i \triangle S_{i+1}| \leq \ell = 1$ for every $i \in \{1, ..., \tau - 1\}$ because only single add and delete actions were performed in the duplicate layers (and those layers following a sequence of duplicate layers), and the subsequences of layers which do not contain duplicate layers fulfill this condition because they were taken from yes-instances of MSVC. Therefore, I is a yes-instance of MSVC.

Assume that $I = (\mathcal{G}, k, 1)$ is a yes-instance of MSVC with $\mathcal{G}$ having layers $G_1, ..., G_t$. Then there is a solution $\mathcal{S} = (S_1, ..., S_t)$ such that S_i is a vertex cover of size at most k of G_i for every $i \in \{1, ..., t\}$ and $|S_i \triangle S_{i+1}| \leq \ell = 1$ for every $i \in \{1, ..., t-1\}$ holds. The layers of the temporal graphs $\mathcal{G}_1, \ldots \mathcal{G}_t$ of the instances $I_1, ..., I_t$ all occur in exactly the same order in the yes-instance $\mathcal{G}$, and therefore also fulfill these conditions. This implies that all of them are yes-instances of MSVC.

We have now proven that MSVC with $|E_i| = 1$ for all layers G_i and $\ell = 1$ AND-cross-composes into itself, which proves that in general no kernel of size polynomial in k exists assuming that NP $\not\subseteq$ coNP / poly. $\square$

For arbitrary values of ℓ, we do not have a simple reduction to Vertex Cover available — thus, we approach this case in a different way. We provide kernels (instead of bikernels, as for the case of $\ell = 0$). In order to decrease the size of an instance, we use two data reduction rules. The first one allows us to delete vertices which are isolated in every layer and thus cannot be used to cover any edges.

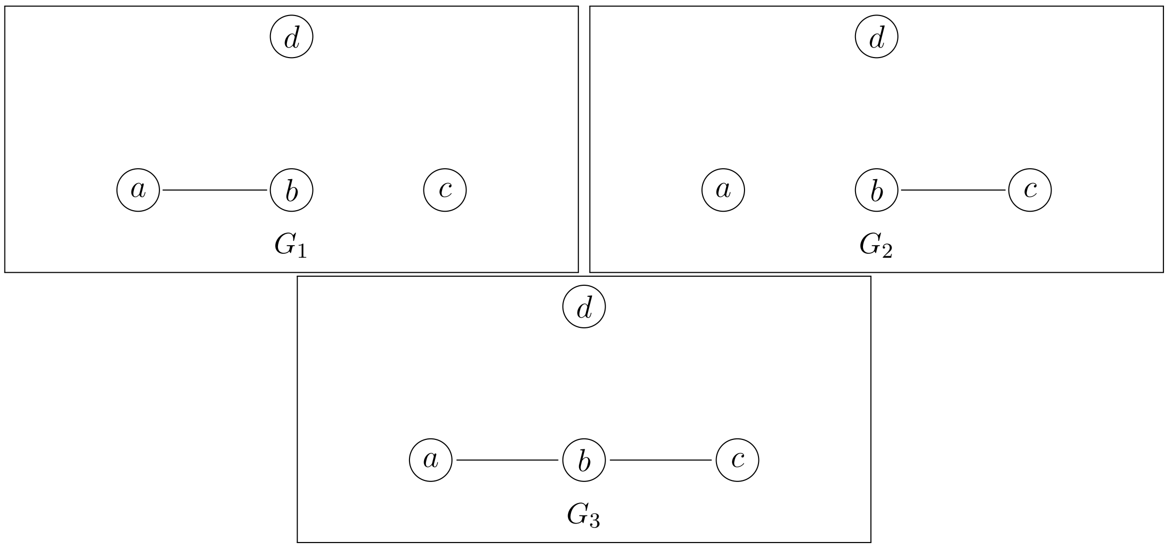

Figure 3.1: A temporal graph $\mathcal{G} = (V, \mathcal{E}, \tau)$ with layers G_1, G_2 and G_3. The vertex d is isolated in all three layers of $\mathcal{G}$. Let $I = (\mathcal{G}, k, \ell)$ be an instance of MSVC. Then, according to Reduction Rule 3.26, d can be deleted from V regardless of the values of k and ℓ.

Reduction Rule 3.26. *Let $I = (\mathcal{G} = (V, \mathcal{E}, \tau), k, \ell)$ be an instance of MSVC or MSVC-Sum, with vertex $v \in V$ such that for all layers $G_i = (V, E_i)$ one has $e \in E_i \Rightarrow v \notin e$. Then delete v from $\mathcal{G}$.*

Lemma 3.27. *Reduction Rule 3.26 is correct and can be applied in linear time.*

Proof. Let $\mathcal{S} = (S_1, ..., S_\tau)$ be a solution to I and let $G_1, ..., G_\tau$ be the layers of $\mathcal{G}$. Further, set $\mathcal{S}' = (S'_1, ..., S'_\tau)$ where $S'_i = S_i \setminus \{v\}$ for every $i \in \{1, ..., \tau\}$, and let I' be the instance I after application of Reduction Rule 3.26. Then $\mathcal{S}'$ is a solution to I' because

- S_i is a vertex cover of size at most k of G_i for every $i \in \{1, ..., \tau\}$. Because v does not have any incident edges, $S'_i = S_i \setminus \{v\}$ also covers G_i, and G'_i which is identical to G_i except that v has been removed in G'_i.
- $|S_i \triangle S_{i+1}| \leq \ell$ for every $i \in \{1, ..., \tau - 1\}$ (if I is an instance of MSVC) because this holds for I and neither any S_i nor ℓ have been changed in I'.
- $\sum_1^{\tau-1} |S_i \triangle S_{i+1}| \leq \ell$ (if I is an instance of MSVC-Sum) because this holds for I and neither any S_i nor ℓ have been changed in I'.

Reduction Rule 3.26 can be applied in linear time ($\mathcal{O}(|V| \cdot \tau)$), because it has to be checked for each vertex if it has any neighbor exactly τ times. □

Figure 3.1 shows an example of Reduction Rule 3.26 being used to eliminate an isolated vertex.

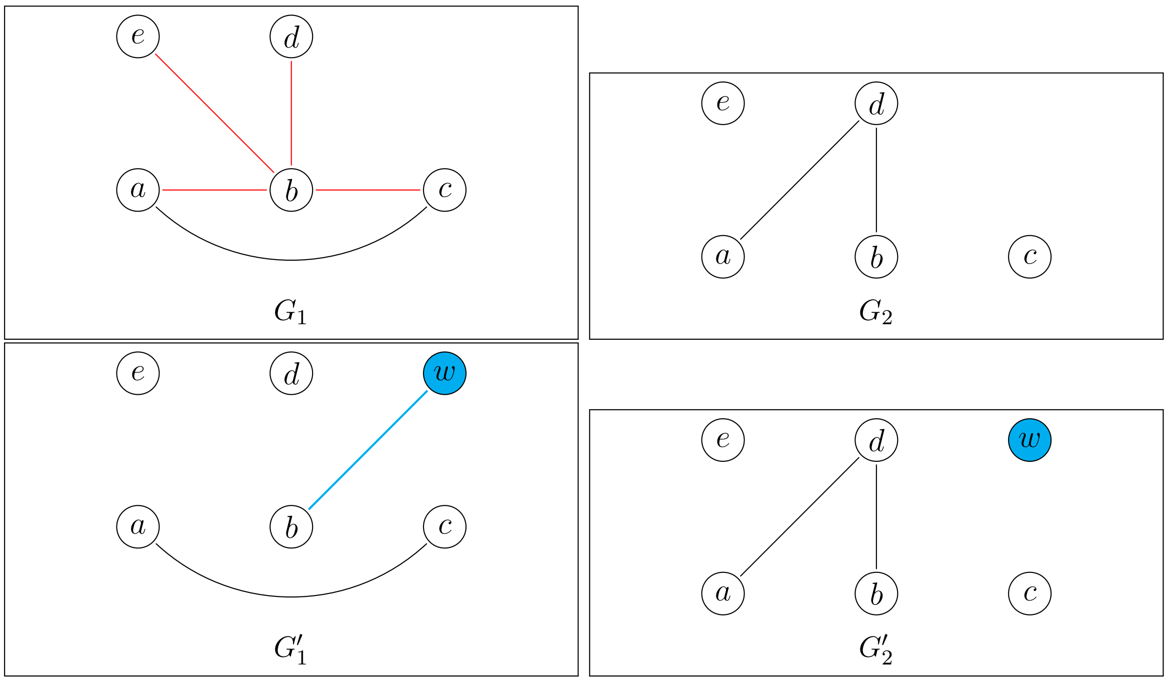

Figure 3.2: Let $I = (\mathcal{G}, k = 3, \ell)$ be an instance of MSVC with $\mathcal{G}$ having layers G_1 and G_2. We use Reduction Rule 3.28 to transform $\mathcal{G}$ into $\mathcal{G}'$ having layers G'_1 and G'_2: Because b is of degree $4 > 3 = k$, we know that $b \in S_1$ where $\mathcal{S} = (S_1, S_2)$ is a solution to I. If (G_1, k) was an instance of Vertex Cover, then we could simply delete b and its incident edges and decrease k by one. However, because all layers of an instance of MSVC share the same vertex set and the same k, we have to keep b. Instead, we delete all of its incident edges and add a new vertex w and an edge $\{b, w\}$ connecting both. We are now forced to select either b or w in S_1 — deleting the edges has not made the instance easier to solve, but we managed to decrease its size.
Because all layers share the same vertex set, w is present in all layers (in this case we can see that w has been added in G_2).

Our second data reduction rule allows us to delete the incident edges of a vertex whenever it is obvious that the vertex has to be part of a solution, in which case it will cover the edges.

Reduction Rule 3.28. *Let $I = (\mathcal{G}, k, \ell)$ be an instance of MSVC or MSVC-Sum, $G_1, ..., G_\tau$ the layers of $\mathcal{G}$ and $v \in V$ a vertex. If v has a degree greater than k in a subset of layers $\mathcal{L} \subseteq \{G_1, ..., G_\tau\}$, then add a vertex w to V, remove all incident edges of v in all layers $L \in \mathcal{L}$ and add the edge $\{v, w\}$ to all of them instead.*

Lemma 3.29. *Reduction Rule 3.28 is correct and can be applied in $\mathcal{O}(|V|^2 \cdot \tau)$ time.*

Proof. Assume that I is a yes-instance of MSVC or MSVC-Sum, and I' the instance we get by applying Reduction Rule 3.28. Then a solution $\mathcal{S} = S_1, ..., S_\tau$ exists such that every S_i is a vertex cover of size at most k of G_i. We know that every (temporal) edge e

in $\mathcal{G}'$ either also exists in $\mathcal{G}$, or is connected to a vertex v which has a degree greater than k in the corresponding layer G_m in $\mathcal{G}$ with $m \in \{1, ..., \tau\}$. In the first case, it is covered by $\mathcal{S}$. In the second case, we know that $v \in S_m$ because v has a degree greater than k and it would therefore be impossible to cover all of its incident edges without adding v to S_m. This implies that $\mathcal{S}$ is a solution to I' (which fulfills $|S_i \triangle S_{i+1}| \leq \ell$ for every $i \in \{1, ..., \tau - 1\}$ for MSVC or $\sum_{i=1}^{\tau-1} |S_i \triangle S_{i+1}| \leq \ell$ for MSVC-Sum as it did in I), and I' is a yes-instance.

Assume that I' is a yes-instance of MSVC or MSVC-Sum. Then a solution $\mathcal{S} = S_1, ..., S_\tau$ exists such that every S_i is a vertex cover of size at most k of G_i. Let $e = (\{v, w\}, t)$ be a temporal edge where v is a vertex which we haveReduction Rule 3.28 used on in G_t. We know that both v and w are incident only to the edge e. Therefore, we can set $\mathcal{S} = \mathcal{S}' = (S_1, ..., S_\tau)$ and modify it to include v but not w (by setting $S'_t = S_t \setminus \{w\} \cup \{v\}$). If we perform this for all edges which we did not use Reduction Rule 3.28 on, then the modified sequence $\mathcal{S}'$ covers all edges of $\mathcal{G}$ because:

- If a (temporal) edge $e \in \mathcal{E}$ has not been affected by Reduction Rule 3.28, then it is still present in $\mathcal{G}'$ and it is therefore covered by $\mathcal{S}$ and also by $\mathcal{S}'$, as $\mathcal{S}'$ is identical to $\mathcal{S}$ with respect to vertices which were not affected by Reduction Rule 3.28.
- If $e = (\{v, w\}, t)$ has been deleted through application of Reduction Rule 3.28, and v was a vertex with degree greater than k in G_t, then by above construction $v \in S_t$.

This implies that $\mathcal{S}$ is a solution to I (which fulfills $|S_i \triangle S_{i+1}| \leq \ell$ for every $i \in \{1, ..., \tau - 1\}$ for MSVC or $\sum_1^{\tau-1} |S_i \triangle S_{i+1}| \leq \ell$ for MSVC-Sum as it did in I'), and I is a yes-instance.

Reduction Rule 3.28 can be applied in $\mathcal{O}(|V|^2 \cdot \tau)$ time because each time we apply it we have to check the degree of every vertex at most τ times (one time in every layer), add at most τ vertices and edges, and remove at most $\tau \cdot |V| - 1$ edges. □

Figure 3.2 shows an example of Reduction Rule 3.28 being used to decrease the amount of edges in a temporal graph.

As we have now proven the correctness of two data reduction rules, we can try to apply them to instances of MSVC in order to decrease their size. Next, we will show how iterative application of Reduction Rule 3.26 and Reduction Rule 3.28 can be used to retrieve a kernel of a bounded size.

Theorem 3.30. *Let $K = (\mathcal{G}, k, \ell)$ be an instance of MSVC such that Reduction Rule 3.28 and Reduction Rule 3.26 are not applicable. Then K is of size at most $3k^2 \cdot \tau$.*

Proof. Because Reduction Rule 3.26 and Reduction Rule 3.28 are not applicable, I is an MSVC instance where there is no vertex v such that for all layers $G_i = (V, E_i)$ one has $e \in E_i \Rightarrow v \notin e$, or v has a degree greater than k in any layer G_i.

If any layer has more than k^2 edges, we can return a trivial no-instance as kernel because k vertices each of degree at most k can cover at most than k^2 edges. Otherwise, we get a maximum of $k^2 \cdot \tau$ edges, with a maximum of $2k^2 \cdot \tau$ incident vertices. Any vertex that is not incident to any edge has already been deleted by Reduction Rule 3.26. Hence, the total size of K is at most $3k^2 \cdot \tau$. □

In cases where ℓ is high enough to allow disjoint vertex covers being chosen for every layer, we can view an instance of MSVC as several separate instances of VERTEX COVER. This observation can be used to retrieve a Turing kernelization.

Observation 3.31. *Let $I = (\mathcal{G}, k, \ell)$ be an instance of MSVC where $\ell \geq 2k$, and $S_1, ..., S_\tau$ vertex covers of size at most k of the layers $G_1, ..., G_\tau$ of $\mathcal{G}$. We have $|S_i \triangle S_{i+1}| \leq \ell$ for every $i \in \{1, ..., \tau - 1\}$ because regardless of the specific solution $\mathcal{S} = (S_1, ..., S_\tau)$ for any two sets S_i and S_{i+1} with $|S_i| \leq k$ and $|S_{i+1|} \leq k$ the maximum symmetric difference is $2k \leq \ell$.*

Theorem 3.32. *Let $I = (\mathcal{G}, k, \ell)$ be an instance of MSVC where $\ell \geq 2k$. If we are able to solve an instance of MSVC with size bounded in k^2 in constant time, then we can decide I in polynomial time.*

Proof. Let $G_1, ..., G_\tau$ be the layers of $\mathcal{G}$. We construct τ instances $I_1, ..., I_\tau$ of VERTEX COVER by setting $I_i = (G_i, k)$ for every $i \in \{1, ..., \tau\}$. This can be performed in polynomial time, as the amount of instances I_i created is linear in τ and each of them can also be created in linear time.

Now, assume that I is a yes-instance of MSVC. Then a sequence $\mathcal{S} = (S_1, ..., S_\tau)$ exists such that S_i is a vertex cover of size at most k of G_i for every $i \in \{1, ..., \tau\}$. The existence of such vertex covers for the G_i implies that $I_i = (G_i, k)$ is a yes-instance for every $i \in \{1, ..., \tau\}$.

Assume that $I_i = (G_i, k)$ is a yes-instance of VERTEX COVER for every $i \in \{1, ..., \tau\}$. Then there are vertex covers $S_1, ..., S_\tau$ of size at most k for $G_1, ..., G_\tau$. Along with Observation 3.31, this implies that I is a yes-instance of MSVC.

In total, τ instances of VERTEX COVER have been created, each of which has a kernel of size at most $\mathcal{O}(k^2)$ (Chen, Jia and Kanj 1999). Thus, if we can solve each instance in constant time, we can decide I in polynomial time.

This is equivalent to a linear-time Turing kernel of size at most $\mathcal{O}(k^2)$. □

Lemma 3.33. *MSVC admits no kernelizations on arbitrary layers for parameter τ.*

Proof. MSVC and MSVC-Sum are para-NP-hard for the parameter τ (Theorem 3.8), which implies that neither problem is fixed-parameter tractable or allows a kernelization for τ. □

Theorem 3.34. *Let $I = (\mathcal{G}, k, \ell)$ be an instance of MSVC where $|E_i| = 1$ for all layers G_i, then I has a kernel of size at most $5\tau + 1$.*

Proof. If $k \geq \tau$ or $\ell \geq 2$, then I is a yes-instance due to Lemma 3.15 and Lemma 3.16, so we can return a trivial yes-instance as kernel. Otherwise, use Reduction Rule 3.26 repeatedly as many times as possible. We receive an MSVC instance $K = (\mathcal{G}' = (V', \mathcal{E}, \tau), k, \ell)$ where Reduction Rule 3.26 is not applicable which means there is no vertex v so that for all layers $G_i = (V, E_i)$ one has $e \in E_i \Rightarrow v \notin e$. Thus, each remaining vertex must be contained in at least one temporal edge. We have $|\mathcal{E}| = \tau$ because we assumed $|E_i| = 1$ for each of the τ layers. Because each temporal edge can contain at most 2 vertices, and each vertex must be contained in at least one temporal edge, at most 2τ vertices remain. Also, by assumption we have $k < \tau$ and $\ell \leq 1$. For the size of K we get $|V| + |\mathcal{E}| + \tau + k + \ell \leq 2\tau + \tau + \tau + \tau + 1 = 5\tau + 1$. □

3.4 FPT Algorithm

Before we study an FPT algorithm for the parameter k in depth, we show that MSVC can be solved in $\mathcal{O}((1.2738^k + kn) \cdot \tau)$ if $\ell \geq 2k$. Note that MSVC is fixed-parameter tractable for τ if $|E_i| = 1$ for all layers G_i due to Theorem 3.34, and for $k + \tau$ due to Theorem 3.30.

Theorem 3.35. *An instance $I = (\mathcal{G}, k, \ell)$ of MSVC can be solved in $\mathcal{O}((1.2738^k + kn) \cdot \tau)$, where n is the size of the given instance if one has $\ell \geq 2k$.*

Proof. We show that any MSVC instance can be reduced to τ separate Vertex Cover instances if $\ell \geq 2k$: Let $I = (\mathcal{G}, k, \ell)$ be an instance of MSVC with $\ell \geq 2k$. We construct τ Vertex Cover instances I'_1, ..., I'_τ by setting $I'_i = (G_i, k)$ for every $i \in \{1, ..., \tau\}$ (obviously this can be performed in linear time).

Assume that I is a yes-instance of MSVC, then a vertex cover of size at most k exists for every layer G_i, and the constructed Vertex Cover instance I'_i is a yes-instance for $\{1, ..., \tau\}$. Assume that I'_i is a yes-instance for every $i \in \{1, ..., \tau\}$ and the respective solutions $S'_1, ..., S'_\tau$ exist, then a vertex cover S'_i of size at most k exists for every G_i. We also have $|S_i \triangle S_{i+1}| \leq \ell$ for every $i \in \{1, ..., \tau - 1\}$ because of Observation 3.31. This implies that I is a yes-instance of MSVC. Moreover, we can derive a solution $\mathcal{S} = (S_1, ..., S_\tau)$ from the solutions to the Vertex Cover instances by setting $S_i = S'_i$ for every $i \in \{1, ..., \tau\}$. We can solve the instance by using the reduction and solving the τ instances of Vertex Cover. Each can be decided in $\mathcal{O}((1.2738^k + kn))$ [CKX06]. In total we get a running time of $\mathcal{O}((1.2738^k + kn) \cdot \tau)$. □

In the remaining section we will provide an FPT algorithm for the parameter k and the respective complexity upper bounds.

Intuitively speaking, the algorithm generates a list of all possible vertex covers of size either k or $k - 1$ for each layer of a given temporal graph. Then, taking the parameter ℓ into consideration, the algorithm checks if one of the possible vertex covers of the final layer is reachable from one of the possible initial configurations. The algorithm consists of a main routine (Algorithm 3.42) and three subroutines searchTree (Algorithm 3.39), checkReachability (Algorithm 3.41) and resolveDC (Algorithm 3.40).

Further, we will allow vertex sets to contain “don't care values” (DCs): As stated above, we want to generate all vertex covers of size either k or $k - 1$ for each layer. If we find a vertex cover of a smaller size, we can fill the remaining slots with DCs, because any vertex will work. If a vertex cover of a layer contains a DC X, then X can be considered as equal to any vertex contained in a vertex cover of the layer before when computing the symmetric difference. Each set can contain multiple DCs but each vertex only once (it is a multiset with respect to DCs and a set with respect to vertices). Because of this, we have to introduce a slightly different notion of symmetric difference.

We will now provide formal definitions of reachability and symmetric difference between vertex sets containing DCs.

Definition 3.36. (Symmetric difference between vertex sets containing DCs) Assume that S and T are vertex sets possibly containing DCs. Then $S\hat{\triangle}T$ is the *symmetric difference* of S and T. Our FPT algorithm does not require us to compute the actual symmetric difference set, but we can compute its cardinality as follows:While both S and T contain at least one DC, remove one DC from both. When we are done, either S, T, or both contain no DCs. Now we can compute $|S\hat{\triangle}T|$ by applying the following rules iteratively:

Let A and B be two vertex sets such that either A or B possibly contains DCs, let v be a vertex and let X be a DC.

- $|A\hat{\triangle}B| = |A\triangle B|$ if A and B do not contain any DCs
- $|(A \cup \{X\})\hat{\triangle}B| = |A\hat{\triangle}B| + 1$
- $|(A \cup \{v\})\hat{\triangle}(B \cup \{X\})| = |A\hat{\triangle}B|$ (choose $v \notin B$ if possible)
- If $A = \emptyset$ then $|A\hat{\triangle}B| = |B|$
- If $B = \emptyset$ then $|A\hat{\triangle}B| = |A|$

It is important to note that $A\hat{\triangle}B$ is no longer commutative if either A or B contain DCs. A DC contained in B is considered as equal to any element in A, a DC contained in A is not.

Definition 3.37. (reachability) Let $I = (\mathcal{G} = (V, \mathcal{E}, \tau), k, \ell)$ be an instance of MSVC with $\mathcal{G}$ having layers $G_1, ..., G_\tau$, and let V_i and V_j be vertex covers of size of at most k of G_i and G_j with $i, j \in \{1, ..., \tau\}$. The vertex set V_j is reachable from V_i if $i + 1 = j$ and $|V_i\hat{\triangle}V_j| \leq \ell$, or if $i + 1 < j$, there is a vertex cover V_{i+1} of size at most k of G_{i+1}, $|V_i\hat{\triangle}V_{i+1}| \leq \ell$, and V_j is reachable from V_{i+1}.

Note that I is a yes-instance of MSVC if and only if there are two vertex covers V_1 and V_τ of size at most k of G_1 and G_τ such that V_τ is reachable from V_1.

Theorem 3.38. *MSVC is fixed-parameter tractable with respect to parameter k, an instance can be decided in $\mathcal{O}(\tau \cdot (2^{2k^2+5k} \cdot k^2) + |V|^3)$ time.*

We want to prove that the algorithm is both correct and runs in FPT time. To do this, we need to prove several lemmata first. For instance, we will prove the correctness of the subroutines of the main algorithm.

Lemma 3.43. *If resolveDC is called by checkReachability, then the returned family $\mathcal{R}$ contains only vertex sets U such that $|U\hat{\triangle}T| \leq \ell$.*

Algorithm 3.39.
searchTree:
Input : *An undirected graph $G = (V, E)$, two integers $k, k' \in \mathbb{N}$, a family $\mathcal{L}$ of vertex sets and a vertex set S.*
Output: *If searchTree is initially called with $\mathcal{L} = S = \emptyset$, then $\mathcal{L}$ now contains all vertex covers of G of size k and $k-1$. Vertex covers of a smaller size are filled with DCs until their size is $k-1$, then until it is k, and the resulting vertex set is added to $\mathcal{L}$ in both cases.*

```
if $k' = 0$ then
    if $|E| = 0$ then
        Add $S$ to $\mathcal{L}$ ;
    end
    return
end
if $|E| = 0$ then
    Add DCs to $S$ such that $|S| = k - 1$ ;
    Add $S$ to $L$ ;
    Add a DC to $S$ ;
    Add $S$ to $L$ ;
    return
end
$e \leftarrow \{u, v\} \in E$ ;
searchTree($G - u$, $k' - 1$, $L$, $S \cup \{u\}$) ;
searchTree($G - v$, $k' - 1$, $L$, $S \cup \{v\}$) ;
return
```

Algorithm 3.40.
resolveDC:
Input : *Two vertex sets S and T such that $\ell + 2d \geq |S \hat{\triangle} T|$, an integer i.*
Output: *A family $\mathcal{R}$ containing vertex sets such that for every $U \in \mathcal{R}$ we have $|U \hat{\triangle} T| \leq \ell$, where i determines the number of DCs removed from S.*

```
$i \leftarrow \lceil \frac{i}{2} \rceil$ ;
$\mathcal{R} \leftarrow \emptyset$ ;
Remove $i$ DCs from $S$ and all DCs from $T$ ;
$U \leftarrow T \setminus S$ ;
for each $i$-element subset $M$ of $U$ do
    Add $S \cup M$ to $\mathcal{R}$ ;
end
return $\mathcal{R}$
```

Algorithm 3.41.
checkReachability:
Input : *Two families of vertex sets $\mathcal{L}_1$ and $\mathcal{L}_2$ and an integer ℓ.*
Output: *$\mathcal{L}_1$ now contains only vertex sets A such that there is a vertex set B in $\mathcal{L}_2$ with $|A \hat{\triangle} B| \leq \ell$.*

```
for each $S \in \mathcal{L}_1$ do
    Let $d$ be the number of DCs in $S$ ;
    for each $T \in \mathcal{L}_2$ do
        if $|S \hat{\triangle} T| \leq \ell$ then
            Break;
        else if $\ell + 2d \geq |S \hat{\triangle} T| > \ell$ then
            $\mathcal{L}_1 = (\mathcal{L}_1 \cup$ resolveDC($S$, $T$, $|S \hat{\triangle} T| - \ell$)$) \setminus \{S\}$ ;
            Break;
        end
        Delete $S$ from $\mathcal{L}_1$;
    end
end
return
```

Algorithm 3.42.
main:
Input : *An instance $I = (\mathcal{G}, k, \ell)$ of MSVC with $\mathcal{G}$ having layers $G_1, ..., G_\tau$, and empty families $\mathcal{L}_1, ..., \mathcal{L}_\tau$ of vertex sets.*
Output: *[TRUE] if I is a yes-instance of MSVC, otherwise [FALSE].*

```
Apply kernelization as described in Theorem 3.30 and its proof ;
searchTree($G_\tau$, $k$, $\mathcal{L}_\tau$, $\emptyset$);
for $i \leftarrow \tau - 1$ to $1$ do
    searchTree($G_i$, $k$, $k'$, $\mathcal{L}_i$, $\emptyset$);
    checkReachability($\mathcal{L}_i$, $\mathcal{L}_{i+1}$, $\ell$);
end
if $\mathcal{L}_1 \neq \emptyset$ then
    return [TRUE];
end
return [FALSE];
```

Proof. We can assume that $\ell + 2d \geq |S \hat{\triangle} T|$ (see input of resolveDC). Let S' denote the vertex set S after line 3 of resolveDC has been executed and i DCs have been removed. We now have $|S' \hat{\triangle} T| = |S \hat{\triangle} T| - i$, because whenever we remove a DC, $|S' \hat{\triangle} T|$ will decrease by one. This is because of Definition 3.36: The DC contained in T is counted as equal to an arbitrary element of S. $\square$

For better understanding, we provide an example:

Let $S = \{v, X\}$ and $T = \{w, X\}$, $S \hat{\triangle} T = \{v, w\}$, and $|S \hat{\triangle} T| = 2$. If we remove a DC from S, then we get $S' = \{v\}$, $S' \hat{\triangle} T = \{w\}$, $|S \hat{\triangle} T| = 1$.

Now, let $U = S'$ in line 6 of resolveDC. We add i elements of T which are not contained in S (see line 4) to U, which decreases $|U \hat{\triangle} T|$ by i. In total, we get $|U \hat{\triangle} T| = |S \hat{\triangle} T| - 2i \leq |S \hat{\triangle} T| - (|S \hat{\triangle} T| - \ell) = \ell$.

Lemma 3.44. *Let $I = (\mathcal{G} = (V, \mathcal{E}, \tau), k, \ell)$ be a yes-instance of MSVC and $k \leq |V|$. Then there is a solution $\mathcal{S} = (S_1, ..., S_\tau)$ with $k - 1 \leq |S_i| \leq k$ for every $i \in \{1, ..., \tau\}$.*

Proof. Because I is a yes-instance of MSVC, we know that a solution $\mathcal{S} = (S_1, ..., S_\tau)$ exists. Because of Lemma 3.14 we can assume that $|S_1| = k$. If $k - 1 \leq |S_i| \leq k$ for every $i \in \{1, ..., \tau\}$, then we are done. Else, let S_p be the first element of the solution $\mathcal{S}$ with $|S_p| < k - 1$, where $p \in \{2, ..., \tau\}$. We know that $|S_p| < |S_{p-1}|$, which implies that a vertex v is deleted in G_p. Now, consider the next add action being performed in $\mathcal{S}$, let w be the vertex added and G_q, where $q \in \{3, ..., \tau\}$, the layer in which the action is performed (if no such G_q exists, we skip the next steps). We construct a new solution $\mathcal{S}' = (S'_1 = S_1, ..., S'_{p-1} = S_{p-1}, S'_p, ..., S'_{q-1}, S'_q = S_q, S'_{q+1} = S_{q+1} ..., S'_\tau = S_\tau)$ as follows: Let $S'_i = S_i$ for every $i \in \{1, ..., p-1\} \cup \{q, ..., \tau\}$ and let $S'_i = S_i \cup \{v, w\}$ for every $i \in \{p, ..., q-1\}$. Effectively, we swap the add and the delete action: The vertex v is not deleted until G_q, and the vertex w is added in G_p rather than in G_q. Next, we prove that $\mathcal{S}'$ is a solution to I: We have $|S'_i| \leq k$ for every $i \in \{1, ..., \tau\}$, because if $i \in \{1, ..., p-1\} \cup \{q, ..., \tau\}$, we have $S'_i = S_i \leq k$. Further, we have $S'_i = S_i \cup \{v, w\}$ for every $i \in \{p, ..., q-1\}$, which implies that $|S'_i| \leq k$, because we know that $|S_p| < k-1$, in the original solution $\mathcal{S}$ no vertex was added until S_q, and in $\mathcal{S}'$ we only add two vertices v and w until S_q. Also, $S_i \subseteq S'_i$ for every $i \in \{1, ..., \tau\}$, and thus we know that S_i is a vertex cover of size at most k of G_i for every $i \in \{1, ..., \tau\}$. We have $|S'_i \triangle S'_{i+1}| \leq \ell$ for every $i \in \{1, ..., p-2\} \cup \{q, ..., \tau - 1\}$ because $S'_i = S_i$. Further, we have $S'_{p-1} \triangle S'_p \leq \ell$ because $S_{p-1} \triangle S_p \leq \ell$, $S'_{p-1} = S_{p-1}$ and $S'_p = S_p \cup \{v, w\}$ where $v \in S'_{p-1}$. Similarly, $S'_{q-1} \triangle S'_q \leq \ell$ holds because of $S_{q-1} \triangle S_q \ell$, $S'_q = S_1$ and $S'_{q-1} = S_{q-1} \cup \{v, w\}$ where $w \in S_q$. Finally, we have $S_i \triangle S_{i+1}$ for every $i \in \{p, ..., q-2\}$ because $S'_i = S_i \cup \{v, w\}$ for every $i \in \{p, ..., q-1\}$ and $S_i \triangle S_{i+1} \leq \ell$ for every $i \in \{p, ..., q-2\}$. Now set $\mathcal{S} = \mathcal{S}'$. We repeat the last steps as long as S_p and G_q an be found as described above. If no S_p can be found, we are done, because we have $S_i \geq k - 1$ for every $i \in \{1, ..., \tau\}$. If a S_p can be found, but no G_q, we know that no more add actions are performed after G_p. Assume that $|S_p| = k - r$ with $r \in \mathbb{N}$. We know that $|S_{p-1}| \geq k - 1$ because S_p is the first element of $\mathcal{S}$ with a cardinality lower than $k - 1$. This implies that $r - 1$ vertices $v_1, ..., v_{r-1}$ are deleted in G_p. We omit the respective delete actions (and all delete actions performed after G_p), and add $v_1, ..., v_{r-1}$ to $S_p, ..., S_\tau$: We set $S'_i = S_i$ for $i \in \{1, ..., p-1\}$ and $S'_i = S_i \cup \{v_1, ..., v_{r-1}\}$ for $i \in \{p, ..., \tau\}$ and receive a new solution $\mathcal{S} = (S'_1, ..., S'_\tau)$. We prove that $\mathcal{S}'$ is a solution: We know that S_i is a vertex cover of size at most k of G_i for every $i \in \{1, ..., \tau\}$. We have $S_i \subseteq S'_i$ for every $i \in \{1, ..., \tau\}$, and we know that $|S_i|' \leq k$ for every $i \in \{1, ..., \tau\}$ because after omitting all delete actions after

G_p, we have $|S_p| = \ldots = |S_\tau| = k - r$, and we add exactly $r-1$ vertices to $S_p, ..., S_\tau$. Further, we have $|S'_i \triangle S'_{i+1}| \leq \ell$ for every $i \in \{1, ..., \tau\}$ because we have $|S_i \triangle S_{i+1}| \leq \ell$ for every $i \in \{1, ..., \tau\}$ and $S'_i = S_i \cup T$ for every $i \in \{1, ..., \tau\}$, where $T \subseteq S_{i-1}$. Now we have $S'_i \geq k-1$ for every $i \in \{1, ..., \tau\}$. □

Lemma 3.45. *Algorithm 3.42 is correct — it returns [TRUE] if and only if the input I is a yes-instance of MSVC.*

Proof. Because kernelizations return a yes-instance if and only if the input is a yes-instance, line 1 does not affect the correctness of the algorithm. Assume that $I = (\mathcal{G}, k, \ell)$ is a yes-instance of MSVC. Then there is a solution $\mathcal{S} = S_1, ..., S_\tau$ with $k-1 \leq |S_i| \leq k$ for $i \in \{1, ..., \tau\}$ (Lemma 3.44)*. Because searchTree initially generates all vertex covers of size either k or $k-1$ of G_i (by trying out both incident vertices of every edge in lines 15-16), we can assume that $S_i \in \mathcal{L}_i$ after the respective searchTree call for $i \in \{1, ..., \tau\}$. Because we have $|S_j \hat{\triangle} S_{j+1}| \leq \ell$ for $j \in \{1, ..., \tau-1\}$, every S_j will trigger the break in line 6 of checkReachability, including S_1, which will stop checkReachability from deleting S_1 from $\mathcal{L}_1$ and leave us with $\mathcal{L} \neq \emptyset$. Consequently, main returns [TRUE].

Assume that Algorithm 3.42 returns [TRUE]. We prove by induction that vertex covers V_1 and V_τ exist such that V_τ is reachable from V_1: We have $\mathcal{L}_1 \neq \emptyset$. Let V_1 be an arbitrary element of $\mathcal{L}_1$. Then there is a $V_2 \in \mathcal{L}_2$ such that $|V_1 \hat{\triangle} V_2| \leq \ell$. Either we had $|V_1 \hat{\triangle} T| \leq \ell$ for $T \in \mathcal{L}_2$ in line 5 of checkReachability, or V_1 was added through a call to resolveDC, in which case we have $|V_1 \hat{\triangle} T| \leq \ell$ for $T \in \mathcal{L}_2$ (Lemma 3.43). Therefore, V_2 is reachable from V_1. Assume that $L_i \neq \emptyset$ for an $i \in \{1, ..., \tau-1\}$. Then there is a $V_{i+1} \in \mathcal{L}_{i+1}$ such that $|V_i \hat{\triangle} V_{i+1}| \leq \ell$. Either we had $|V_i \hat{\triangle} T \leq \ell|$ for $T \in \mathcal{L}_{i+1}$ in line 5 of checkReachability, or V_i was added through a call to resolveDC, in which case we have $|V_i \hat{\triangle} T \leq \ell|$ for $T \in \mathcal{L}_{i+1}$ (Lemma 3.43). Therefore, V_{i+1} is reachable from V_i. We can conclude that there are vertex covers V_1 and V_τ of G_1 and G_τ such that V_τ is reachable from V_1, there is a solution $\mathcal{S} = V_1, ..., V_\tau$, and I is a yes-instance of MSVC.
*Note that technically Lemma 3.44 can only be applied if $k \leq |V|$. However, since our algorithm allows DCs, we can easily achieve $|S_i| \geq k-1$ for a solution $\mathcal{S} = (S_1, ..., S_\tau)$ and every $i \in \{1, ..., \tau\}$ by adding DCs to the S_i. Alternatively, we can modify the algorithm such that it instantly returns [TRUE] in the trivial case of $k > |V|$. □

Lemma 3.46. *When searchTree is called, the number of subsequent recursive calls is bounded in $\mathcal{O}(2^k)$.*

Proof. Each call causes further recursive calls with a branching factor of 2 (searchTree calls itself 2 times), and a maximum call stack depth of k: In each call k is decreased by one, and when k reaches zero, the function terminates. This implies a maximum of $\mathcal{O}(2^k)$ calls. □

Lemma 3.47. *Whenever checkReachability is called, we have $|\mathcal{L}_1| \cdot |\mathcal{L}_2| \leq 2^{2k^2+4k}$.*

Proof. Whenever checkReachability is called, $\mathcal{L}_1$ contains a list of vertex covers generated by searchTree and has not yet been modified by checkReachability. Because searchTree is called a maximum of $\mathcal{O}(2^k)$ times with each vertex set family (Lemma 3.46) and each call generates a maximum of two vertex covers, we can assume that $|\mathcal{L}_1| \in \mathcal{O}(2^k)$. However, $|\mathcal{L}_2|$ has possibly been modified through calls to resolveDC. Let $S' \in \mathcal{L}_1$ be a vertex set. We prove that resolveDC is called a maximum of 2^{2k^2+k} times with $S = S'$, because there are no more than 2^{2k^2+k} possible values for T which cause resolveDC to be called along with S'. If resolveDC is called with $S = S'$ and $T = T'$, we have $\ell + 2d \geq |S' \hat{\triangle} T'| > \ell$. We can think of S' and T' as two consecutive parts of a possible solution $\mathcal{S} = S_1, ..., S' = S_i, T' = S_{i+1}, ..., S_\tau$ such that at most p add and q delete actions are performed in G_{i+1}, and $p + q = |S' \hat{\triangle} T'|$. There are $\binom{k}{q} \leq 2^k$ ways to delete q vertices from a set containing at most k vertices. Next, we have to add p vertices. The vertices added in T' have to be selected such that they cover the layer G_{i+1}, or, if it is already covered by the other vertices in T', the next layer G_{i+2}, etc. This is because if it was possible to add arbitrary vertices, the respective entry in T' would be a DC (DCs are only replaced with specific vertices if a certain vertex is needed to cover the next layers without letting the cardinality of the symmetric difference exceed the parameter ℓ). We can find all vertex sets which cover as many layers as possible with a branching algorithm analogous to searchTree. However, the branching algorithm may only find a partial vertex cover for the last layer it attempts to cover. To find every possible partial vertex cover of a layer, we have to try out both vertices contained in every edge in every step of the algorithm (except for the vertices already added in earlier steps), and the branching factor increases to $2k^2$ (as after applying the kernelization mentioned in Theorem 3.30, only k^2 edges with $2k^2$ incident vertices remain). We get a maximum of $\mathcal{O}(\binom{2k^2}{k}) \leq \mathcal{O}(2^{2k^2})$ possible results, which is the amount of k-element subsets of the $2k^2$ vertices which are incident to the remaining edges.

In total, we get $\mathcal{O}(2^k \cdot 2^{2k^2}) = 2^{2k^2+k}$ possible values for T' which cause solveDC to be called with S' and T', each call to solveDC causes at most $\mathcal{O}(2^k)$ elements to be added to a list, and each family has at most $\mathcal{O}(2^k)$ entries before checkReachability is called. This implies that on a checkReachability call, $\mathcal{L}_2$, which has been modified by checkReachability and resolveDC before, has at most $2^k \cdot 2^k \cdot 2^{k\cdot(2k+1)} = 2^{2k^2+3k}$ entries, and $|L_1| \cdot |\mathcal{L}_2| \leq 2^k \cdot 2^{2k^2+3k)} = 2^{2k^2+4k}$. □

Lemma 3.48. *Algorithm 3.42 can decide an instance of MSVC in $\mathcal{O}(\tau \cdot (2^{2k^2+5k} \cdot k^2) + |V|^3)$ time.*

Proof. The kernelization runs in $\mathcal{O}(|V|^3 \cdot \tau)$ as Reduction Rule 3.26 and Reduction Rule 3.28 are performed at most $|V|$ times and run in $\mathcal{O}(|V| \cdot \tau)$ and $\mathcal{O}(|V|^2 \cdot \tau)$ (see Lemma 3.27 and Lemma 3.29).

The function searchTree is called $1 + \tau - 1 = \tau$ times from the main function and each calls induces no more than $\mathcal{O}(2^k)$ subsequent recursive calls (Lemma 3.46). We get a total running time of $\mathcal{O}(\tau \cdot 2^k \cdot |V|)$, because each call requires $\mathcal{O}(|V|)$ time to remove a vertex and its incident edges from the graph in the final two steps. All other steps can be performed in constant time.

The function checkReachability is called $\tau - 1$ times from the main function. The inner loop is executed $|L_1| \cdot |L_2| \leq 2^{2k^2+4k}$ times (Lemma 3.47), and in each iteration we compute the symmetric difference of two sets with cardinality at most k, which can be performed in $\mathcal{O}(k^2)$, and call resolveDC. In resolveDC, we iterate over i-element subsets of a set with at most k elements, which allows for at most $\binom{k}{i} \leq 2^k$ iterations where each iteration can be performed in $\mathcal{O}(k^2)$ because $|S| < k$ and $|M| < k$. Therefore, we get a running time of $\mathcal{O}(2^k \cdot k^2)$ for each call to resolveDC, and $\mathcal{O}(\tau \cdot 2^{2k^2+4k} \cdot (2^k \cdot k^2 + k^2)) \subseteq \mathcal{O}(\tau \cdot 2^{2k^2+5k} \cdot k^2)$ for the total running time of all checkReachability calls. To compute the total running time, we need to add the running times of both searchTree (which does not change the asymptotic complexity) and the kernelization. In total we get $\mathcal{O}(\tau \cdot (2^{2k^2+5k} \cdot k^2) + |V|^3)$.

□

Proof of Theorem 3.38. We know that Algorithm 3.42 solves MSVC correctly (Lemma 3.45) in $\mathcal{O}(\tau \cdot (2^{2k^2+5k} \cdot k^2) + |V|^3)$ time (Lemma 3.48), which implies fixed-parameter tractability. □

Lemma 3.49. *After executing Algorithm 3.42 on an instance $I = (\mathcal{G}, k, \ell)$ of MSVC, each element contained in the family $\mathcal{L}_i$ is a vertex cover of size either k or $k-1$ of layer G_i of $\mathcal{G}$ for $i \in \{1, ..., \tau\}$.*

Proof. All elements are generated by either searchTree or resolveDC. The subroutine searchTree only adds a vertex set S to a family if the two following conditions hold:

- We have $|E| = 0$, which implies that all edges in $|E|$ have been removed by searchTree. This happens if and only if searchTree has added vertices to S such that S covers all edges, and is thus a vertex cover of $G = (V, E)$.
- Either we have $k' = 0$, which implies that exactly k vertices have been added to S, or we add DCs to S such that $|S| \geq k + 1$.

  The subroutine resolveDC only replaces DCs with specific vertices, which doesn't affect the vertex cover property or the cardinality of the respective vertex set.

□

Algorithm 3.42 allows us to solve MSVC as a decision program, but is useless if we want to extract a specific solution. With some small modifications we can trace back the solution after the main algorithm terminates. However, we have to decide if we want to exatact a solution or optimize memory consumption. The main idea behind Algorithm 3.51, which will be discussed later, is that we reduce the amount of vertex set families used, which we need to save enough history to be able to trace back the solution.

Theorem 3.50. *Algorithm 3.42 can be modified such that a solution $\mathcal{S}$ can be extracted whenever the input contains a yes-instance of MSVC.*

Proof. We make the following changes to Algorithm 3.42 and its subroutines: The L_i where $i \in \{1, ..., \tau\}$ contain pairs of vertex sets instead of just single vertex sets. The first element is the same vertex set as used in the unmodified version of the algorithm

— a vertex cover of size at most k generated by searchTree. The second element is initialized as NULL by searchTree,

We modify checkReachability and resolveDC as follows: Whenever an element $S \in L_1$ is not deleted but kept (see line 5 of checkReachability), or resolved and replaced by elements $S_1, ..., S_m$ with $m \in \mathbb{N}$ after being compared to T (see line 6 of resolveDC), we add a tuple containing both S and T rather than just S to $\mathcal{L}_1$. The vertex set T can be considered as a possible successor to S in a solution $\mathcal{S} = (S_1, ..., S, T, S_\tau)$, because both S and T are vertex covers of size at most k of their respective (consecutive) layer (if they were not, searchTree would not have generated them in the first place) and we have $|S \hat{\triangle} T| \leq \ell$ (because if we did not, checkReachability would have deleted S from its respective list), or resolveDC would not have added them (Lemma 3.43).

Next, assume that our input contains a yes-instance of MSVC. We extract a solution $\mathcal{S} = (S_1, ..., S_\tau)$ as follows: We select vertex sets $S_1, ..., S_\tau$ from the lists $L_1, ..., L_\tau$. We know that each element of the family $\mathcal{L}_i$ is a vertex cover of size at most k of S_i Because our input contains a yes-instance, we have $\mathcal{L}_1 \neq \emptyset$, and we can select an arbitrary tuple $(S_1', S_2') \in \mathcal{L}_1$ and set $S_1 = S_1'$. Next, we set $S_2 = S_2'$, with $(S_2', S_3') \in \mathcal{L}_3$, and continue to iteratively add S_i' to $\mathcal{S}$ whenever S_{i-1}' was the last element added and we have $(S_{i-1}', S_i \in \mathcal{L}_i)$. □

Algorithm 3.42 uses a different family of vertex sets for every layer, which can be considered wasteful as each family is used as an input to checkReachability only twice. The algorithm can be optimized such that the space complexity is lowered drastically. The basic idea here is to reduce the amount of families used to only two ($\mathcal{L}_0$ and $\mathcal{L}_1$). However, by doing so we lose the ability to extract a specific solution as described in Theorem 3.50.

Theorem 3.52. *Algorithm 3.42 can be modified such that its space complexity is lowered to $\mathcal{O}(2^{2k^2+3k} \cdot k + 3k^2 \cdot \tau)$ without increasing the running time*

Proof. Intuitively, we want to use only two (instead of τ) families of vertex sets. Instead of using one family per layer, we alternate between $\mathcal{L}_0$ and $\mathcal{L}_1$. Consider a modified version of Algorithm 3.42, Algorithm 3.51. The proof of correctness is analogous to that of Algorithm 3.42, as is the proof of it running in $\mathcal{O}(\tau \cdot (2^{2k^2+5k} \cdot k^2) + |V|^3)$ time Lemma 3.48. We will analyze the space complexity by listing the variables used.

- The input instance $I = (\mathcal{G}, k, \ell)$ with $\mathcal{G}$ having layers $G_1, ..., G_\tau$ requires space bounded in $\mathcal{O}(3k^2 \cdot \tau)$ (after the kernelization has been applied),

- The families $\mathcal{L}_0$ and $\mathcal{L}_1$ contain a total of at most 2^{2k^2+3k} elements in every step (Lemma 3.47) where every element is a vertex set of size at most k. We get a total space requirement of $\mathcal{O}(2^{2k^2+3k} \cdot k)$,

  The space requirements for all variables used in searchTree are multiplied by k because the maximum call stack depth is k (see Lemma 3.46).

- The input graphs $G = (V, E)$ require space bounded in $\mathcal{O}(k \cdot 3k^2)$.

Algorithm 3.51.
main_space:
Input : *An instance $I = (\mathcal{G}, k, \ell)$ of MSVC with $\mathcal{G}$ having layers $G_1, ..., G_\tau$, and empty families $\mathcal{L}_0, \mathcal{L}_1$ of vertex sets*
Output: *[TRUE] if I is a yes-instance of MSVC, else [FALSE]*

```
Apply kernelization as described in Theorem 3.30 and its proof ;
searchTree($G_\tau$, $k$, $k'$ $\mathcal{L}_1$, $\emptyset$);
$j \leftarrow 0$ ;
for $i \leftarrow \tau - 1$ to 1 do
    searchTree($G_i$, $k$, $\mathcal{L}_j$, $\emptyset$);
    checkReachability($\mathcal{L}_j$, $\mathcal{L}_{(j+1) \mod 2}$, $\ell$);
    $j \leftarrow (j+1) \mod 2$ ;
end
if $\mathcal{L}_1 \neq \emptyset$ then
    return [TRUE];
end
return [FALSE];
```

- The integers k require space bounded in $\mathcal{O}(k \cdot k) = \mathcal{O}(k^2)$.
- The vertex sets S, each having a size of at most k, require $\mathcal{O}(k^2)$.
- In total, we get a space complexity of $\mathcal{O}(2^{2k^2+3k} \cdot k + 3k^2 \cdot \tau)$.

  We neglect certain variables such as the loop counters and the parameters k and τ because it is obvious that these do not affect the asymptotic complexity.

□

4 Multistage Vertex Cover Sum (MSVC-Sum)

First, we investigate MSVC-Sum with respect to NP-hardness. It is important to note that in many cases, the proof is completely analogous to regular MSVC. This is due to the fact that many of our reductions are based on corner cases such as $\ell = 0$ or $\tau = 1$, where MSVC and MSVC-Sum are equivalent.

Theorem 4.1. *MSVC-Sum is NP-complete even if*

- *τ is any fixed constant,*
- *ℓ is any fixed constant,*
- *every layer G_i is a tree and $\ell = 0$,*
- *or $|E_i| = 1$ for every layer $G_i = (V, E_i)$ and $\ell = 0$.*

Proof. There are polynomial-time many-one reductions from Vertex Cover to MSVC-Sum with arbitrary values of τ (analogous to Lemma 3.4) and ℓ (analogous to Lemma 3.3), to MSVC-Sum with $|E_i| = 1$ for all layers G_i and $\ell = 0$ (analogous to Lemma 3.5), and to MSVC-Sum where each layer G_i is a tree and $\ell = 0$ (analogous to Lemma 3.7), which proves NP-hardness for these problem variants. MSVC-Sum is contained in NP (analogous to Lemma 3.2), therefore all of its variants which were proven to be NP-hard are also NP-complete. □

Theorem 4.2. *MSVC-Sum can be solved in polynomial time if every layer is a tree and $\tau = 1$ or $\ell \geq 2k \cdot (\tau - 1)$*

Proof. The proof is analogous to that of Theorem 3.20. □

Theorem 4.3. *MSVC-Sum is para-NP-hard for the parameter τ even if every layer is a tree.*

Proof. The proof is analogous to that of Theorem 3.8. □

Theorem 4.4. *Let $I = (\mathcal{G} = (V, \mathcal{E}, \tau), k, \ell)$ be an instance of MSVC-Sum and let $G_1, ..., G_\tau$ be the layers of $\mathcal{G}$. If one has $|E_i| = 1$ for all layers G_i and $k \geq \tau$ or $\ell \geq 2 \cdot (\tau - 1)$, or if one has $k \geq |V|$, then I is a yes-instance of MSVC-Sum..*

Proof. The proof is analogous to that of Theorem 3.17 and Lemma 3.19. □

Next, we will discuss kernelizations for MSVC-Sum. Again, in most cases the results are analogous or very similar to MSVC. One notable exception is the proof that MSVC does not admit a kernel of size polynomial in k for high values of ℓ: For MSVC, we used an AND-cross-composition approach that combines the layers of the temporal graphs of t input instances of MSVC, $I_1, ..., I_t$, into the single temporal graph of a output instance of MSVC, I. Between the layers which originally belonged to I_i and I_j, where $i \in \{1, ..., t-1\}$, we added duplicate layers such that in a solution we can select disjoint vertex covers for the last of the original layers of I_i and the first of the original layers of I_j. Intuitively speaking, the original layers of I_i and I_j remain independent in I, and I effectively consists of t independent instances of MSVC (for details, see Theorem 3.25). We cannot use this approach for MSVC-Sum because the value of ℓ is "distributed" between the layers of I: For instance, if I_1 is a yes-instance but I_2 is a no-instance, we expect I to be a no-instance. But if the symmetric difference between the layers of I_1 is very small, and "I_1 uses very little ℓ", we can use the remaining ℓ for I_2, and I might become a yes-instance. Therefore, MSVC-Sum requires a different approach.

Theorem 4.5. *Let $I = (\mathcal{G}, k, \ell)$ be an instance of MSVC-Sum where $\ell = 0$. Then I has a bikernel of size at most $\mathcal{O}(k^2)$.*

Proof. The proof is analogous to that of Theorem 3.24 □

Theorem 4.6. *MSVC-Sum admits no kernel of size polynomial in k, even if $\ell \geq 2k \cdot (\tau - 1)$.*

Proof. We prove Vertex Cover AND-cross-composes into MSVC-Sum with $\ell \geq 2k \cdot (\tau - 1)$: Let $\mathcal{R}$ be a relation on the set which contains all possible valid inputs for Vertex Cover such that for Vertex Cover instances $I = (G_i = (V_i, E_i), k_i)$ and $J = (G_j = (V_j, E_j), k_j)$ one has $(I, J) \in \mathcal{R}$ if and only if $|V_i| = |V_j|$ and $k_i = k_j$. Now $\mathcal{R}$ is a polynomial equivalence relation because:

- It is obvious that $\mathcal{R}$ is an equivalence relation.
- Given two Vertex Cover instances $I = (G_i = (V_i, E_i), k_i)$ and $J = (G_j = (V_j, E_j), k_j)$, we can decide whether they belong to the same equivalence class in polynomial time by comparing V_i to V_j and k_i to k_j.
- For any finite set S of Vertex Cover instances, $\mathcal{R}$ partitions its elements into a number of classes that is polynomially bounded in the size of its largest element: Let $I = (G = (V, E), k)$ be the largest element of S. Let $I_1 = (G_1 = (V_1, E_1), k_1)$ be the element of S with the highest number of vertices in its graph, and let $I_2 = (G_2 = (V_2, E_2), k_2)$ be the element of S with the highest parameter k. Then the number of equivalence classes is at most $|V_1| \cdot k_2$.

Our AND-cross-composition algorithm now takes t instances $I_1 = (G_i, k)..., I_t = (G_t, k)$ of Vertex Cover which belong to the same equivalence class of $\mathcal{R}$ as its input and creates an instance of MSVC-Sum $I = (\mathcal{G}, k, \ell)$ as follows: Set $\ell = c$ with c

being an arbitrary integer equal to or greater than $2k \cdot (\tau - 1)$. The temporal graph $\mathcal{G}$ receives $G_1, ..., G_t$ as its layers (which implies $\tau = t$).

The parameter k is polynomially bounded in $\max_i |x_i| + \log t$ as I uses the same k as the I_i.

Assume that $I_i = (G_i, k)$ is a yes-instance of Vertex Cover for every $i \in \{1, ..., t\}$. Then there are vertex covers $S_1, ..., S_t$ of size at most k for $G_1, ..., G_t$. Moreover, we have $|S_i \triangle S_{i+1}| \leq 2k$ for every $i \in \{1, ..., t-1\}$ because regardless of the specific solution $\mathcal{S} = (S_1, ..., S_\tau)$ for any two sets S_i and S_{i+1} with $|S_i| \leq k$ and $|S_{i+1}| \leq k$ the maximum symmetric difference is $2k$. This implies that $\sum_{i=1}^{t-1} |S_i \triangle S_{i+1}| \leq 2k \cdot (t-1) \leq \ell$, and that I is a yes-instance of MSVC-Sum with $\ell \geq 2k \cdot (\tau - 1)$.

Assume that I is a yes-instance of MSVC-Sum with $\ell \geq 2k \cdot (\tau - 1)$. Then a solution $\mathcal{S} = (S_1, ..., S_\tau)$ exists such that every S_i is a vertex cover of size at most k of G_i. This implies that I_i is a yes-instance of Vertex Cover for every $i \in \{1, ..., t\}$.

We have now proven that Vertex Cover AND-cross-composes into MSVC-Sum with $\ell \geq 2k \cdot (\tau - 1)$, which for the latter proves that no kernel of size polynomial in k exists assuming that NP $\nsubseteq$ coNP / poly. □

Reduction Rule 4.7. *Let $I = (\mathcal{G}, k, \ell)$ be an instance of MSVC-Sum, and let be $G_1, ..., G_\tau$ its layers. Add an additional layer $G_0 = G_1$ before G_1.*

Lemma 4.8. *Reduction Rule 4.7 is correct and can be applied in linear time.*

Proof. Assume that I is a yes-instance of MSVC-Sum with solution $\mathcal{S} = (S_1, ..., S_\tau)$., and I' the same instance after applying Reduction Rule 4.7. Then $\mathcal{S}' = (S_0 = S_1, S_1, S_2, ..., S_\tau)$ is a solution to I': We know that S_1 is a vertex cover of size at most k the newly added layer $G_0 = G_1$, and that S_i is a vertex cover of size at most k of G_i for $\imath in\{1, ..., \tau\}$. Further, we have $\sum_{i=1}^{\tau-1} |S_i \triangle S_{i+1}| \leq \ell$ for every $\imath in\{1, ..., \tau - 1\}$, and we know that $S_0 = S_1$. Assume that I' is a yes-instance of MSVC-Sum with solution $\mathcal{S}' = (S_0 = S_1, S_1, S_2, ..., S_\tau)$, then $\mathcal{S} = (S_1, ..., S_\tau)$ is a solution to I because we have $\sum_{i=1}^{\tau-1} |S_i \triangle S_{i+1}| \leq \ell$ for every $\imath in\{0, ..., \tau - 1\}$ and S_i is a vertex cover of G_i for every $i \in \{1, ..., \tau\}$.

Obviously, copying a layer can be performed in linear time. □

Observation 4.9. *The AND-cross-composition given in Theorem 4.6 always outputs an instance $I = (\mathcal{G}, k, \ell)$ of MSVC-Sum with $\ell \geq 2k \cdot (\tau - 1)$, which rules out polynomial kernels only for this case. However, we can use Reduction Rule 4.7 to increase τ arbitrarily at the end of the AND-cross-composition, which allows us to apply the result tothe case of $2k \cdot (\tau - 1) > \ell > 0$.*

Theorem 4.10. *Any instance $I = (\mathcal{G}, k, \ell)$ of MSVC-Sum has a kernel of size at most $3k^2 \cdot \tau$.*

Proof. The proof is analogous to that of Theorem 3.30 □

Theorem 4.11. *Any instance $I = (\mathcal{G}, k, \ell)$ of MSVC-Sum where $\ell \geq 2k \cdot (\tau - 1)$ has a Turing kernel of size at most $\mathcal{O}(k^2 \cdot \tau)$.*

Proof. The proof is analogous to that of Theorem 3.32. The condition $\sum_{i=1}^{\tau-1} |S_i \triangle S_{i+1}| \leq \ell$ always holds when $\ell \geq 2k \cdot (\tau - 1)$ because regardless of the specific solution $\mathcal{S} = (S_1, ..., S_\tau)$ for any two sets S_i and S_{i+1} with $|S_i| \leq k$ and $|S_{i+1|} \leq k$ the maximum symmetric difference is $2k \leq \ell$, and thus we have $\sum_{i=1}^{\tau-1} |S_i \triangle S_{i+1}| \leq 2k \cdot (\tau - 1) \leq \ell$. □

Theorem 4.12. *Let $I = (\mathcal{G}, k, \ell)$ be an instance of MSVC-Sum where $|E_i| = 1$ for all layers G_i has a kernel of size at most $7\tau - 1$.*

Proof. If $k \geq \tau$ or $\ell \geq 2\tau - 2)$, then I is a yes-instance due to Lemma 3.15 and Lemma 3.18, so we can return a trivial yes-instance as kernel. Otherwise, use Reduction Rule 3.26 repeatedly as many times as possible. We receive an MSVC-Sum instance $K = (\mathcal{G}' = (V', \mathcal{E}, \tau), k, \ell)$ where Reduction Rule 3.26 is not applicable which means there is no vertex v so that for all layers $G_i = (V, E_i)$ one has $e \in E_i \Rightarrow v \notin e$. Thus, each remaining vertex must be contained in at least one temporal edge. We have $|\mathcal{E}| = \tau$ because we assumed $|E_i| = 1$ for each of the τ layers. Because each temporal edge can contain at most 2 vertices, and each vertex must be contained in at least one temporal edge, at most 2τ vertices remain. Also, by assumption we have $k < \tau$ and $\ell < 2\tau - 2$. For the size of K we get $|V| + |\mathcal{E}| + \tau + k + \ell \leq 2\tau + \tau + \tau + \tau + 2\tau - 2 = 7\tau - 2$. □

Lemma 4.13. *MSVC-Sum does not admit a kernelization on arbitrary layers for parameter τ.*

Proof. The proof is analogous to that of Lemma 3.33. □

Theorem 4.14. *An instance $I = (\mathcal{G}, k, \ell)$ of MSVC-Sum can be solved in $\mathcal{O}((1.2738^k + kn) \cdot \tau)$, where n is the size of the given instance if one has $\ell \geq 2k \cdot (\tau - 1)$. This implies fixed-parameter tractability for the parameter $k + \tau$.*

Proof. The proof is analogous to that of Theorem 3.35. □

Theorem 4.15. *MSVC-Sum with $|E_i| = 1$ for all layers G_i is fixed-parameter tractable for the parameter τ.*

Proof. MSVC-Sum on instances $I = (\mathcal{G} = (V, \mathcal{E}, \tau), k, \ell)$ with $|E_i| = 1$ for all of its layers G_i has a polynomial kernel for the parameter τ according to Theorem 4.12, which implies fixed-parameter tractability. □

Finally, we discuss the possibility of an FPT algorithm for MSVC-Sum. While we cannot use Algorithm 3.42 without modifications, we can assume that a slightly modified version of the same algorithm can solve any instance of MSVC-Sum.

Conjecture 4.16. *MSVC-Sum could fixed-parameter tractable with respect to parameter k, an instance can be decided in $\mathcal{O}(\tau \cdot (2^{2k^2+5k} \cdot k^2) + |V|^3)$. The proof idea is analogous to that of Theorem 3.38. However, we need to change the way the concept of reachability and the subroutine checkReachability work. Let $I = (\mathcal{G} = (V, \mathcal{E}, \tau), k, \ell)$ be an instance of MSVC with $\mathcal{G}$ having layers $G_1, ..., G_\tau$, let $S_1, ..., S_\tau$ be vertex covers of the respective layers, let $c, c', c'' \in \mathbb{N}$ and let $i, j \in \{1, ..., \tau\}$. Now a vertex set V_j is considered reachable* *with a "consumption" of c from a vertex set V_i if $i+1=j$ and $|V_i \triangle V_j| = c \leq \ell$, or if $i+1<j$, there is a vertex cover V_{i+1} of size at most k of G_{i+1}, $|V_i \triangle V_{i+1}| = c' \leq \ell$, and V_j is reachable from V_{i+1} with consumption $c'' \leq \ell$ such that $c = c' + c'' \leq \ell$. As soon we know that a vertex set is not reachable with a consumption of $c \leq \ell$, it is considered unreachable and deleted. The subroutine resolveDC is called in every iteration of the inner loop of checkReachability.*

5 Conclusion

We studied the computational complexity of MSVC and MSVC-Sum, provided upper and lower bounds for kernelization when possible, and developed an FPT algorithm for MSVC and the parameter k.

We proved NP-hardness for MSVC and many of its variants, as seen in Theorem 3.1. This is quite unsurprising, as the NP-hard Vertex Cover can be viewed as a corner case of MSVC, where $\tau = 1$. In general, we can divide all variations of MSVC into three subgroups: If $\tau = 1$ or $\ell = 0$, we have to cover every temporal edge with just a single vertex set, which means that the given instance can be viewed as an instance of Vertex Cover. If $\ell \geq 2k$, we can choose a completely new vertex set for every layer. In this case, the MSVC instance effectively degrades into τ separate, independent instances of Vertex Cover. In both cases, MSVC inherits properties of Vertex Cover, such as NP-hardness. The intermediate case of $\tau > 1$ and $0 < \ell < 2k$ is more interesting: No direct relationship with Vertex Cover can be found, and in some cases MSVC exhibits different behavior: For example, while Vertex Cover can be solved in linear time when the input graph is a tree, MSVC remains NP-hard on trees if $\tau > 1$ and $0 < \ell < 2k$, and even if we have $|E_i| = 1$ for every layer G_i. This is comparable to some of the results of Akrida et al. [Akr+18], who showed that TVC remains NP-hard even on star graphs. All of our complexity results are summed up in Table 1.1.

We studied kernelizations for MSVC and proved various lower and upper bounds for kernels and bikernels. Unsurprisingly, we found that MSVC does not admit a polynomial kernelization for the parameter τ, because as stated above, Vertex Cover can be considered as a corner case of MSVC with $\tau = 1$, and a polynomial kernel would allow a kernel of constant size for Vertex Cover. A notable exception is the case where every layer contains just a single edge. Further, we found that MSVC does not admit a polynomial kernel for the parameter k, unless in the case of $\ell = 0$ where MSVC is equivalent to Vertex Cover. The parameter $k + \tau$ is more interesting here: We proved the correctness of two data reduction rules which allow us to reduce the size of an instance to $3k^2 \cdot \tau$. As there are many studies on kernelizations of Vertex Cover, future research could aim to transfer these results to MSVC and reduce the kernel size further. All of our kernelization results are summed up in Table 1.2.

We discussed an FPT algorithm (Algorithm 3.42) for the parameter k, which runs in $\mathcal{O}(\tau \cdot (2^{2k^2+5k} \cdot k^2) + |V|^3)$. This makes it possible for us to solve instances of MSVC efficiently when k is small, even for high values of τ. Unfortunately, the proof of this (Lemma 3.48) uses several very coarse-grained estimates. If one were to find more accurate estimates, they might be able to show that the running time is in fact much lower. Also, because networks in real-world applications such as our introductory example of worm propagation are often large, it might be desirably to develop approximation algorithms as Akrida et al. [Akr+18] have done for the TVC problem: For high values of k, the running time of our algorithm is still relatively high. Further, we have provided optimizations which either reduce space complexity or grant the ability to extract a specific solution — so far no algorithm can perform both.

Finally, we discussed a variation of MSVC, MSVC-Sum. The results are mostly analogous to MSVC, however, we cannot use the same algorithm. In Conjecture 4.16 we suggest changes which could allow the FPT algorithm for MSVC (Algorithm 3.42) to be modified such that instances of MSVC-Sum can be solved; however, the details of such an algorithm are subject to future research.

Literature

[AK+04] F. N. Abu-Khzam, R. L. Collins, M. R. Fellows, M. A. Langston, W. H. Suters, and C. T. Symons. “Kernelization Algorithms for the Vertex Cover Problem: Theory and Experiments.” In: *ALENEX/ANALC* 69 (2004) (cit. on p. 11).

[Akr+18] E. C. Akrida, G. B. Mertzios, P. G. Spirakis, and V. Zamaraev. “Temporal Vertex Cover with a Sliding Time Window”. In: *arXiv preprint arXiv:1802.07103* (2018) (cit. on pp. 11, 51, 52).

[BG93] J. F. Buss and J. Goldsmith. “Nondeterminism within P^”. In: *SIAM Journal on Computing* 22.3 (1993), pp. 560–572 (cit. on p. 9).

[BJK14] H. L. Bodlaender, B. M. Jansen, and S. Kratsch. “Kernelization lower bounds by cross-composition”. In: *SIAM Journal on Discrete Mathematics* 28.1 (2014), pp. 277–305 (cit. on pp. 11, 15, 16, 27).

[CKJ01] J. Chen, I. A. Kanj, and W. Jia. “Vertex cover: further observations and further improvements”. In: *Journal of Algorithms* 41.2 (2001), pp. 280–301 (cit. on pp. 9, 28).

[CKX06] J. Chen, I. A. Kanj, and G. Xia. “Improved parameterized upper bounds for vertex cover”. In: *International Symposium on Mathematical Foundations of Computer Science*. Springer. 2006, pp. 238–249 (cit. on pp. 9, 35).

[DF95] R. G. Downey and M. R. Fellows. “Fixed-parameter tractability and completeness I: Basic results”. In: *SIAM Journal on Computing* 24.4 (1995), pp. 873–921 (cit. on p. 11).

[DFS99] R. G. Downey, M. R. Fellows, and U. Stege. “Parameterized complexity: A framework for systematically confronting computational intractability”. In: *Contemporary trends in discrete mathematics: From DIMACS and DIMATIA to the future*. Vol. 49. 1999, pp. 49–99 (cit. on p. 9).

[Fil+07] E. Filiol, E. Franc, A. Gubbioli, B. Moquet, and G. Roblot. “Combinatorial optimisation of worm propagation on an unknown network”. In: *International Journal of Computer Science* 2 (2007) (cit. on p. 9).

[FSS10] H. Fleischner, G. Sabidussi, and V. I. Sarvanov. “Maximum independent sets in 3-and 4-regular Hamiltonian graphs”. In: *Discrete Mathematics* 310.20 (2010), pp. 2742–2749 (cit. on p. 21).

[GNW05] J. Guo, R. Niedermeier, and S. Wernicke. “Parameterized complexity of generalized vertex cover problems”. In: *Workshop on Algorithms and Data Structures*. Springer. 2005, pp. 36–48 (cit. on p. 11).

[Hua] X Huang. “Applications of Parameterized Computation in Computational Biology”. In: () (cit. on p. 9).

[Kar72] R. M. Karp. “Reducibility among combinatorial problems”. In: *Complexity of computer computations*. Springer, 1972, pp. 85–103 (cit. on p. 21).

[Mic16] O. Michail. “An introduction to temporal graphs: An algorithmic perspective”. In: *Internet Mathematics* 12.4 (2016), pp. 239–280 (cit. on p. 11).

[MS16] O. Michail and P. G. Spirakis. “Traveling salesman problems in temporal graphs”. In: *Theoretical Computer Science* 634 (2016), pp. 1–23 (cit. on p. 11).

[VLL90] J. Van Leeuwen and J. Leeuwen. *Handbook of theoretical computer science*. Vol. 1. Elsevier, 1990 (cit. on pp. 13, 19, 22).

[Wu+14] H. Wu, J. Cheng, S. Huang, Y. Ke, Y. Lu, and Y. Xu. “Path problems in temporal graphs”. In: *Proceedings of the VLDB Endowment* 7.9 (2014), pp. 721–732 (cit. on p. 11).