

# Algorithms and Complexity for Edge Set Switching in Graphs

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

**Diplom-Ingenieur**

in

**Logic and Artificial Intelligence**

by

**Christoph Kern, BSc**

Registration Number 11904675

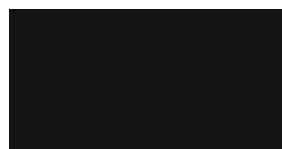
to the Faculty of Informatics

at the TU Wien

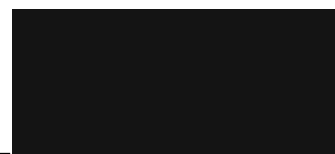
Advisor: Univ.Ass. Dipl.-Inf. Dr.rer.nat. Manuel Sorge

Assistance: Projektass. Dipl.-Ing. Alexander Firbas, BSc

Vienna, September 3, 2025



Christoph Kern



Manuel Sorge



# Declaration of Authorship

Christoph Kern, BSc

I hereby declare that I have written this Thesis independently, that I have completely specified the utilized sources and resources and that I have definitely marked all parts of the work - including tables, maps and figures - which belong to other works or to the internet, literally or extracted, by referencing the source as borrowed.

I further declare that I have used generative AI tools only as an aid, and that my own intellectual and creative efforts predominate in this work. In the appendix “Overview of Generative AI Tools Used” I have listed all generative AI tools that were used in the creation of this work, and indicated where in the work they were used. If whole passages of text were used without substantial changes, I have indicated the input (prompts) I formulated and the IT application used with its product name and version number/date.

Vienna, September 3, 2025

A solid black rectangular box used to redact the author's signature.

---

Christoph Kern



# Acknowledgements

I would like to sincerely thank my advisor, Manuel Sorge, for his support and guidance throughout the entire thesis process and for introducing me to the topic of this thesis in the first place. I am especially thankful for his patience with my many questions and for always being available to discuss ideas and provide feedback. In particular, our weekly meetings were invaluable in helping me stay on track and make steady progress. It is rare to find an advisor who is so approachable and dedicated to helping their students succeed.

I am also very grateful to Alex Firbas for his support as a co-advisor. He joined this project out of his own interest in the topic, and his ideas and suggestions proved to be extremely helpful. Like Manuel, he provided frequent and constructive feedback on my writing, which greatly improved the quality of this thesis.

Finally, I would like to thank Martin Nöllenburg for giving me the opportunity to get to know his research group a few years ago and for helping me find a suitable topic for my thesis within his group.



# Abstract

A switch graph is a graph with a variable edge set controlled by a collection of switches. Each switch has several positions, each corresponding to a subset of edges. A configuration selects one position per switch, which results in a simple graph where the edges are the union of the selected edge sets. This framework gives rise to a variety of switching problems, where the objective is to find a configuration such that the resulting graph satisfies a specified property. Several variants of these problems with different structural constraints on the switch graph have been studied in the literature.

This thesis focuses on connectivity-related switching problems, such as finding a configuration that yields a connected graph (global connectivity) or one that connects a specific pair of vertices (s-t-connectivity). We combine existing results with new complexity-theoretic insights and investigate how structural restrictions on switches affect computational hardness. Our results include a strong complexity dichotomy for s-t-connectivity and a polynomial-time algorithm for global connectivity based on matroid theory. Additionally, we introduce a weighted variant of the problem and analyze its complexity, particularly from a parameterized complexity perspective. For several restricted classes of switches, we establish  $W[1]$ -hardness parameterized by the maximum allowed weight of the solution. On the other hand, we present a fixed-parameter tractable algorithm using dynamic programming over tree decompositions.



# Contents

<b>Abstract</b>	vii
<b>Contents</b>	ix
<b>1 Introduction</b>	<b>1</b>
1.1 The Switching Problem	1
1.2 Adding Weights	2
1.3 Related Work	4
1.4 Our Results	5
<b>2 Preliminaries</b>	<b>9</b>
2.1 Common Graphs	9
2.2 Propositional Logic	10
2.3 Switch Graphs	11
2.4 Parameterized Complexity	15
2.5 Matroid Theory	16
<b>3 Global Connectivity Switching and Matroids</b>	<b>17</b>
3.1 The Matroid Intersection Problem	18
3.2 Defining the Matroids	19
3.3 Adaption to the Weighted Variant	21
<b>4 Connection to Transition-Compatible Paths</b>	<b>25</b>
4.1 From Transition Graphs to Switches	26
4.2 From Switches to Transition Graphs	30
4.3 Dichotomy for $\{1\}^*$ -Switches	33
<b>5 The Complexity of (Weighted) Satisfiability</b>	<b>35</b>
5.1 Unweighted Satisfiability	37
5.2 Weighted Satisfiability	39
<b>6 <math>s</math>-<math>t</math>-Connectivity Switching and Satisfiability</b>	<b>53</b>
6.1 Reduction from Satisfiability	54
6.2 Unweighted Switching	56
	ix

6.3	Weighted Switching	57
<b>7</b>	<b>Global Connectivity Switching and Satisfiability</b>	<b>65</b>
7.1	Initial Intractability Results	65
7.2	Intractability for Star Switches	66
7.3	Intractability for Vertices of Bounded Switch Degree	69
<b>8</b>	<b><math>s</math>-<math>t</math>-Disconnectivity Switching and Satisfiability</b>	<b>73</b>
8.1	Connection to $s$ - $t$ -Connectivity Switching	73
8.2	Intractability for Star Switches	76
<b>9</b>	<b>FPT Algorithms via Dynamic Programming</b>	<b>81</b>
9.1	Preliminaries	82
9.2	Algorithm for Global Connectivity	84
9.3	Adaption for $s$ - $t$ -(Dis)Connectivity	96
9.4	Global Connectivity with Bounded Radius	97
<b>10</b>	<b>Conclusion</b>	<b>101</b>
	<b>Overview of Generative AI Tools Used</b>	<b>105</b>
	<b>List of Figures</b>	<b>107</b>
	<b>List of Tables</b>	<b>109</b>
	<b>List of Algorithms</b>	<b>111</b>
	<b>Bibliography</b>	<b>113</b>

# Introduction

The concept of *switch graphs*, along with related combinatorial structures, has been extensively explored in several papers published over the past few decades [11, 17, 7, 8, 9, 19]. While the definitions vary depending on the specific application, the core idea remains consistent: a switch graph is an ordinary graph augmented by a set of *switches*. Each switch can assume multiple states, with each state introducing different modifications to the graph. The literature suggests a range of practical applications of switch graphs, including solving Boolean equation systems [7], managing valves in a system of pipes [8, 9], and SNP genotyping [19]. The general objective is to determine a configuration of these switches such that the resulting graph satisfies a desired graph property, for instance, connectivity, planarity, etc.

In this thesis, we introduce a general definition of switch graphs capable of encompassing the majority of variants discussed in the literature, with the aim of providing a comprehensive complexity analysis of the associated decision problems under various structural constraints. Furthermore, we extend the model to include weights on switch positions, a concept not previously examined in the literature but with significant practical applications. In this chapter, in addition to briefly introducing these two concepts, we review the existing literature on switch graphs, outline our contributions, and provide an overview of the thesis structure.

## 1.1 The Switching Problem

A switch graph consists of a set of vertices together with a set of *switches*, where each switch is a finite sequence of *positions*, and each position specifies a set of edges between the vertices. A *configuration* of the switch graph selects exactly one position for every switch, thereby inducing a simple graph on the same vertex set with edges taken from the chosen positions. The central question we study is whether there exists a configuration of switches such that the resulting graph belongs to a given graph family (or equivalently,

satisfies a certain graph property). For example, if switches represent valves in a system of pipes, one might be interested in finding a configuration that allows water to flow from one point to another, corresponding to the family of graphs that contain a path between two distinguished vertices [8, 9]. We refer to this general problem as  $\Pi$ -SWITCHING, where  $\Pi$  denotes the target graph family.

By picking an appropriate family of graphs  $\Pi$  and additionally imposing specific constraints on the structure of each switch in the input, most of the switch graph variants discussed in the literature can be expressed using  $\Pi$ -SWITCHING. For instance, the model of Katz et al. [11] restricts switches to one edge per position, all sharing a common endpoint. This unified perspective allows us to study the complexity of these problems under a single framework and identify common algorithmic techniques that can be applied to solve them. Furthermore, it enables us to investigate under which conditions the problem becomes tractable or remains intractable in a more general setting, thus providing a deeper understanding of the underlying combinatorial structure. This distinction is important when implementing solutions in practice: tractable problems can often be solved efficiently without the need for heuristics or approximations, whereas intractable problems may require more sophisticated techniques or may not be solvable at all within a reasonable amount of time.

Since much of the existing literature has focused on connectivity-related problems in switch graphs, this thesis concentrates on three specific target graph families: (i) connected graphs (global connectivity), (ii) graphs containing an  $s$ - $t$  path between two distinguished vertices  $s$  and  $t$  ( $s$ - $t$ -connectivity), and (iii) graphs where  $s$  and  $t$  are disconnected ( $s$ - $t$ -disconnectivity). These correspond to the switching problems GCON-SWITCHING, STCON-SWITCHING, and STDISCON-SWITCHING, respectively.

In our complexity analysis, we investigate structural properties of switches that may render these problems tractable, or otherwise establish their NP-hardness. Specifically, we examine restrictions on the number of positions per switch, the number of edges per position, and the graph structure induced by the union of all edges belonging to a single switch, as these constraints are common in the literature and frequently arise in practical applications (e.g., railway switches that choose between two edges sharing a common endpoint). Beyond individual switches, we also study global restrictions on the switch graph itself, in particular the *union graph*, formed by the union of all edges across all switch positions. We will resort to matroid theory and dynamic programming for our algorithmic results, while our hardness proofs primarily utilize reductions from NP-hard problems, in particular from restricted variants of SAT.

## 1.2 Adding Weights

While our proposed model has been studied (to some extent) under various restrictions, a weighted variant of the problem has not yet been explored in the literature. A natural extension, which we refer to as WEIGHTED- $\Pi$ -SWITCHING, assigns a weight to each switch position. The goal is to find a configuration that not only satisfies the desired graph

property but also minimizes the total accumulative weight of the selected switch positions. This has vast practical applications; for instance, in railway systems, one may wish to minimize the number of switches that need to be flipped to achieve a configuration that allows a train to travel between two points. In this context, each switch is assigned a weight of zero for its current position and a weight of one for the alternative position, so minimizing the total weight corresponds to minimizing the number of switches that must be flipped. We call this restricted variant, where each switch has two positions — an off-position of weight zero and an on-position of weight one — **ONOFF-II-SWITCHING**, which we frequently consider when studying intractability.

Further motivating the study of **WEIGHTED-II-SWITCHING** as well as **ONOFF-II-SWITCHING** is that many familiar optimization problems can naturally be reduced to them. As an example, consider **CLUSTER EDITING**, where the task is to transform a given simple graph into a disjoint union of cliques using the minimum number of edge modifications (insertions or deletions) [2]. This problem can be easily reduced to **ONOFF-II-SWITCHING** with  $\Pi$  being the family of disjoint unions of cliques. For each pair of vertices in the original graph, we introduce an on-off switch with one position containing no edge and the other containing the edge between the two vertices. If the edge already exists in the input graph, it is placed in the off-position, so switching it on corresponds to deleting the edge. Conversely, if the edge is absent in the input graph, it is placed in the on-position, and switching it on corresponds to inserting the edge.

Another problem that also served as a key motivation for studying the weighted variant is **ROBOT CONFIGURATION** [1], where the task is to determine the shortest sequence of moves that transforms one configuration of a robot into another. To solve this problem, one can reduce it to a temporal variant of **ONOFF-GCON-SWITCHING**.

Note that **WEIGHTED-II-SWITCHING** and **ONOFF-II-SWITCHING** are technically not formulated as optimization problems but rather as decision problems, where the input additionally includes a maximum allowed weight and we ask whether there exists a configuration whose total weight does not exceed this bound while also satisfying the desired graph property. We do this because it is often more convenient to work with a decision problem for complexity analysis.

Building on the insights from the unweighted setting, we investigate the complexity of **WEIGHTED-II-SWITCHING** and **ONOFF-II-SWITCHING** for global connectivity,  $s$ - $t$ -connectivity, and  $s$ - $t$ -disconnectivity under various structural restrictions on the switches and the union graph, analogous to those considered in the unweighted case. It is evident that the weighted problem is at least as hard as its unweighted counterpart, if not harder. Nevertheless, many algorithmic techniques developed for the unweighted setting can be adapted to the weighted case. Since intractability in the unweighted problem implies intractability for the weighted variant, our focus shifts toward parameterized complexity. Specifically, we examine whether the problem becomes tractable when certain parameters, such as the maximum total weight or the maximum number of allowed switch positions, are bounded in size.

### 1.3 Related Work

The switch graph model proposed by Katz et al. [11] served as a foundation for the ideas explored in this thesis. While their model is similar to ours, their switches are more constrained: each switch has exactly one edge for every position, and all positions share a common endpoint. In their model, they presented polynomial-time algorithms for both global connectivity and  $s$ - $t$ -connectivity, leveraging matroid and matching theory, respectively. In Chapter 3, we will show how their matroid-based approach can be adapted to handle more general switches and extended to work with weights. Conversely, they demonstrated that finding a planar, triangle-free, or bipartite configuration is NP-hard. Meinel [17] also used this model, focusing on a variant of  $s$ - $t$ -connectivity where all reachable edges must be part of a path from  $s$  to  $t$ . They proved that, in its most general form, this problem is NP-complete.

The model described by Groote and Ploeger [7] closely resembles that of Katz et al. [11], with the additional constraint that each switch must have exactly two positions. They illustrated how this model can represent Boolean equation systems and analyze several target graph families, including  $s$ - $t$ -connectivity and  $s$ - $t$ -disconnection. Moreover, they introduced integer labels for vertices and defined a collection of labeled target graph families involving loops.

Reinhardt [19] investigated  $s$ - $t$ -connectivity using a switch graph model inspired by train tracks. In this model, each vertex in an ordinary graph represents a switch, with exactly one adjacent edge that must be traversed on any path passing through it. This design reflects the idea that trains can only make smooth turns when crossing a switch. Reinhardt showed that, for the undirected case, the problem is solvable in polynomial time, while in the directed case, it becomes NP-hard. This variant also has practical applications in computational biology, specifically in SNP genotyping, by linking the switch graph model to certain matching problems.

Huckenbeck [8, 9] developed a switch graph model inspired by valve systems in breweries. Here, each vertex in a directed graph represents a valve, and each adjustment of the valve connects one incoming arc to a set of outgoing arcs. Instead of calling this structure a switch graph, they coined the term *valve graph*. Huckenbeck showed that the complexity of finding an  $s$ - $t$ -path in a valve graph depends on the structure of the valves. They identified a class of valves for which the problem can be solved in polynomial time, whereas in the general case, it is NP-hard [8]. They also performed a similar analysis for  $s$ - $t$ -disconnection [9].

Finally, Szeider [20] proposed another model focused on  $s$ - $t$ -connectivity. In this model, each vertex of an ordinary graph is associated with a transition graph, where the vertices of the transition graph correspond to the edges incident to the original vertex. An edge in the transition graph indicates that the corresponding edges are allowed to appear consecutively in the desired  $s$ - $t$ -path. Szeider identified a family of transition graphs for which the problem is solvable in polynomial time, whereas in the general case, the problem is NP-hard. Notably, many of these transition graphs can be effectively

Symbol	Meaning
$n$	Number of vertices
$m$	Total number of edges across all switches
$k$	Maximum number of switches set to the on-position
$T$	Time complexity of circuit-finding oracle
$\Delta$	Maximum number of switches incident to a vertex
tw	Treewidth of the union graph
$s$	Maximum number of positions of a switch

Table 1.1: Variables and parameters used in the complexity results

represented using regular switches in our switch graph model, which we will explore in more detail in Chapter 4, as this connection leads to strong complexity results for STCON-SWITCHING.

## 1.4 Our Results

In this thesis, we study the tractability of II-SWITCHING, WEIGHTED-II-SWITCHING, and ONOFF-II-SWITCHING for global connectivity,  $s$ - $t$ -connectivity, and  $s$ - $t$ -disconnectivity under various structural restrictions on the switches and the union graph. To describe switch sizes, we use the notation  $\{a, b, c, \dots\}$  to indicate that a switch has at most  $a$  edges in one position, at most  $b$  in another, and so on. If switches may have an arbitrary number of positions but at most one edge per position, we write  $\{1\}^*$ . We also impose restrictions on the *switch-induced graph* of each switch, that is, the graph obtained from the union of all edges across the positions of that switch, requiring it, for instance, to be connected, complete multipartite (c.m.p.), or a star. Likewise, we restrict the structure of the union graph — the graph formed by all edges across all switches and positions — considering cases such as paths, series-parallel graphs (SP), planar graphs, or star graphs. The main results are summarized in Tables 1.2 to 1.4, with Fig. 1.1 providing a visual overview of the complexity landscape based on the switch structure. A formal definition of the switching problems, the structural restrictions on switch graphs, and other concepts needed for the proofs are provided in Chapter 2.

In Chapter 3, we revisit the results of Katz et al. [11], who showed that GCON-SWITCHING is solvable in polynomial time when restricted to  $\{1\}^*$ -star switches. Their result relies on matroid theory — matroids being set systems satisfying specific axioms, extensively studied and equipped with powerful algorithmic techniques (more details on matroids in Chapter 2). We show that the restriction to star-switches is unnecessary and, at the same time, extend the result to the weighted variant WEIGHTED-GCON-SWITCHING. Moreover, for the unweighted variant, we relax the switch constraints even further, demonstrating that it suffices for the positions of each switch to form a matroid-like structure.

Switches	Union Graph	Variant	Runtime/Complexity	Reference
Matroid	-	Unweighted	$\mathcal{O}(mn \cdot (n + T + \log m))$	Theorem 1
$\{1\}^*$	-	Unweighted	$\mathcal{O}(mn \cdot (n + \log m))$	Corollary 1
		Weighted		Theorem 2
Connected	-	Weighted	FPT (tw, $\Delta$ , $s$ )	Theorem 30
$\{2, 1\}$	Path	Unweighted	NP-hard	Theorem 20
		OnOff	W[1]-hard (k)	Theorem 21
$\{2, 1\} \cap \text{Star}$	Star	Unweighted	NP-hard	Theorem 22
		OnOff	W[1]-hard (k)	Theorem 23
$\{2, 1\} \cap \text{Star}$	Planar	Unweighted	para-NP-hard ( $\Delta$ )	Theorem 24

Table 1.2: Complexity results for GCON-SWITCHING, WEIGHTED-GCON-SWITCHING, and ONOFF-GCON-SWITCHING under various restrictions on switches and the union graph (see Table 1.1 for the meaning of variables and parameters).

In Chapter 4, we turn to STCON-SWITCHING. Specifically, we investigate the connection between our switch graph model restricted to  $\{1\}^*$ -switches and the transition graph model introduced by Szeider [20]. This connection leads to a sharp complexity dichotomy for STCON-SWITCHING under  $\{1\}^*$ -switches: if the set of admissible switch-induced graphs is limited to complete multipartite graphs, then the problem is solvable in linear time; otherwise, it is NP-hard.

To better understand under which conditions II-SWITCHING becomes NP-hard across the three target graph families, we employ reductions from SAT, where restrictions on the propositional formula, such as bounding variable occurrences or formula depth, naturally translate into structural restrictions on the switches and the union graph. Consequently, in Chapter 5 we study several NP-hard variants of SAT as starting points for our reductions. For the weighted setting, WSAT (a weighted variant of SAT) reduces to ONOFF-II-SWITCHING in a similar way, so we also examine the hardness of several WSAT variants. We focus on those unlikely to be fixed-parameter tractable when parameterized by the number of variables set to true, since this directly implies the same for ONOFF-II-SWITCHING when parameterized by the number of switches set to the on-position. To this end, we show that several restricted variants of WSAT are W[1]-hard via a parameterized reduction from PARTITIONED SUBGRAPH ISOMORPHISM, a well-known W[1]-hard problem. Along the way, we also resolve an open question posed by Kanj and Szeider [10] regarding the complexity of a particular constrained variant of WSAT.

In Chapters 6 to 8, we build on the insights from Chapter 5 to analyze the complexity of II-SWITCHING and ONOFF-II-SWITCHING for global-connectivity,  $s$ - $t$ -connectivity, and  $s$ - $t$ -disconnectivity. In each case, we present a reduction from SAT, which can be adapted directly to the weighted setting. Most notably, we show that for all three target graph families, II-SWITCHING and ONOFF-II-SWITCHING are NP-hard and W[1]-hard, respectively, even when restricted to  $\{2, 1\}$ -star switches. In the case of GCON-

Switches	Union Graph	Variant	Runtime/Complexity	Reference
$\{1\}^* \cap \text{c.m.p.}$	-	Unweighted	$\mathcal{O}(n + m)$	Theorem 5
$\{1\}^* \cap \mathcal{A}$	-	Unweighted	NP-hard	Theorem 6
Connected	-	Weighted	FPT ( $tw, \Delta, s$ )	Theorem 31
$\{1, 1\}$	SP	Unweighted	NP-hard	Theorem 15
		OnOff	W[1]-hard ( $k$ )	Theorem 18
$\{2, 1\}$	Path	Unweighted	NP-hard	Theorem 14
		OnOff	W[1]-hard ( $k$ )	Theorem 17
$\{2, 1\} \cap \text{Star}$	-	Unweighted	NP-hard	Theorem 4
		OnOff	W[1]-hard ( $k$ )	Theorem 19
Any	SP	OnOff	W[SAT]-hard ( $k$ )	Theorem 16

Table 1.3: Complexity results for STCON-SWITCHING, WEIGHTED-STCON-SWITCHING, and ONOFF-STCON-SWITCHING under various restrictions on switches and the union graph (see Table 1.1 for the meaning of variables and parameters).

SWITCHING, we further strengthen this by proving NP-hardness even when the union graph is planar and the number of switches incident to each vertex is bounded by a constant.

In Chapter 9, we complement the hardness results with fixed-parameter algorithms for WEIGHTED-II-SWITCHING across all three target graph families, restricted to connected switches and parameterized by the treewidth of the union graph (a measure of tree-likeness), the maximum number of switches incident to a vertex, and the maximum number of positions per switch. These algorithms rely on dynamic programming over a tree decomposition of the union graph. We also discuss potential applications of this result for designing a fixed-parameter algorithm on planar union graphs without assuming bounded treewidth, using a win-win approach motivated by the ROBOT CONFIGURATION problem introduced earlier.

Finally, in Chapter 10, we summarize the main findings and outline directions for future research.

As previously mentioned, the results established in Chapters 3 to 9 are summarized in Tables 1.2 to 1.4 for the three target graph families studied in this thesis. In addition, Fig. 1.1 gives a visual overview of the complexity landscape under various switch restrictions. The figure depicts a Hasse diagram, with the most general case of arbitrary switches at the top and the most restricted cases at the bottom. Edges between two switch restrictions represent containment, meaning the lower restriction is a special case of the upper one. Consequently, hardness results propagate upwards, while tractability results propagate downwards.

Switches	Union Graph	Variant	Runtime/Complexity	Reference
Connected	-	Weighted	FPT $(tw, \Delta, s)$	Theorem 32
$\{1, 1\}$	SP	Unweighted	NP-hard	Theorem 25
		OnOff	W[1]-hard $(k)$	Theorem 27
$\{2, 1\} \cap \text{Star}$	-	Unweighted	NP-hard	Theorem 28
		OnOff	W[1]-hard $(k)$	Theorem 29
Any	SP	OnOff	W[SAT]-hard $(k)$	Theorem 26

Table 1.4: Complexity results for  $\text{STCON-SWITCHING}$ ,  $\text{WEIGHTED-STCON-SWITCHING}$ , and  $\text{ONOFF-STCON-SWITCHING}$  under various restrictions on switches and the union graph (see Table 1.1 for the meaning of parameters).

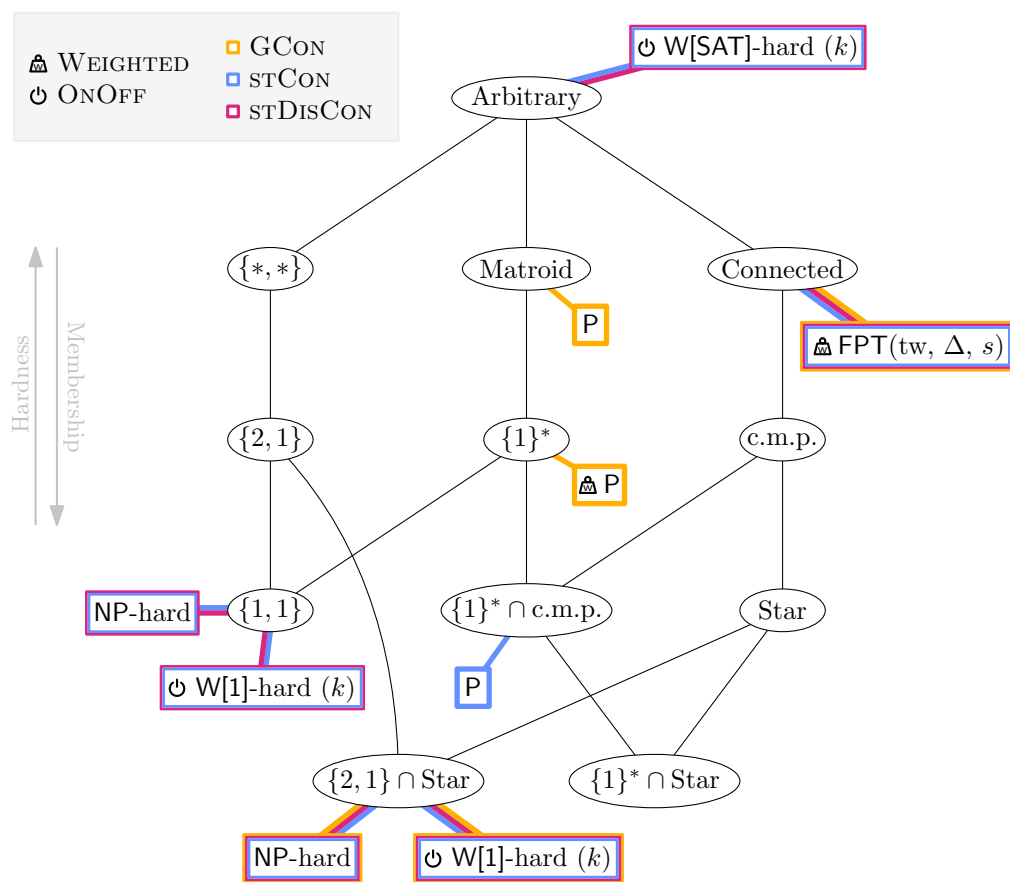


Figure 1.1: An overview of the complexity results for  $\Pi\text{-SWITCHING}$ ,  $\text{WEIGHTED-}\Pi\text{-SWITCHING}$ , and  $\text{ONOFF-}\Pi\text{-SWITCHING}$  for global connectivity,  $s$ - $t$ -connectivity, and  $s$ - $t$ -disconnectivity under various restrictions on switches (see Table 1.1 for the meaning of parameters). Edges between two switch restrictions represent containment, meaning the lower restriction is a special case of the upper one. Consequently, hardness results propagate upwards, while tractability results propagate downwards.

# Preliminaries

We begin with some basic definitions. For any  $n \in \mathbb{Z}$ , let  $[n]$  denote the set  $\{i \mid 1 \leq i \leq n\}$ . Observe that  $[n] = \emptyset$  if  $n < 1$ .

Let  $X$  be a sequence over some universe  $\mathcal{U}$ . We denote the  $i$ -th element of  $X$  by  $X[i]$  and the length (or size) of  $X$  by  $|X|$ . For some  $u \in \mathcal{U}$ , the number of occurrences of  $u$  in  $X$  is denoted by  $|X|_u$ . The notation for size and number of occurrences of an element extends naturally to multisets. The latter can also be applied on discrete functions  $f$  to indicate how often an element in the codomain is mapped from elements of the domain, i.e.,  $|f|_u = |f^{-1}(u)|$ .

Given a binary relation  $R \subseteq \mathcal{U}^2$  over some universe  $\mathcal{U}$ , we denote  $R^+$  as the transitive closure of  $R$ . Suppose  $R$  is reflexive, symmetric and transitive, then it is an *equivalence relation*. In that case, for some  $u \in \mathcal{U}$ ,  $R[u] = \{v \in \mathcal{U} \mid (u, v) \in R\}$  denotes the equivalence class of  $u$  under  $R$ . Furthermore, let  $\mathcal{U}/R$  be the set of equivalence classes of  $\mathcal{U}$  by  $R$ , that is,  $\mathcal{U}/R = \{R[u] \mid u \in \mathcal{U}\}$ .

## 2.1 Common Graphs

An *edge*  $e$  over a vertex set  $V$  is a set of two distinct vertices in  $V$ , formally defined as  $e = \{u, v\}$  for  $u, v \in V$  with  $u \neq v$ . For better readability, we often represent an edge  $\{u, v\}$  as  $uv$ . A *simple graph*  $G$  consists of a vertex set  $V(G)$  and an edge set  $E(G)$  over  $V(G)$ . For several definitions, we additionally require that the graph possess a distinguished *source* vertex  $s(G)$  and *target* vertex  $t(G)$ , often referred to as *terminals*.

For a vertex  $v \in V(G)$ , let  $N_G(v)$  denote the set of *neighbors* of  $v$  in  $G$ , that is,  $N_G(v) = \{u \in V(G) \mid uv \in E(G)\}$ . The *degree* of a vertex  $v$  in  $G$  is defined as  $\deg_G(v) = |N_G(v)|$ . Moreover, let  $E_G(v)$  denote the set of incident edges of  $v$ , that is,  $E_G(v) = \{uv \mid u \in N_G(v)\}$ . For a vertex set  $U \subseteq V(G)$ , let  $G[U]$  be the subgraph of  $G$  induced by  $U$ , i.e.,  $V(G[U]) = U$  and  $E(G[U]) = E(G) \cap \binom{U}{2}$ .

A *path* of length  $\ell \geq 0$  between two vertices  $u, v \in V(G)$  is a sequence of  $\ell + 1$  distinct vertices  $u = w_0, w_1, \dots, w_\ell = v$  of  $V(G)$  such that  $w_{i-1}w_i \in E(G)$  for all  $i \in [\ell]$ . The *reachability relation*  $R(G)$  of graph  $G$  contains every pair  $(u, v) \in V(G)^2$  of vertices where there is a path between  $u$  and  $v$ . Observe that  $R(G)$  is an equivalence relation and that  $V(G)/R(G)$  forms a partition where each set is a maximal connected component of  $G$ . Graph  $G$  is called (*globally*) *connected* if there is a path between every pair of vertices, or equivalently, if  $R(G) = V(G)^2$ .

The *union* of two graphs  $G_1$  and  $G_2$  is the union of their vertex and edge sets. Unless stated otherwise, we assume that  $V(G_1)$  and  $V(G_2)$ , as well as  $E(G_1)$  and  $E(G_2)$  are disjoint. However, if a vertex  $u \in V(G_1)$  is *identified* with a vertex  $v \in V(G_2)$ , meaning that  $u = v$ , then  $u$  and  $v$  are merged into a single vertex in the resulting graph.

A *multigraph*  $G$  is a graph that allows multiple edges between the same pair of vertices. Consequently, the edge set  $E(G)$  is a multiset of edges over  $V(G)$ . With additional distinguished terminals,  $G$  is called *series-parallel* if it is a *single-edge graph*, consisting of a source vertex  $s(G)$  connected to a target vertex  $t(G)$  by a single edge, or if it is the composition of two series-parallel graphs  $G_1$  and  $G_2$  in one of the following ways:

- *Series composition*: Formed by taking the union of  $G_1$  and  $G_2$  by identifying  $s(G) = s(G_1)$ ,  $t(G_1) = s(G_2)$ , and  $t(G_2) = t(G)$ .
- *Parallel composition*: Formed by taking union of  $G_1$  and  $G_2$  by identifying  $s(G) = s(G_1) = s(G_2)$  and  $t(G) = t(G_1) = t(G_2)$ .

A *tree*  $T$  is a connected graph without cycles. A *rooted tree* is a tree with a distinguished vertex  $r(T)$  called the *root*. This introduces a natural parent-child and ancestor-descendent relationship between vertices, with the root being the ancestor of all other vertices.

## 2.2 Propositional Logic

Let  $\mathcal{V}$  be a countably infinite set of *propositional variables*. For the purpose of this thesis, the set  $\Phi$  of *propositional formulas* is defined inductively as follows:

1. Any propositional variable  $x \in \mathcal{V}$  is in  $\Phi$ .
2. If  $\varphi \in \Phi$ , then  $\neg\varphi \in \Phi$ .
3. If  $\varphi, \psi \in \Phi$ , then  $(\varphi \wedge \psi) \in \Phi$  and  $(\varphi \vee \psi) \in \Phi$ .

We omit other connectives such as implication ( $\Rightarrow$ ), equivalence ( $\Leftrightarrow$ ), as they can be rewritten using the basic connectives above. However, they may still be used as syntactic sugar to enhance readability. For example, the implication  $\varphi \Rightarrow \psi$  is simply another way of writing  $\neg\varphi \vee \psi$ .

For a formula  $\varphi \in \Phi$ , let  $\mathcal{V}(\varphi)$  denote the set of variables in  $\varphi$ . A *truth assignment* or *interpretation* of  $\varphi$  is a function  $I: \mathcal{V}(\varphi) \rightarrow \{0, 1\}$  that assigns each variable  $x \in \mathcal{V}(\varphi)$  a truth value, where 1 represents *true* and 0 represents *false*. We extend the definition of  $I$  to *evaluate* formulas inductively as follows:

1.  $I(\neg\psi) = 1 - I(\psi)$ .
2.  $I(\psi \wedge \gamma) = \min\{I(\psi), I(\gamma)\}$
3.  $I(\psi \vee \gamma) = \max\{I(\psi), I(\gamma)\}$ .

Let  $\mathcal{I}(\varphi)$  denote the set of all possible interpretations of  $\varphi$ . Formula  $\varphi$  is *satisfiable* if there exists a truth assignment  $I \in \mathcal{I}(\varphi)$  such that  $I(\varphi) = 1$ . For a class of propositional formulas  $F \subseteq \Phi$ , we define  $\text{SAT}(F)$  as the problem of determining whether a given formula  $\varphi \in F$  is satisfiable. In the weighted variant,  $\text{WSAT}(F)$ , the input includes an additional integer  $k$ . The problem asks whether there exists a truth assignment  $I \in \mathcal{I}(\varphi)$  such that  $I(\varphi) = 1$  and the *weight* of  $I$ , defined as the number of variables set to true or simply  $|I|_1$ , is at most  $k$ .

## 2.3 Switch Graphs

A *switch*  $S$  is a finite, non-empty sequence of switch positions, where each position is a set of edges over an underlying vertex set  $V$ . It must contain at least one position that is not empty. A graph with switches instead of edges is what we call a *switch graph*. Formally, a switch graph  $\hat{G}$  consists of a vertex set  $V(\hat{G})$  and a set of switches  $\mathcal{S}(\hat{G})$  over  $V(\hat{G})$ . As with simple graphs, certain definitions require additional distinguished terminals  $s(\hat{G})$  and  $t(\hat{G})$ . Fig. 2.1 (left side) shows an example of a switch graph  $\hat{G}$ , demonstrating the visualization conventions for switch graphs used throughout this thesis.

A *configuration* of  $\hat{G}$  is a function  $c: \mathcal{S}(\hat{G}) \rightarrow \mathbb{N}_{>0}$  that maps each switch  $S \in \mathcal{S}(\hat{G})$  to a natural number  $i \in [|S|]$  representing one of its switch positions. In other words, the configuration  $c$  chooses the edges of exactly one position for each switch, resulting in the so called *configured switch graph*. Notated by the expression  $\hat{G} \circ c$  and read as  $\hat{G}$  *configured by*  $c$ , it is a simple graph where  $V(\hat{G} \circ c) = V(\hat{G})$  and  $E(\hat{G} \circ c) = \bigcup_{S \in \mathcal{S}(\hat{G})} S[c(S)]$ . Fig. 2.1 (right side) shows a possible configuration  $c$  of the illustrated switch graph  $\hat{G}$ . If terminals were defined for the switch graph, they are inherited by the configured switch graph, i.e.,  $s(\hat{G} \circ c) := s(\hat{G})$  and  $t(\hat{G} \circ c) := t(\hat{G})$ . Furthermore, we denote  $\mathcal{C}(\hat{G})$  to be the set of all configurations of  $\hat{G}$ .

Frequently, it is desirable to have edges in the switch graph that are present in any configuration. This can be accomplished by including *static edges*. A static edge is simply a switch  $S_e$  containing exactly a single position of a single edge  $e$ , that is,  $S_e = (\{e\})$ .

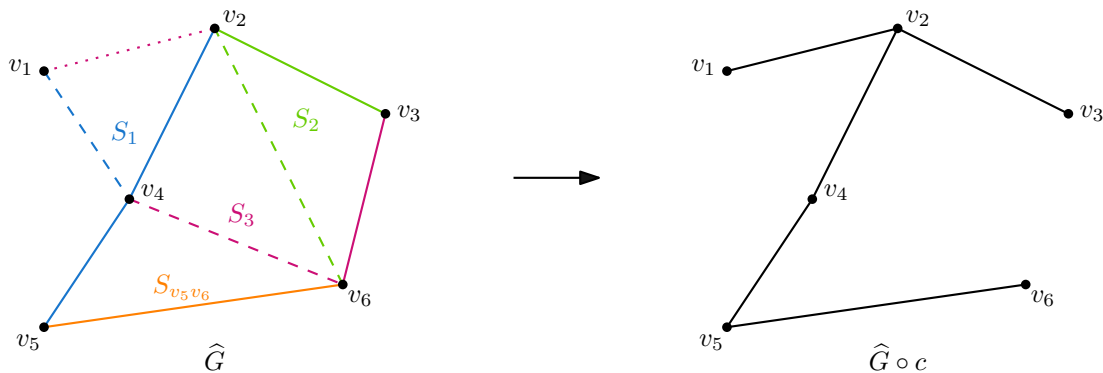


Figure 2.1: The left shows a switch graph  $\widehat{G}$  with four switches (including one static edge):  $S_1 := (\{v_2v_4, v_4v_5\}, \{v_1v_4\})$ ,  $S_2 := (\{v_2v_3\}, \{v_2v_6\})$ ,  $S_3 := (\{v_3v_6\}, \{v_4v_6\}, \{v_1v_2\})$ ,  $S_{v_5v_6} = (\{v_5v_6\})$ . Each color represents a different switch, while the different dash styles represent the different switch positions. The right shows  $\widehat{G}$  configured by  $c$  where  $c(S_1) = c(S_2) = c(S_{v_5v_6}) = 1$  and  $c(S_3) = 3$ .

### 2.3.1 Structural Properties

For several proofs in this thesis, we need better tools to study the structure of switch graphs. Moreover, tools that allow us to constrain the structure of switch graphs are used to obtain more nuanced results. To achieve all this, we introduce additional notation and definitions to better characterize different structural properties of switch graphs.

First of all, consider some switch  $S$ . Let  $E(S)$  be the combined set of edges occurring in  $S$ , that is,  $E(S) := \bigcup_{i \in [|S|]} S[i]$ . Additionally, let  $V(S)$  be the set of vertices that are incident to an edge in  $E(S)$ . Together, these form the vertex and edge set of the *switch-induced graph*  $G(S)$ , where  $V(G(S)) := V(S)$  and  $E(G(S)) := E(S)$ . Note that  $G(S)$  never contains isolated vertices.

Switch-induced graphs enable us to restrict the type of switches used in the switch graph via classical graph properties. Specifically, given a family of graphs  $\Pi$  closed under isomorphisms and containing no isolated vertices, we say that a switch  $S$  is a  $\Pi$ -switch if  $G(S) \in \Pi$ . For example, a common family of graphs used throughout this thesis is the set of star graphs, denoted by  $K_{1,\star} := \{K_{1,i} \mid i \in \mathbb{N}_{>0}\}$ . We refer to a switch as a *star switch* if it is a  $K_{1,\star}$ -switch. For example, in Fig. 2.1, switches  $S_1$  and  $S_{v_5v_6}$  are star switches. Similarly, we denote by  $\Pi_{\text{Conn}}$  the family of all connected graphs without isolated vertices, and refer to a switch as a *connected switch* if it is a  $\Pi_{\text{Conn}}$ -switch. Except for switch  $S_3$ , every switch in  $\widehat{G}$  illustrated in Fig. 2.1 is a connected switch.

Apart from classifications based on the switch-induced graph, we can also classify switches based on their size. The *size structure*  $\Sigma(S) := \{|S[i]| \mid i \in [|S|]\}$  of switch  $S$  is a multiset containing the number of edges for each position of the switch. For a multiset  $\sigma$  of natural numbers, let  $S$  be a  $\sigma$ -switch if  $\Sigma(S) \subseteq \sigma$ . In a nutshell,  $\sigma$  restricts the number of edges in each position of the switch while also limiting the number of positions

in total. We say that  $\sigma$  *bounds* the size structure of the switch  $S$ . To illustrate this way of classifying switches, consider the example switch graph in Fig. 2.1. Here, we have  $\Sigma(S_1) = \{2, 1\}$ ,  $\Sigma(S_2) = \{1, 1\}$ ,  $\Sigma(S_3) = \{1, 1, 1\}$ , and  $\Sigma(S_{v_5v_6}) = \{1\}$ . Thus, we can say, for instance, that every switch apart from  $S_3$  in  $\widehat{G}$ , as illustrated in Fig. 2.1, is a  $\{2, 1\}$ -switch.

If we simply want to express that each switch position of  $S$  contains at most  $i$  edges, then we can say that  $S$  is a  $\{i\}^*$ -switch, where  $\{i\}^*$  is the multiset  $\{i, i, \dots\}$  with an arbitrary number of occurrences of  $i$ . For example, in switch graph  $\widehat{G}$  illustrated in Fig. 2.1, every switch apart from  $S_1$  is a  $\{1\}^*$ -switch. We frequently also include the wildcard symbol  $*$  in  $\sigma$  to indicate that a certain position may have an arbitrary number of edges. Thus,  $*$  can simply be seen as a very large number.

Note that we can combine the two switch classifications above. For example, a switch is a  $\{1\}^*$ -star switch if it is a star switch with size structure bounded by  $\{1\}^*$ . In switch graph  $\widehat{G}$  illustrated in Fig. 2.1, only switches  $S_2$  and  $S_{v_5v_6}$  are  $\{1\}^*$ -star switches.

Secondly, we aim to establish tools that pertain to the entire switch graph  $\widehat{G}$  rather than just the individual switches. Let  $E(\widehat{G})$  be the combined set of edges from all switches  $\mathcal{S}(\widehat{G})$ , that is,  $E(\widehat{G}) := \bigcup_{S \in \mathcal{S}(\widehat{G})} E(S)$ . These edges together with  $V(\widehat{G})$  form the *union graph*  $G(\widehat{G})$ , where  $V(G(\widehat{G})) := V(\widehat{G})$  and  $E(G(\widehat{G})) := E(\widehat{G})$ . This allows us to restrict the structure of the switch graph  $\widehat{G}$  using classical graph properties similar to the switch-induced graph.

A switch graph  $\widehat{H}$  is a *subswitch graph* of  $\widehat{G}$  if  $V(\widehat{H}) \subseteq V(\widehat{G})$  and  $\mathcal{S}(\widehat{H}) \subseteq \mathcal{S}(\widehat{G})$ . Moreover, each switch  $S \in \mathcal{S}(\widehat{H})$  retains its identity and number of positions, but each position may contain a restricted subset of the edges from  $\widehat{G}$ , at least restricted to only contain edges with both endpoints in  $V(\widehat{H})$ . Since the identity of switches is retained, we can apply any configuration  $\text{cof } \widehat{G}$  on  $\widehat{H}$ . Observe that  $\widehat{H} \circ c$  is a subgraph of  $\widehat{G} \circ c$ .

Given a vertex set  $U \subseteq V(\widehat{G})$ , we denote  $\widehat{G}[U]$  as the *vertex-induced subswitch graph*, which is the *maximal* subgraph of  $\widehat{G}$  such that  $V(\widehat{G}[U]) \subseteq U$ . Here, maximal means that every subswitch graph  $\widehat{H}$  of  $\widehat{G}$  with  $V(\widehat{H}) \subseteq U$  is a subswitch graph of  $\widehat{G}[U]$ . Intuitively, this construction removes all edges from switches that have at least one endpoint outside of  $U$ . If, as a result, a switch no longer contains any edges, it is removed from the graph entirely (as empty switches are not allowed).

The union of two switch graphs  $\widehat{G}$  and  $\widehat{H}$ , denoted by  $\widehat{G} \cup \widehat{H}$ , is the switch graph with vertex set  $V(\widehat{G}) \cup V(\widehat{H})$  and switch set  $\mathcal{S}(\widehat{G}) \cup \mathcal{S}(\widehat{H})$ . A switch with the same identity in both switch graphs must have the same number of positions and is *merged* into a single switch, with each position being the union of the corresponding positions from both switch graphs.

Lastly, given a switch graph  $\widehat{G}$ , we say that a vertex  $v \in V(\widehat{G})$  is *incident* to a switch  $S \in \mathcal{S}(\widehat{G})$  if  $v \in V(S)$ . The *switch neighborhood*  $\mathcal{S}_{\widehat{G}}(v)$  of vertex  $v$  is the set of incident switches, that is,  $\mathcal{S}_{\widehat{G}}(v) = \{S \in \mathcal{S}(\widehat{G}) \mid v \in V(S)\}$ . The *switch degree*  $\text{deg}_{\widehat{G}}(v)$  of vertex  $v$  is defined as the number of incident switches, i.e.,  $\text{deg}_{\widehat{G}}(v) = |\mathcal{S}_{\widehat{G}}(v)|$ .

### 2.3.2 Switching Problems

Let  $\Pi$  be a family of graphs closed under isomorphisms. The first problem definition we explore is the simplest variant of the switching problem, closely related to the concepts seen in the literature. In particular, many of these problem variants can be viewed as special case of the following problem, with certain restrictions on the input.

**$\Pi$ -SWITCHING**

**Input:** A switch graph  $\hat{G}$ .

**Question:** Does there exist a configuration  $c \in \mathcal{C}(\hat{G})$  such that  $\hat{G} \circ c \in \Pi$ ?

The next problem variant that is explored in this thesis involves an additional weight function  $w : \mathcal{S}(\hat{G}) \times \mathbb{N}_{>0} \rightarrow \mathbb{R}$  defined on the positions of the switches  $\mathcal{S}(\hat{G})$ . The total weight of a configuration  $c \in \mathcal{C}(\hat{G})$  is defined as  $w(c) = \sum_{S \in \mathcal{S}(\hat{G})} w(S, c(S))$ . This leads to the following problem definition:

**WEIGHTED- $\Pi$ -SWITCHING**

**Input:** A switch graph  $\hat{G}$ , a weight function  $w : \mathcal{S}(\hat{G}) \times \mathbb{N}_{>0} \rightarrow \mathbb{R}$ , and a constant  $\lambda \in \mathbb{R}$ .

**Question:** Does there exist a configuration  $c \in \mathcal{C}(\hat{G})$  such that  $\hat{G} \circ c \in \Pi$  and  $w(c) \leq \lambda$ ?

A restricted variant of **WEIGHTED- $\Pi$ -SWITCHING** which is often focused on when studying the complexity of the weighted variant of the switching problem concerns itself with  $\{*, *\}$ -switches, so-called *on-off switches*. Here, the first position of any switch represents the off-position and the second position of any switch (if one exists) represents the on-position. A *on-off-switch graph* is simply a switch graph where every switch is an on-off-switch. For this variant of switch graphs, we define the following problem:

**ONOFF- $\Pi$ -SWITCHING**

**Input:** An on-off-switch graph  $\hat{G}$  and a constant  $k \in \mathbb{N}_{\geq 0}$ .

**Question:** Does there exist a configuration  $c \in \mathcal{C}(\hat{G})$  such that  $\hat{G} \circ c \in \Pi$  and at most  $k$  switches of  $\mathcal{S}(\hat{G})$  are turned on, i.e.,  $|c|_2 \leq k$ ?

Observe that we can simulate the behavior of **ONOFF- $\Pi$ -SWITCHING** using **WEIGHTED- $\Pi$ -SWITCHING** by defining the weight function  $w$  such that  $w(S, 1) = 0$  and  $w(S, 2) = 1$  for each switch  $S \in \mathcal{S}(\hat{G})$ , and setting  $\lambda := k$ . Note that  $k$  is commonly referred to as the *activation budget* in the context of **ONOFF- $\Pi$ -SWITCHING**.

In the remainder of the paper, we explore the complexity of these problems for various graph families  $\Pi$  and various restrictions on the input of these three problems. Considered graph families include *globally connected* graphs (**GCON**), where between every pair of

vertices there is a path, and *s-t-connected* graphs (STCON), where each graph has a path between two distinguished terminals.

Since we often apply transformations to instances in order to prove complexity results for different structural properties of the switch graph, we say that two instances of the same problem are *equivalent* when one is a yes-instance if and only if the other is also a yes-instance.

## 2.4 Parameterized Complexity

Parameterized complexity is a framework for studying the computational complexity of problems with respect to one or more parameters in addition to the overall input size. This approach is particularly useful for analyzing problems that are intractable in general, but may be efficiently solvable when certain parameters are small. In this section, we present the fundamental concepts relevant to this thesis. For a more comprehensive overview, see Cygan et al. [4], Niedermeier [18], Flum and Grohe [6].

In a *parameterized problem*, aside from the input  $X$ , we are also given a parameter  $k \in \mathbb{N}_{\geq 0}$  which captures some underlying structural property of the instance. For example, in the vertex cover problem, a possible parameter might be the size of the resulting vertex cover itself.

A parameterized problem is said to be *fixed-parameter tractable* (FPT) if there exists an algorithm that runs in  $f(k) \cdot |X|^{O(1)}$  time, where  $f$  is a computable function depending only on  $k$ .

A *parameterized reduction* from a parameterized problem  $A$  to another parameterized problem  $B$  is a mapping that transforms an instance  $(X, k)$  of  $A$  into an instance  $(X', k')$  of  $B$  such that:

- the reduction is *correct*, that is,  $(X, k)$  is a yes-instance of  $A$  if and only if  $(X', k')$  is a yes-instance of  $B$
- the transformation runs in time  $f(k) \cdot |x|^{O(1)}$  for some computable function  $f$ ,
- $k' \leq g(k)$  for some computable function  $g$ .

Using parameterized reductions, we can demonstrate that certain problems are not fixed-parameter tractable. To formalize this notion, the *W-hierarchy* is introduced, which captures various levels of parameterized intractability. The classes  $W[1], W[2], \dots$  form a hierarchy of parameterized complexity classes, satisfying  $FPT \subseteq W[1] \subseteq W[2] \subseteq \dots$ . A problem is  $W[t]$ -hard if there exists a parameterized reduction from a known  $W[t]$ -complete problem to it. Problems that are  $W[t]$ -hard for some  $t \geq 1$  are widely believed not to be fixed-parameter tractable.

Additionally, the class  $W[\text{SAT}]$  consists of all problems that admit a parameterized reduction to  $\text{WSAT}(\Phi)$  parameterized by the maximum weight of the truth assignment. Within the  $W$ -hierarchy, it is believed that  $W[t] \subseteq W[\text{SAT}]$  for all  $t \geq 1$ .

## 2.5 Matroid Theory

An *independence system* is a pair  $(E, \mathcal{I})$ , where  $E$  is a finite set called the ground set, and  $\mathcal{I} \subseteq 2^E$  is a collection of subsets of  $E$  called independent sets. The defining properties are:

- **Non-emptiness:**  $\emptyset \in \mathcal{I}$ .
- **Heredity:** If  $A \in \mathcal{I}$  and  $B \subseteq A$ , then  $B \in \mathcal{I}$ .

In this context, a set  $A \subseteq E$  is called *independent* if  $A \in \mathcal{I}$ ; otherwise,  $A$  is called *dependent*.

A *matroid* is an independence system  $(E, \mathcal{I})$  that also satisfies the exchange property:

- **Exchange:** If  $A, B \in \mathcal{I}$  and  $|A| < |B|$ , then there exists  $e \in B \setminus A$  such that  $A \cup \{e\} \in \mathcal{I}$ .

The *rank* of a matroid is defined as the size of the largest independent set. A *base* is a maximal independent set, and all bases of a matroid have the same size. A *circuit* is a minimal dependent set, meaning it is a dependent set such that every proper subset is independent.

Given two matroids  $M_1 = (E_1, \mathcal{I}_1)$  and  $M_2 = (E_2, \mathcal{I}_2)$  with disjoint ground sets, their *direct sum* is a matroid  $(E_1 \cup E_2, \mathcal{I})$ , where  $\mathcal{I} = \{A \cup B \mid A \in \mathcal{I}_1, B \in \mathcal{I}_2\}$ .

We now present several important examples of matroids:

- *Uniform matroid:* The independent sets are all subsets of  $E$  of size at most  $r$ .
- *Partition matroid:* The partition matroid is defined as the direct sum of uniform matroids with disjoint ground sets.
- *Graphic matroid:* For a graph  $G = (V, E)$ , the independent sets are edge sets that contain no cycles (i.e., forests).

For a more comprehensive overview on matroid theory, see Korte et al. [13].

# Global Connectivity Switching and Matroids

Many combinatorial problems can be naturally modeled using matroids — set systems of independent sets that satisfy the heredity and exchange properties (see Chapter 2 for a formal definition). What constitutes an “independent” set depends on the underlying structure of the matroid. For instance, for a set of edges in a graph, independence may mean that the set contains no cycles. The main advantage of modeling problems with matroids is that it enables the application of powerful algorithmic tools, often leading to efficient solutions.

Katz et al. [11, Section 4] utilize matroid theory to solve problems in their switching model, which closely resembles ours when restricted to  $\{1\}^*$ -star switches. They show that, under this restriction, a configuration ensuring global connectivity of the configuration graph, if one exists, can be found in polynomial time. This is achieved via a reduction to the *matroid intersection problem*, which itself can be solved in polynomial time [5]. This problem involves two matroids defined over the same ground set, and the goal is to find the largest set that is independent in both. In their reduction, they define two matroids over the edges of all switches: one matroid enforces connectivity, while the other ensures that the selected edges correspond to a valid configuration of the switch graph. A solution of the matroid intersection problem thus yields a connected configuration of the switch graph.

We begin by revisiting this result in more detail and tailoring it to the setting of GCON-SWITCHING. In doing so, we explore how the approach can be extended to a broader class of switches. Specifically, we identify a matroid-like structure for switches that still permits a polynomial-time algorithm for GCON-SWITCHING (Theorem 1). Moreover, we demonstrate that the result naturally generalizes to the weighted variant of the problem when restricted to  $\{1\}^*$ -switches (Theorem 2).

### 3.1 The Matroid Intersection Problem

As previously mentioned, Katz et al. [11] proposed a reduction to the matroid intersection problem. We adopt a similar approach but work with a slightly modified formulation of that problem, where the goal is to decide whether an independent set of a prescribed cardinality exists rather than finding one of maximum size:

#### MATROID INTERSECTION

**Input:** Two matroids  $(E, \mathcal{I}_1)$  and  $(E, \mathcal{I}_2)$ , and an integer  $k$ .

**Question:** Does there exist an independent set  $A \subseteq E$  such that  $|A| = k$  and  $A \in \mathcal{I}_1 \cap \mathcal{I}_2$ ?

To later extend the result to the weighted variant of the switching problem, we also require a corresponding weighted formulation of the matroid intersection problem:<sup>1</sup>

#### WEIGHTED MATROID INTERSECTION

**Input:** Two matroids  $(E, \mathcal{I}_1)$  and  $(E, \mathcal{I}_2)$ , a weight function  $w : E \rightarrow \mathbb{R}$ , an integer  $k$ , and a real number  $\lambda$ .

**Question:** Does there exist an independent set  $A \subseteq E$  such that  $|A| = k$ ,  $A \in \mathcal{I}_1 \cap \mathcal{I}_2$ , and  $\sum_{e \in A} w(e) \leq \lambda$ ?

Both of these problems are solvable in polynomial time — in particular, in time  $\mathcal{O}(mk(k + T_1 + T_2 + \log m))$ , where  $m = |E|$  and for  $i \in [2]$ ,  $T_i$  denotes the complexity of the *circuit finding oracle* of matroid  $(E, \mathcal{I}_i)$  [3]. Given an independent set  $A \subseteq E$  and an additional element  $e \in E \setminus A$ , this oracle either confirms that  $A \cup \{e\}$  remains independent in matroid  $(E, \mathcal{I}_i)$ , or finds a circuit contained in  $A \cup \{e\}$ , thereby witnessing dependence. Importantly, this oracle-based formulation allows us to avoid storing the independent sets explicitly. Instead, the underlying algorithm for solving the matroid intersection problem calls the oracle to verify independence.

In both the unweighted and weighted settings, a witnessing independent set (if one exists) is produced as part of the matroid intersection algorithm [3]. In the case of the weighted variant, the algorithm even outputs one of minimum weight. This set can then be used to construct a corresponding (minimum-weight) connected configuration of  $\hat{G}$ . However, since our focus lies on the complexity of the decision problem, we do not elaborate on this construction in this chapter.

<sup>1</sup>In the literature, this problem is typically framed as finding a maximum-weight independent set. However, by negating the weights, it equivalently reduces to a minimum-weight version, allowing us to compare the total weight against a given threshold  $\lambda$ .

### 3.2 Defining the Matroids

Given a switch graph  $\widehat{G}$ , the two matroids are defined over the multiset  $\widehat{E}$ , which is formed by the disjoint union of the edges over all switches:

$$\widehat{E} := \bigsqcup_{S \in \mathcal{S}(\widehat{G})} E(S).$$

Note that elements in  $\widehat{E}$  are tagged with their originating switch. As a result, two edges with identical endpoints are treated as distinct elements in  $\widehat{E}$  if they come from different switches.

The first matroid,  $(\widehat{E}, \mathcal{I}_1)$ , responsible for guaranteeing connectedness, is simply the graphic matroid over  $\widehat{E}$ . Its independent sets correspond to the cycle-free subsets of  $\widehat{E}$ . Thus, the independent sets of size  $|V(\widehat{G})| - 1$  correspond precisely to spanning trees in the union graph  $G(\widehat{G})$ . If no such independent set exists, then  $G(\widehat{G})$  is disconnected, and thus no configuration of  $\widehat{G}$  can yield global connectivity. However, the existence of a spanning tree alone does not guarantee a connected configuration, since the edges may originate from conflicting switch positions.

To address this, we define the second matroid  $(\widehat{E}, \mathcal{I}_2)$ , where every independent set includes edges from at most one position per switch, or equivalently, consists of a subset of edges taken from a valid configuration of  $\widehat{G}$ . When restricted to  $\{1\}^*$ -star switches, this means selecting at most one edge per switch. Note, however, that  $(\widehat{E}, \mathcal{I}_2)$  forms a matroid only under certain structural assumptions about the switches, which we discuss later.

Independent sets in  $\mathcal{I}_1 \cap \mathcal{I}_2$  of size  $|V(\widehat{G})| - 1$  then correspond precisely to spanning trees that can be realized by a valid configuration of  $\widehat{G}$ . Thus, a globally connected configuration exists if and only if such an independent set exists.

To determine when  $(\widehat{E}, \mathcal{I}_2)$  forms a matroid, observe that it corresponds to the direct sum of the independence systems  $(E(S), \mathcal{I}_S)$  defined for each switch  $S \in \mathcal{S}(\widehat{G})$ , where  $\mathcal{I}_S$  consists of all subsets of edges contained within a single position of  $S$ . If each individual system  $(E(S), \mathcal{I}_S)$  is a matroid, then their direct sum  $(\widehat{E}, \mathcal{I}_2)$  is a matroid as well. We refer to such switches as *matroid switches*, and  $(E(S), \mathcal{I}_S)$  as the corresponding *switch matroid* for  $S$ .

In the setting of Katz et al. [11], the switches are restricted to  $\{1\}^*$ -star switches, which form a special case of matroid switches. Specifically, for each switch  $S \in \mathcal{S}(\widehat{G})$ , the independent sets in  $(E(S), \mathcal{I}_S)$  consist of all subsets of  $E(S)$  of size at most one, forming a uniform matroid of rank one. As a result,  $(\widehat{E}, \mathcal{I}_2)$  becomes a partition matroid — the direct sum of uniform matroids, one for each switch. Notably, the star property of the switches plays no role in this argument and is therefore irrelevant for the correctness of the reduction.

The following theorem summarizes the result for global connectivity switching and determines the concrete complexity for the problem:

**Theorem 1.** GCON-SWITCHING restricted to matroid switches can be decided in time  $\mathcal{O}(mn \cdot (n + T + \log m))$ , where:

- $m = \sum_{S \in \mathcal{S}} |E(S)|$  is the total number of edges across all switches, i.e., the size of the multiset  $\widehat{E}$ .
- $n = |V(\widehat{G})|$  is the number of vertices in the switch graph, and
- $T = \max_{S \in \mathcal{S}(\widehat{G})} T_S$  is the maximum time complexity of the circuit-finding oracle among all switch matroids  $(E(S), \mathcal{I}_S)$ .

*Proof.* Given an instance  $\widehat{G}$  of GCON-SWITCHING, we define the two matroids  $(\widehat{E}, \mathcal{I}_1)$  and  $(\widehat{E}, \mathcal{I}_2)$  as previously described. We have established that  $\widehat{G}$  admits a globally connected configuration if and only if there exists an independent set  $A \subseteq \widehat{E}$  with  $|A| = n - 1$  and  $A \in \mathcal{I}_1 \cap \mathcal{I}_2$ .

Recall that MATROID INTERSECTION can be solved in time  $\mathcal{O}(m'k(k+T_1+T_2+\log m'))$ , where  $m'$  is the size of the ground set,  $k$  is the target cardinality of the independent set, and  $T_1, T_2$  are the complexities of the circuit finding oracles for the two matroids. In our case, we have  $k = n - 1$  and  $m = m'$ .

The circuit finding oracle, given an independent set  $A \subseteq \widehat{E}$  and an additional edge  $e \in \widehat{E} \setminus A$ , must determine whether  $A \cup \{e\}$  remains independent or else return a circuit contained in  $A \cup \{e\}$ . For the graphic matroid, this requires finding a cycle. Since  $A$  forms a forest, this amounts to finding a path between the endpoints of  $e$  in  $A$ , which can be done in  $\mathcal{O}(n)$  time using depth first search. Thus,  $T_1 = \mathcal{O}(n)$ .

Suppose edge  $e$  is contained in switch  $S$ . Then, for the second matroid  $(\widehat{E}, \mathcal{I}_2)$ , the circuit-finding oracle needs to find a circuit in  $(A \cap E(S)) \cup \{e\}$  within the switch matroid  $(E(S), \mathcal{I}_S)$ . This takes time  $T_S$ , so in the worst case, we have  $T_2 = T$ .

Since the matroids are defined implicitly, the reduction does not incur additional overhead. The overall complexity of solving the matroid intersection instance becomes:

$$\mathcal{O}(m(n-1)((n-1) + n + T + \log m)) = \mathcal{O}(mn(n + T + \log m)). \quad \square$$

One question remains: what is the complexity of the circuit-finding oracle for a switch matroid  $(E(S), \mathcal{I}_S)$ ? In the case where  $(E(S), \mathcal{I}_S)$  is a uniform matroid, which is the case for  $\{1\}^*$ -switches, the oracle can be implemented in constant time. This is because, in a uniform matroid, a set is independent if and only if its size does not exceed the rank. Therefore, if adding the new edge would exceed the rank, we have immediately identified a circuit; otherwise, the set remains independent. Denoting such switches as *uniform matroid switches*, we obtain the following overall running time for this variant of the switching problem:

**Corollary 1.** *GCON-SWITCHING restricted to uniform matroid switches (which includes  $\{1\}^*$ -switches) can be decided in time  $\mathcal{O}(mn \cdot (n + \log m))$ , where  $m = \sum_{S \in \mathcal{S}} |E(S)|$  and  $n = |V(\widehat{G})|$ .*

However, for more general switches, this guarantee no longer holds. Nevertheless, as long as we can determine in polynomial time whether a given set of edges is independent in  $(E(S), \mathcal{I}_S)$ , for instance via a polynomial-time independence oracle or because the number of switch positions is polynomially bounded, we can still decide in polynomial time whether  $\widehat{G}$  admits a globally connected configuration.

### 3.3 Adaption to the Weighted Variant

As hinted at previously, the above result can be extended to the weighted variant of the switching problem. Here, we are given a switch graph  $\widehat{G}$ , a weight function  $w : \mathcal{S}(\widehat{G}) \times \mathbb{N}_{>0} \rightarrow \mathbb{R}$  that assigns a weight to each switch position, and a real number  $\lambda$ . The goal is to determine whether there exists a globally connected configuration of  $\widehat{G}$  with total weight at most  $\lambda$ .

In the weighted variant of the WEIGHTED MATROID INTERSECTION, recall that we are additionally given a weight function  $f : E \rightarrow \mathbb{R}$  over the ground set  $E$  and a real number  $\gamma$ . The goal is to find an independent set contained in both matroids of a given cardinality  $k$  such that the cumulative weight of the independent set does not exceed  $\gamma$ .

When restricted to  $\{1\}^*$ -switches, each switch position contains at most one edge. In this case, assigning weights directly to edges in  $\widehat{E}$  — the disjoint union of edges over all switches — is essentially equivalent to assigning weights to the switch positions themselves. This allows us to adapt the previous reduction to the weighted setting by defining a suitable weight function  $f$  over the ground set  $\widehat{E}$ , derived from the weight function  $w$  on switch positions. However, this correspondence breaks down for more general switches, where a single switch position may contain multiple edges. For this reason, we restrict our attention to  $\{1\}^*$ -switches for the time being.

For each switch  $S \in \mathcal{S}(\widehat{G})$  and edge  $e \in E(S)$ , the weight function  $f : \widehat{E} \rightarrow \mathbb{R}$  is defined as follows:

$$f(e) := w(S, i) \text{ where } i := \arg \min_{j \in [|S|]} \{w(S, j) \mid e \in S[j]\}.$$

That is, the weight of an edge  $e \in \widehat{E}$  is mapped to the weight of the corresponding switch position that contains that edge. If  $e$  appears in multiple positions within a switch, the position with the smallest weight is selected.

As in the unweighted variant, an independent set  $A \in \mathcal{I}_1 \cap \mathcal{I}_2$  of cardinality  $|V(\widehat{G})| - 1$  represents a spanning tree of a configuration of  $\widehat{G}$ . The total weight of  $A$  corresponds to the accumulative weight of the switches that provide the edges in  $A$ . However, this does

not yet capture the total weight of a complete configuration, as some switches may not contribute any edges to  $A$  and are thus unaccounted for.

To resolve this, we restrict the instance such that, for each switch  $S \in \mathcal{S}(\widehat{G})$ , the minimum weight among all its positions is zero. This guarantees that each switch not contributing to the independent set  $A$  can be configured using its zero-weight position, thereby incurring no additional cost. As a result, the total weight of  $A$  matches the minimum weight of a configuration of  $\widehat{G}$  that includes the spanning tree represented by  $A$ . Consequently, independent sets in  $\mathcal{I}_1 \cap \mathcal{I}_2$  of size  $|V(\widehat{G})| - 1$  and total weight at most  $\lambda$  correspond precisely to the connected configurations of  $\widehat{G}$  whose total weight does not exceed  $\lambda$ .

Since we aim to work with unrestricted weight functions, we introduce a general transformation that modifies both the weight function  $w$  and the maximum allowed weight  $\lambda$  to adhere to the constraints above:

**Construction 1.** Let  $(\widehat{G}, w, \lambda)$  be an instance of WEIGHTED-GCON-SWITCHING. We construct an equivalent instance  $(\widehat{G}, w', \lambda')$  of WEIGHTED-GCON-SWITCHING by altering the weight function and maximum allowed weight of a configuration as follows:

$$w'(S, i) := w(S, i) - \min_{j \in [|S|]} \{w(S, j)\} \quad \text{for all } S \in \mathcal{S}(\widehat{G}), i \in [|S|], \text{ and}$$

$$\lambda' := \lambda - \sum_{S \in \mathcal{S}(\widehat{G})} \min_{i \in [|S|]} \{w(S, i)\}.$$

Consequently, for each switch  $S \in \mathcal{S}(\widehat{G})$ , we have  $\min_{i \in [|S|]} \{w'(S, i)\} = 0$ .

To see the correctness of this transformation, let  $c$  be an arbitrary configuration of  $\widehat{G}$ . Then the total weight of  $c$  according to  $w'$  can be expressed as follows:

$$\begin{aligned} w'(c) &= \sum_{S \in \mathcal{S}(\widehat{G})} w'(S, c(S)) \\ &= \sum_{S \in \mathcal{S}(\widehat{G})} \left( w(S, c(S)) - \min_{i \in [|S|]} \{w(S, i)\} \right) \\ &= \sum_{S \in \mathcal{S}(\widehat{G})} w(S, c(S)) - \sum_{S \in \mathcal{S}(\widehat{G})} \min_{i \in [|S|]} \{w(S, i)\} \\ &= w(c) - \sum_{S \in \mathcal{S}(\widehat{G})} \min_{i \in [|S|]} \{w(S, i)\} \end{aligned}$$

Consequently, the following equivalence holds:

$$\begin{aligned} w(c) \leq \lambda &\Leftrightarrow w'(c) + \sum_{S \in \mathcal{S}(\widehat{G})} \min_{i \in [|S|]} \{w(S, i)\} \leq \lambda \\ &\Leftrightarrow w'(c) \leq \lambda - \min_{i \in [|S|]} \{w(S, i)\} \Leftrightarrow w'(c) \leq \lambda \end{aligned}$$

Thus, the transformation ensures that each configuration of  $\widehat{G}$  which has a total weight at most  $\lambda$  according to  $w$  also has a total weight at most  $\lambda'$  according to  $w'$ , and vice versa, yielding the following lemma:

**Lemma 1.** *Let  $(\widehat{G}, w, \lambda)$  be an instance of WEIGHTED-GCON-SWITCHING. Then  $(\widehat{G}, w', \lambda')$  obtained from Construction 1 is an equivalent instance of WEIGHTED-GCON-SWITCHING.*

After employing this transformation on any given instance of WEIGHTED-GCON-SWITCHING restricted to  $\{1\}^*$ -switches, we can apply the reduction to WEIGHTED MATROID INTERSECTION as described above, yielding the following result:

**Theorem 2.** *WEIGHTED-GCON-SWITCHING restricted to  $\{1\}^*$ -switches can be decided in time  $\mathcal{O}(mn \cdot (n + \log m))$ , where  $m = \sum_{S \in \mathcal{S}} |E(S)|$  and  $n = |V(\widehat{G})|$ .*

*Proof.* We have already shown that the reduction from WEIGHTED-GCON-SWITCHING to WEIGHTED MATROID INTERSECTION is correct. Since the running time of the weighted matroid intersection problem is identical to that of the unweighted variant, we can apply the same analysis as in Theorem 1 and Corollary 1, obtaining the desired result.  $\square$

**Discussion** Extending this result for WEIGHTED-GCON-SWITCHING to more general matroid switches is non-trivial. The core challenge lies in the fact that the weight function  $w$  is defined on switch positions rather than on individual edges. While this poses no issue for  $\{1\}^*$ -switches where each position contains at most one edge, it complicates matters for more general switches since we cannot naively spread the weight of a switch position across its edges. Consequently, the complexity of WEIGHTED-GCON-SWITCHING restricted to matroid switches remains an open problem.

For switches that do not necessarily satisfy the matroid property, we will later show in Chapter 7 that both the unweighted and weighted variants of GCON-SWITCHING become intractable.



# Connection to Transition-Compatible Paths

Szeider [20] introduced a switching model that, at first glance, appears only loosely related to the switching model studied in this thesis. Their approach focuses solely on the existence of an  $s$ - $t$ -path and employs a set of structures known as *transition graphs*, defined on top of a simple graph. Unlike switching models that define switches to enable or disable certain edges, these transition graphs specify, for each vertex, which pairs of incident edges are allowed to appear consecutively in a path. The associated decision problem, TRANSITION-COMPATIBLE- $s$ - $t$ -PATH, then asks whether there exists an  $s$ - $t$ -path that respects these local transition constraints.

In this chapter, we explore the connection between TRANSITION-COMPATIBLE- $s$ - $t$ -PATH and STCON-SWITCHING, presenting reductions in both directions. As a result, we prove that STCON-SWITCHING is NP-complete even when restricted to  $\{1, 1\}$ -switches and to  $\{2, 1\}$ -star switches (Theorems 3 and 4, respectively). Additionally, we show that STCON-SWITCHING restricted to  $\{1\}^*$ -switches is solvable in linear time if the set of admissible switch-induced graphs are all complete multipartite; otherwise, the problem remains NP-complete (see Theorem 6).

Before delving into the reductions, we first introduce the necessary preliminaries and formally define TRANSITION-COMPATIBLE- $s$ - $t$ -PATH. Given a graph  $G$ , let  $T(v)$  be a transition graph for a vertex  $v \in V(G)$ , whose vertices are the edges of  $G$  incident to  $v$ , that is,  $V(T(v)) := E_G(v)$ . Each edge  $ef \in E(T(v))$  represents an *allowed transition* between  $e$  and  $f$  via vertex  $v$ . A *transition system*  $T$  is the collection of all transition graphs  $T(v)$  for every vertex  $v \in V(G)$ . A path  $P = v_0, v_1, \dots, v_k$  in  $G$  is called  *$T$ -compatible* if it consists only of allowed transitions; that is,  $\{v_{i-1}v_i, v_iv_{i+1}\} \in E(T(v_i))$  for every  $i \in [k - 1]$ . This leads to the following problem definition:

TRANSITION-COMPATIBLE- $s$ - $t$ -PATH

**Input:** A graph  $G$  with a designated source  $s(G)$  and target  $t(G)$  vertex, and a transition system  $T$ .

**Question:** Is there a  $T$ -compatible path between  $s(G)$  and  $t(G)$  in  $G$ ?

### 4.1 From Transition Graphs to Switches

Given a graph  $G$  with transition system  $T$ , observe that for every transition-compatible path  $P$  and vertex  $v \in V(G)$ , at most one allowed transition of  $T(v)$  is used in  $P$ . This is because  $P$  is a path, so  $v$  is visited at most once and may only participate in at most one transition. This observation enables us to model each transition graph in  $T$  using a  $\{1\}^*$ -switch, where each position corresponds to a single allowed transition. Formally, the construction proceeds as follows and is illustrated in Fig. 4.1 by a simple example:

**Construction 2.** Let  $G$  be a graph with transition system  $T$ . We construct a corresponding switch graph  $\hat{G}_T$  as follows:

- The vertex set is defined as  $V(\hat{G}_T) := \{s(G), t(G)\} \cup E(G)$ , i.e., the original terminals and one *edge vertex* for each edge in  $G$ .
- For each non-terminal vertex  $v \in V(G) \setminus \{s(G), t(G)\}$ , where the allowed transitions in  $T(v)$  are given by  $E(T(v)) = \{e_1, \dots, e_k\}$ , we create a  $\{1\}^*$ -switch  $S_v = (\{e_i\}_{i \in [k]})$ . That is, each allowed transition  $e_i \in E(T(v))$  becomes a position in the switch.
- For each edge  $e \in E_G(v)$  incident to a terminal  $v \in \{s(G), t(G)\}$ , we introduce a static edge  $S_e = (\{ve\})$ , connecting the terminal to the corresponding edge vertex.
- We define the source and target of the switch graph as  $s(\hat{G}_T) := s(G)$  and  $t(\hat{G}_T) := t(G)$ , respectively.

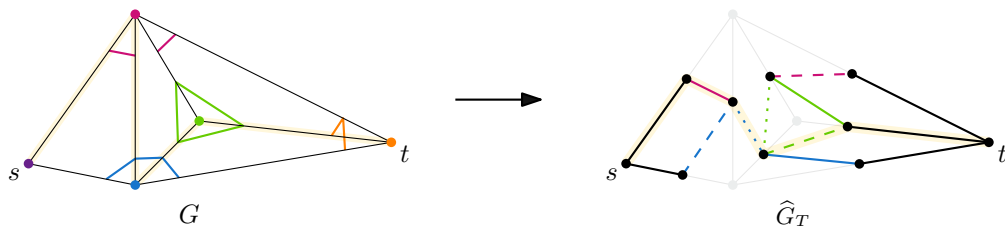


Figure 4.1: Example of a graph  $G$  with transition system  $T$  (left) and the corresponding switch graph  $\hat{G}_T$  (right), constructed according to Construction 2. Each transition graph with its vertex and its corresponding switch are shown in the same color. Static edges in  $\hat{G}_T$  are shown in black. The yellow-highlighted edges indicate a possible  $T$ -compatible  $s$ - $t$ -path in  $G$  and the corresponding  $s$ - $t$ -path in  $\hat{G}_T$  under a valid configuration.

Observe that for each vertex  $v \in V(G)$ , the transition graph  $T(v)$  is isomorphic to the switch-induced graph of the corresponding switch  $S_v$ .

Intuitively, a  $T$ -compatible  $s$ - $t$ -path in  $G$  can be viewed as a sequence of allowed transitions, which directly corresponds to a valid path in the switch graph  $\widehat{G}_T$  under a suitable configuration. Conversely, any  $s$ - $t$ -path in  $\widehat{G}_T$  under some configuration can be interpreted as a sequence of allowed transitions in  $G$ , thereby yielding a  $T$ -compatible  $s$ - $t$ -path. The following lemma formalizes this equivalence:

**Lemma 2.** *Let  $G$  be a graph with transition system  $T$ . Then, there exists a  $T$ -compatible  $s$ - $t$ -path in  $G$  if and only if the switch graph  $\widehat{G}_T$  constructed in Construction 2 admits an  $s$ - $t$ -path under some configuration.*

*Proof.* ( $\Rightarrow$ ) Let  $P := \langle v_0, v_1, \dots, v_{k-1}, v_k \rangle$  be a  $T$ -compatible  $s$ - $t$ -path in  $G$ . For each inner vertex  $v_i$  with  $i \in [k-1]$ , the pair  $\{v_{i-1}v_i, v_iv_{i+1}\}$  forms an allowed transition in  $T(v_i)$ . We define a configuration  $c$  of  $\widehat{G}_T$  by setting each corresponding switch  $S_{v_i}$  to the position that includes edge  $\{v_{i-1}v_i, v_iv_{i+1}\}$ . All other switches are set to position 0. Then, the sequence  $P' := \langle v_0, v_0v_1, v_1v_2, \dots, v_{k-1}v_k, v_k \rangle$  forms an  $s$ - $t$ -path in  $\widehat{G}_T$  under configuration  $c$ , since  $P$  is a path and therefore no edge is repeated, which implies that no edge vertex is revisited in  $P'$ .

( $\Leftarrow$ ) Let  $c$  be a configuration of  $\widehat{G}_T$  that admits an  $s$ - $t$ -path  $P' := \langle s, e_1, e_2, \dots, e_k, t \rangle$ . For each pair of consecutive edge vertices  $\{e_i, e_{i+1}\}$  in  $P'$ , we know by construction that this pair shares a common endpoint  $v \in V(G)$ , and that it forms a valid transition in  $T(v)$  since  $\{e_i, e_{i+1}\} \in S_v[c(S_v)]$ . This implies that a path may transition from edge  $e_i$  to  $e_{i+1}$  via  $v$  in a transition compatible path in  $G$ . As  $\{e_i, e_{i+1}\}$  is the only edge contributed by switch  $S_v$  in configuration  $c$ , it is also the only transition via  $v$  of any pair of consecutive edge vertices in  $P'$ . Therefore, the sequence  $\langle e_1, e_2, \dots, e_k \rangle$  of edges forms a  $T$ -compatible path, as this prevents a single vertex to be visited multiple times. Finally, since  $e_1$  is incident to  $s$  and  $e_k$  to  $t$  by the definition of the static edges  $\{s, e_1\}$  and  $\{e_k, t\}$ , we conclude that  $G$  admits an  $T$ -compatible  $s$ - $t$ -path.  $\square$

Let  $\mathcal{A}$  be a family of graphs closed under isomorphism, and let  $\mathcal{A}^{\text{ind}}$  denote the closure of  $\mathcal{A}$  under taking vertex-induced subgraphs. Szeider [20, Theorem 1] established that TRANSITION-COMPATIBLE- $s$ - $t$ -PATH is NP-complete if the transition graphs are drawn from  $\mathcal{A}$  and  $\mathcal{A}^{\text{ind}}$  contains one of the sets:

$$\{K_2 + K_2, K_3\}, \quad \{K_2 + K_2, P_3\}, \quad \{P_4\}, \quad \text{or} \quad \{L_4\},$$

where  $K_2 + K_2$  denotes the disjoint union of two  $K_2$  graphs,  $P_i$  denotes a path on  $i$  vertices, and  $L_4$  is a triangle ( $K_3$ ) with an additional edge attached to one of its vertices.

A key consequence of this theorem is that the problem remains NP-complete even when the transition graphs are restricted to graphs isomorphic to  $P_3$  and  $K_2 + K_2$ . Notably, both of these graphs consist of exactly two edges, which translate to  $\{1, 1\}$ -switches via the reduction in Construction 2. Hence, by applying Lemma 2, we obtain the following hardness result for STCON-SWITCHING:

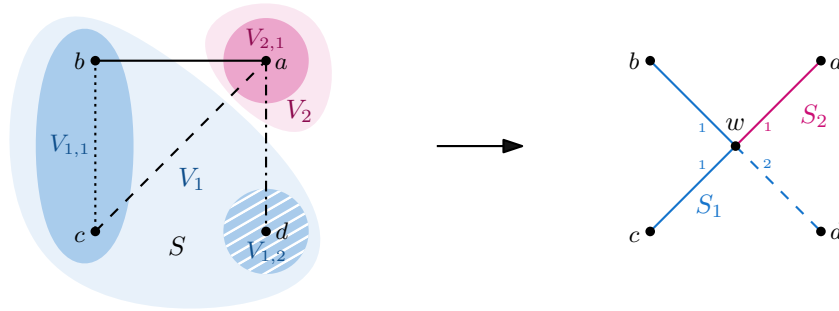


Figure 4.2: Example of Construction 3 applied to a  $\{1\}^*$ -switch  $S$ . The left side shows the original switch  $S$ , along with a partition of  $V(S)$  into two parts and an edge clique cover for each part. On the right, the corresponding two star switches centered at a fresh vertex  $w$  are depicted. Small numbers indicate the switch position of each edge.

**Theorem 3.** *STCON-SWITCHING is NP-hard, even when restricted to  $\{1, 1\}$ -switches.*

Although  $\{1, 1\}$ -switches are already highly constrained, they do not necessarily result in a connected switch-induced graph. To address this, we present a transformation that converts a single  $\{1\}^*$ -switch into set of star switches, all centered around a fresh vertex. We use this transformation to prove Theorem 4, which establishes that STCON-SWITCHING is NP-hard, even when restricted to  $\{2, 1\}$ -star switches. The construction required for this transformation is detailed in the following, and is illustrated in Fig. 4.2 by a simple example:

**Construction 3.** Let  $\widehat{G}$  be a switch graph and  $S \in \mathcal{S}(\widehat{G})$  a  $\{1\}^*$ -switch. We construct an equivalent switch graph  $\widehat{G}'$  by introducing a fresh vertex  $w$  and replacing  $S$  by a set of new switches whose edges are all incident to  $w$ .

Let  $\{V_1, \dots, V_k\}$  be a partition of  $V(S)$  such that for every pair of indices  $i \neq j$ , all possible edges between vertices in  $V_i$  and vertices in  $V_j$  are contained in  $E(S)$ . For each set  $i \in [k]$ , we introduce a new switch  $S_i$  to locally model the structure of  $G(S)[V_i]$ .

Let  $\{V_{i,1}, \dots, V_{i,\ell}\}$  be an *edge clique cover* of  $G(S)[V_i]$ , that is, a collection of subsets of  $V_i$  such that, for each  $j \in [\ell]$ , the induced subgraph  $G(S)[V_{i,j}]$  is a clique, and every vertex and edge in  $G(S)[V_i]$  is contained in at least one of the  $V_{i,j}$ . For each subset  $V_{i,j}$ , we define a switch position  $S_i[j] = \{wv \mid v \in V_{i,j}\}$ .

Intuitively, the construction ensures that for any pair of vertices  $u, v \in V(S)$ , the original switch  $S$  can be configured to connect  $u$  and  $v$  if and only if the new switches can be configured to connect them via the fresh vertex  $w$ . Since  $w$  can be traversed at most once in any path, only one such connection can be used, thereby replicating the behavior of the original  $\{1\}^*$ -switch  $S$ . The correctness of Construction 3 is established by the following lemma:

**Lemma 3.** *Let  $\widehat{G}$  be a switch graph and let  $S \in \mathcal{S}(\widehat{G})$  be a  $\{1\}^*$ -switch. Then,  $\widehat{G}$  admits an  $s$ - $t$ -path under some configuration if and only if the transformed switch graph  $\widehat{G}'$ , obtained via Construction 3, also admits an  $s$ - $t$ -path under some configuration.*

*Proof.* Let  $\{V_1, \dots, V_k\}$  be a partition of  $V(S)$  used in the construction of  $\widehat{G}'$ . We prove both directions of the equivalence separately.

( $\Rightarrow$ ) Let  $c$  be a configuration of  $\widehat{G}$  that admits an  $s$ - $t$ -path  $P$ . We extend it to a configuration  $c'$  of  $\widehat{G}'$  as follows:

- If  $P$  does *not* use any edge from  $S$ , then  $c'$  can assign arbitrary positions to the new switches in  $\widehat{G}'$ , and  $P$  remains a valid  $s$ - $t$ -path in  $\widehat{G}' \circ c'$ .
- Otherwise, let  $uv \in S[c(S)]$  be the edge used by  $P$ .
  - If  $u \in V_i$  and  $v \in V_j$  for some  $i \neq j$ , then we configure  $S_i$  such that  $uw \in S_i[c'(S_i)]$  and  $S_j$  such that  $wv \in S_j[c'(S_j)]$ . The remaining new switches can be configured arbitrarily. Replacing  $uv$  in  $P$  with the subpath  $u \rightarrow w \rightarrow v$  yields a valid  $s$ - $t$ -path in  $\widehat{G}' \circ c'$ .
  - If instead  $u, v \in V_i$  for some  $i \in [k]$ , then by the edge clique cover  $V_{i,1}, \dots, V_{i,\ell}$  used in the construction, there exists a  $V_{i,j}$  such that  $u, v \subseteq V_{i,j}$ . We set  $c'(S_i) = j$  so that  $uw, wv \in S_i[j]$ . Again, replacing  $uv$  in  $P$  by  $u \rightarrow w \rightarrow v$  yields an  $s$ - $t$ -path in  $\widehat{G}' \circ c'$ .

( $\Leftarrow$ ) Let  $c'$  be a configuration of  $\widehat{G}'$  that admits an  $s$ - $t$ -path  $P$ . We extend  $c'$  to a configuration  $c$  of  $\widehat{G}$  as follows:

- If  $P$  does not traverse the fresh vertex  $w$ , then it avoids all edges added in place of  $S$ , and  $S$  can be configured arbitrarily. Hence,  $P$  is also an  $s$ - $t$ -path in  $\widehat{G} \circ c$ .
- Otherwise, let  $u$  and  $v$  be the neighbors of  $w$  in  $P$ .
  - If  $uw \in S_i[c'(S_i)]$  and  $wv \in S_j[c'(S_j)]$  for some  $i \neq j$ , then  $u \in V_i$  and  $v \in V_j$ . By construction,  $w \in E(S)$ , so we configure  $S$  such that  $w \in S[c(S)]$ . Replacing the subpath  $u \rightarrow w \rightarrow v$  in  $P$  by the edge  $uv$  yields an  $s$ - $t$ -path in  $\widehat{G} \circ c$ .
  - If instead both  $uw$  and  $wv$  belong to  $S_i[c'(S_i)]$  for some  $i \in [k]$ , then both  $u$  and  $v$  lie in  $V_{i,j}$  with  $j := c'(S_i)$ . Since  $G(S)[V_{i,j}]$  is a clique, we have  $w \in E(S)$ , so we set  $c(S)$  such that  $w \in S[c(S)]$ . Replacing  $u \rightarrow w \rightarrow v$  with  $uv$  in  $P$  again yields an  $s$ - $t$ -path in  $\widehat{G} \circ c$ .  $\square$

Now, the result of Szeider [20, Theorem 1] entails that TRANSITION-COMPATIBLE- $s$ - $t$ -PATH remains NP-complete even when each transition graph is isomorphic to the

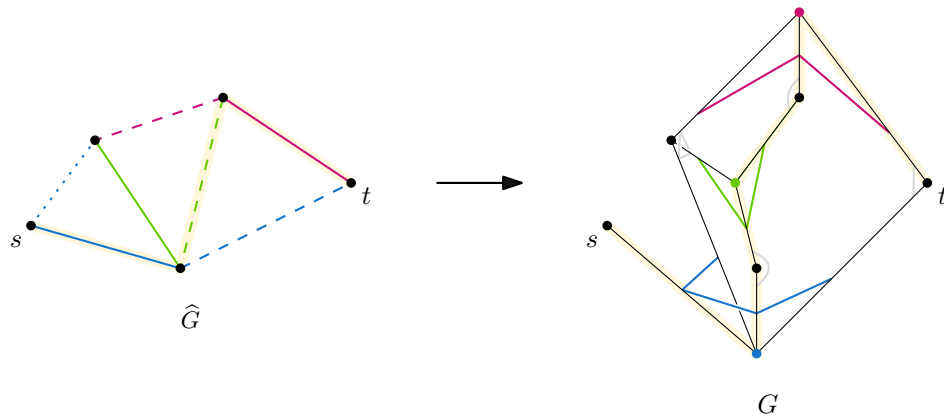


Figure 4.3: Example of a switch graph  $\hat{G}$  (left) and the corresponding graph  $G$  with transition system  $T$  (right), constructed according to Construction 4. Each switch and its corresponding vertex and transition graph are shown in the same color. The transition graphs of original vertices are shown in gray. The yellow-highlighted edges indicate a possible  $s$ - $t$ -path in  $\hat{G}$  under a valid configuration and the corresponding  $T$ -compatible  $s$ - $t$ -path in  $G$ .

lollipop graph  $L_4$ . Applying Construction 2 to such an instance yields a switch graph where each switch-induced graph is also isomorphic to  $L_4$ .

Fig. 4.2 illustrates the application of Construction 3 to a  $\{1\}^*$ -switch  $S$  where  $G(S)$  is isomorphic to  $L_4$ . Let  $V(S) = \{a, b, c, d\}$  and  $E(S) = \{\{a, b\}, \{a, c\}, \{a, d\}, \{b, c\}\}$ . We partition  $V(S)$  into two sets:  $V_1 = \{b, c, d\}$  and  $V_2 = \{a\}$ , such that all edges between the sets are contained in  $E(S)$ . The edge clique cover of  $G(S)[V_1]$  consists of the subsets  $\{b, c\}$  and  $\{d\}$ , and that of  $G(S)[V_2]$  consists of  $\{a\}$ . This results in two star switches:  $S_1 = \{\{bw, cw\}, \{dw\}\}$  and  $S_2 = \{\{aw\}\}$ , both of which are  $\{2, 1\}$ -star switches.

Hence, by applying Construction 3 to each switch, we obtain a switch graph where every switch is a  $\{2, 1\}$ -star switch. Since the original instance of TRANSITION-COMPATIBLE- $s$ - $t$ -PATH is NP-complete and we have established the correctness of both the reduction to STCON-SWITCHING (Lemma 2) and the switch transformation (Lemma 3), we conclude the following result:

**Theorem 4.** STCON-SWITCHING is NP-hard, even when restricted to  $\{2, 1\}$ -star switches.

## 4.2 From Switches to Transition Graphs

We have now seen how to reduce an instance of TRANSITION-COMPATIBLE- $s$ - $t$ -PATH to an instance of STCON-SWITCHING restricted to  $\{1\}^*$ -switches, along with the implications this has for the complexity of STCON-SWITCHING. As previously hinted at, the reduction also works in the reverse direction via the following construction, which is illustrated in Fig. 4.3 by a simple example:

**Construction 4.** Let  $\widehat{G}$  be a switch graph restricted to  $\{1\}^*$ -switches. We construct a corresponding graph  $G$  with transition system  $T$  as follows:

- The vertex set of  $G$  is defined as  $V(G) := V(\widehat{G}) \cup \{v_S \mid S \in \mathcal{S}(\widehat{G})\}$ , that is, it consists of all original vertices of  $\widehat{G}$  along with a fresh *switch vertex*  $v_S$  for each switch  $S \in \mathcal{S}(\widehat{G})$ .
- For each switch  $S \in \mathcal{S}(\widehat{G})$ , we add the edges  $\{uv_S \mid u \in V(S)\}$  to  $E(G)$ , making  $v_S$  adjacent to every vertex involved in  $S$ .
- For each switch vertex  $v_S \in V(G)$ , we define the transition graph  $T(v_S)$  such that  $E(T(v_S)) := \{\{uv_S, wv_S\} \mid uw \in E(S)\}$ . Observe that the transition graph  $T(v_S)$  is isomorphic to the switch-induced graph  $G(S)$ .
- For each original vertex  $v \in V(G)$ , the transition graph  $T(v)$  is the complete graph on its incident edges, that is,  $E(T(v)) := \binom{E_G(v)}{2}$ . This ensures that the transitions of the original vertices are unrestricted.
- Furthermore, we designate the source and target vertices of  $G$  as  $s(G) := s(\widehat{G})$  and  $t(G) := t(\widehat{G})$ , respectively.

Observe that every edge  $uw$  in a switch  $S$  corresponds to an allowed transition in  $T(v_S)$ , enabling a connection between  $u$  and  $w$  through the switch vertex  $v_S$ . Intuitively, this allows any path in a configured version of  $\widehat{G}$  to be transformed into a  $T$ -compatible path in  $G$ . The converse direction follows by an analogous reasoning. This leads to the following lemma:

**Lemma 4.** *Let  $\widehat{G}$  be a switch graph restricted to  $\{1\}^*$ -switches. Then, there exists an  $s$ - $t$ -path in  $\widehat{G}$  under some configuration if and only if the graph  $G$  constructed in Construction 4 admits a  $T$ -compatible  $s$ - $t$ -path.*

*Proof.* ( $\Rightarrow$ ) Let  $c$  be a configuration of  $\widehat{G}$  that admits an  $s$ - $t$ -path  $P := \langle v_1, v_2, \dots, v_k \rangle$ . Since every switch adds at most one edge to  $\widehat{G} \circ c$ , each consecutive pair of vertices  $v_i, v_{i+1}$  in  $P$  can be mapped to a distinct switch  $S_i \in \mathcal{S}(\widehat{G})$  where  $v_i v_{i+1} \in E(S_i)$ . By construction, the pair  $\{v_i v_{S_i}, v_{i+1} v_{S_i}\}$  is an allowed transition in  $T(v_{S_i})$ . Thus,  $P' := \langle v_1, v_{S_1}, v_2, v_{S_2}, \dots, v_{k-1}, v_{S_{k-1}}, v_k \rangle$  is a  $T$ -compatible  $s$ - $t$ -path in  $G$ .

( $\Leftarrow$ ) Let  $P'$  be a  $T$ -compatible  $s$ - $t$ -path in  $G$ . Since every switch vertex is only adjacent to original vertices and vice versa, the path  $P' := \langle v_1, v_{S_1}, v_2, v_{S_2}, \dots, v_{k-1}, v_{S_{k-1}}, v_k \rangle$ , alternates between original vertices and switch vertices. Therefore, for each switch vertex  $v_{S_i}$  within the path, the pair  $\{v_i v_{S_i}, v_{i+1} v_{S_i}\}$  must be an allowed transition in  $T(v_{S_i})$ . By the construction of  $G$ , this implies that  $v_i v_{i+1} \in E(S_i)$ . We create a configuration  $c$  of  $\widehat{G}$  such that  $v_i v_{i+1} \in S_i[c(S_i)]$  for each  $i \in [k-1]$ . Then, path  $P := \langle v_1, v_2, \dots, v_k \rangle$  is an  $s$ - $t$ -path in  $\widehat{G} \circ c$ .  $\square$

Szeider [20, Proposition 9] established that TRANSITION-COMPATIBLE- $s$ - $t$ -PATH can be solved in linear time if each transition graph is a complete multipartite graph. In Construction 4, the transition graph of each original vertex is a complete graph, which is a special case of a complete multipartite graph. Meanwhile, the transition graph of each switch vertex is isomorphic to the switch-induced graph of the corresponding switch. Hence, as long as all switch-induced graphs are complete multipartite, we can reduce to an instance of TRANSITION-COMPATIBLE- $s$ - $t$ -PATH with complete multipartite transition graphs, for which the correctness of the reduction is ensured by Lemma 4. Denoting such switches as *complete multipartite*, we obtain the following result:

**Theorem 5.** *STCON-SWITCHING restricted to complete multipartite  $\{1\}^*$ -switches can be decided in time  $\mathcal{O}(n + m)$ , where  $n := |V(G)|$  and  $m := \sum_{S \in \mathcal{S}} |E(S)|$ .*

*Proof.* Given a switch graph  $\widehat{G}$  with complete multipartite  $\{1\}^*$ -switches, we construct the corresponding graph  $G$  with transition system  $T$  according to Construction 4. As discussed previously, the correctness of this reduction is guaranteed by Lemma 4.

We begin by analyzing the running time of Construction 4. Creating a vertex in  $G$  for each vertex and each switch in  $\widehat{G}$  takes  $\mathcal{O}(n + |\mathcal{S}(\widehat{G})|)$  time. For each switch  $S \in \mathcal{S}(\widehat{G})$ , we add  $|V(S)|$  edges to  $G$  and construct a transition graph for its corresponding switch vertex. This transition graph contains  $|V(S)|$  vertices and  $|E(S)|$  edges. Per switch, this amounts to  $\mathcal{O}(V(S) + (V(S) + E(S)))$  time. Since  $G(S)$  does not contain isolated vertices, we have  $V(S) \leq 2E(S)$ , and thus the total per-switch running time simplifies to  $\mathcal{O}(E(S))$ . Note that the transition graphs for the original vertices need not be stored explicitly and therefore do not affect the overall complexity. Thus, the total running time of the construction becomes

$$\mathcal{O}\left(n + |\mathcal{S}(\widehat{G})| + \sum_{S \in \mathcal{S}(\widehat{G})} |E(S)|\right) = \mathcal{O}(n + |\mathcal{S}(\widehat{G})| + m).$$

Graph  $G$  itself consists of  $n + |\mathcal{S}(\widehat{G})|$  vertices and  $|V(S)|$  edges for each switch  $S \in \mathcal{S}(\widehat{G})$ . Since  $V(S) \leq 2E(S)$ , this amounts to a maximum of  $\sum_{S \in \mathcal{S}(\widehat{G})} 2|E(S)|$  edges in  $G$ . Deciding whether there exists a  $T$ -compatible  $s$ - $t$ -path in  $G$  can then be done in time  $\mathcal{O}(|V(G)| + |E(G)|)$  using the algorithm of Szeider [20, Proposition 9]. This then yields a total running time of

$$\mathcal{O}(|V(G)| + |E(G)|) = \mathcal{O}\left(n + |\mathcal{S}(\widehat{G})| + \sum_{S \in \mathcal{S}(\widehat{G})} 2|E(S)|\right) = \mathcal{O}(n + |\mathcal{S}(\widehat{G})| + m).$$

Consequently, both the reduction and the algorithm for TRANSITION-COMPATIBLE- $s$ - $t$ -PATH run in the same time, so the total running time amounts to  $\mathcal{O}(n + |\mathcal{S}(\widehat{G})| + m)$ . We can safely assume that each switch contains at least one edge, so  $|\mathcal{S}(\widehat{G})| \leq m$  (Switches with no edges are not useful for solving the problem and can be configured arbitrarily). Thus, we can simplify the running time to  $\mathcal{O}(n + m)$ , which concludes the proof.  $\square$

### 4.3 Dichotomy for $\{1\}^*$ -Switches

As a consequence of Theorem 5, if a graph family  $\mathcal{A}$  (closed under isomorphism and containing no isolated vertices<sup>1</sup>) consists solely of complete multipartite graphs, then STCON-SWITCHING restricted to  $\{1\}^*$ - $\mathcal{A}$ -switches can be solved in linear time. What remains is the complementary case: what if  $\mathcal{A}$  contains at least one graph that is not complete multipartite? We show that in this case STCON-SWITCHING becomes NP-hard, thus yielding a strict dichotomy for  $\{1\}^*$ -switches.

To establish this, we prove a stronger statement: let  $H$  be a graph without isolated vertices that is *not* complete multipartite. Then STCON-SWITCHING restricted to  $\{1\}^*$ -switches is NP-hard if every switch-induced graph is isomorphic to  $H$ . The general hardness result follows immediately.

We first characterize such graphs  $H$  in a manner similar to the approach taken by Szeider [20]. Since  $H$  is not complete multipartite, it must contain vertices  $u, v, w \in V(H)$  such that  $uv \in E(H)$  but  $uw, vw \notin E(H)$ . Because  $H$  has no isolated vertices,  $w$  must be adjacent to some other vertex  $x \in V(H)$ . Depending on the adjacency between  $x$  and  $\{u, v\}$ , the vertex-induced subgraph  $H[\{u, v, w, x\}]$  is isomorphic to one of the following:

- $K_2 + K_2$  if  $x$  is not adjacent to  $u$  or  $v$ ,
- $P_4$  if  $x$  is adjacent to exactly one of  $\{u, v\}$ ,
- $L_4$  ( $K_3$  with an additional pendant edge) if  $x$  is adjacent to both  $u$  and  $v$ .

This yields the following characterization:

**Lemma 5.** *A graph  $H$  without isolated vertices is not complete multipartite if and only if  $H$  contains  $P_4$ ,  $L_4$ , or  $K_2 + K_2$  as a vertex-induced subgraph.*

We now analyze these three cases separately.

**$P_4$  or  $L_4$  as a vertex-induced subgraph of  $H$ :** A consequence of Szeider [20, Theorem 1] is that TRANSITION-COMPATIBLE- $s$ - $t$ -PATH is NP-complete even if every transition graph is isomorphic to  $H$ . Given such an instance  $(G, T)$ , we construct a switch graph  $\widehat{G}_T$  restricted to  $\{1\}^*$ -switches using Construction 2, which is a correct reduction to STCON-SWITCHING by Lemma 2. The resulting switch-induced graphs are either isomorphic to  $H$  or to  $K_2$  (from static edges).

To eliminate these  $K_2$ -switches, we replace each static edge  $S_e$  with a  $\{1\}^*$ -switch  $S'_e$  whose switch-induced graph is isomorphic to  $H$ , containing  $e$  in one of its positions. Except for the two endpoints of  $e$ , the vertex set consists of fresh vertices unique to  $S'_e$ . Since only the position containing  $e$  can help towards  $s$ - $t$ -connectivity,  $S'_e$

<sup>1</sup>By definition, the switch-induced subgraph of any switch has no isolated vertices, since its vertex set consists of the endpoints of its edges.

behaves exactly like the original static edge. Thus, all switch-induced graphs are isomorphic to  $H$ , and STCON-SWITCHING is NP-hard in this setting.

**$K_2 + K_2$  as a vertex-induced subgraph of  $H$ :** Despite the fact that TRANSITION-COMPATIBLE- $s-t$ -PATH is *not* NP-complete when all transition graphs are isomorphic to  $H$ , it *is* NP-complete when the transition graphs are restricted to  $\{H, K_3\}$  [20, Theorem 1]. Reducing such an instance  $(G, T)$  to a switch graph  $\hat{G}_T$  via Construction 2 (as in the previous case) yields switch-induced graphs isomorphic to  $H$ ,  $K_3$ , or  $K_2$ .

Applying Construction 3 to each  $K_3$ -switch transforms it into three static edges (since  $K_3$  is complete tripartite), which is correct reduction by Lemma 3. We then replace all static edges as in the previous case, thus obtaining a switch graph where every switch-induced graph is isomorphic to  $H$ . Hence, STCON-SWITCHING is NP-hard in this setting as well.

This proves the following dichotomy theorem:

**Theorem 6.** *Let  $\mathcal{A}$  be a family of graphs closed under isomorphism and containing no isolated vertices. Then STCON-SWITCHING restricted to  $\{1\}^*$ - $\mathcal{A}$ -switches is solvable in linear time if  $\mathcal{A}$  consists only of complete multipartite graphs; otherwise it is NP-hard.*

**Discussion** In summary, this chapter has established a tight correspondence between TRANSITION-COMPATIBLE- $s-t$ -PATH and STCON-SWITCHING, providing reductions in both directions. These reductions not only clarify the structural relationship between the two models but also yield a strict complexity dichotomy for STCON-SWITCHING restricted to  $\{1\}^*$ -switches (Theorem 6).

Notably, Katz et al. [11, Theorem 7] showed that STCON-SWITCHING can be decided in the same running time as in Theorem 6 for  $\{1\}^*$ -star switches, which form a special case of complete multipartite  $\{1\}^*$ -switches (since star graphs are complete bipartite graphs). Interestingly, their algorithm follows the same core idea as that of Szeider [20] for TRANSITION-COMPATIBLE- $s-t$ -PATH: Both approaches construct an auxiliary graph along with a matching and determine the existence of an  $s-t$ -path by checking for an augmenting path in the auxiliary graph.

In this chapter, we also established the NP-hardness of STCON-SWITCHING for switches with more than one edge per position, specifically for  $\{2, 1\}$ -star switches (Theorem 4). This shows that it is not sufficient for switches to be complete multipartite to guarantee tractability once switches contain multiple edges per position. It would therefore be interesting to explore whether a dichotomy result can still be obtained for such more general switches. Moreover, our discussion so far has been limited to the unweighted setting. Therefore, in Chapter 6, we continue the investigation of STCON-SWITCHING, focusing on additional restrictions of the union graph and extending the analysis to the weighted variant.

# The Complexity of (Weighted) Satisfiability

To show hardness for switching problems of various graph families — both in the weighted and unweighted variant — we will reduce from known hard problems. A natural candidate for such reductions is the (weighted) satisfiability problem, which is known to be intractable even for various restrictions on the formula (for instance satisfiability for 3-CNF formulas is NP-hard).

In Chapters 6 to 8, we transform propositional formulas into switch graphs whose structure depends on the target graph family, such as globally connected graphs or graphs that do (or do not) contain an  $s$ - $t$ -path, thereby establishing hardness results for the corresponding switching problems. Interestingly, structural restrictions on the formula translate into structural constraints on the resulting switch graph. For instance, bounding the number of variable occurrences leads to smaller switches, while the formula's depth often results in a structurally simpler union graph.

In this chapter, we examine various such restrictions on formulas that preserve the hardness of the satisfiability problem. In doing so, we resolve an open question posed by Kanj and Szeider [10] concerning the parameterized complexity of weighted satisfiability for a particular class of formulas. These restricted satisfiability problems will later serve as the foundation for proving hardness in Chapters 6 to 8, even for highly constrained switch graphs.

We begin by defining some well-known classes of propositional formulas explored in this chapter. Recall that  $\Phi$  denotes the set of all propositional formulas and  $\mathcal{V}$  denotes the set of propositional variables.

A propositional formula  $\varphi \in \Phi$  is in *negation normal form* (NNF) if negations are only applied directly to variables. For the reductions to switching problems, we will always

reduce from formulas in this form. Every propositional formula in  $\Phi$  can be brought into negation normal form in linear time using De Morgan's laws to push the negation inward, and eliminating double negation.

Let  $\mathcal{L}$  denote the set of literals, that is, each variable in  $\mathcal{V}$  and its negation:

$$\mathcal{L} = \{\neg x, x \mid x \in \mathcal{V}\}.$$

For  $t \geq 1$  and  $d \geq 1$ , let  $\Delta_{t,d}$  and  $\nabla_{t,d}$  be the sets of formulas built from alternating conjunctions and disjunctions over  $t$  layers, ending with literals. Formulas in  $\Delta_{t,d}$  start with a conjunction at the outermost level, while those in  $\nabla_{t,d}$  start with a disjunction. Parameter  $t$  bounds the depth of the formula. Parameter  $d$  limits the number of literals that can appear in each disjunction or conjunction at the innermost layer. Formally, the sets are inductively defined as follows:

- **Base case** ( $t = 1$ ): The sets  $\Delta_{1,d}$  and  $\nabla_{1,d}$  are conjunctions and disjunctions of at most  $d$  literals, respectively:

$$\Delta_{1,d} = \left\{ \bigwedge_{\ell \in L} \ell \mid L \subseteq \mathcal{L}, |L| \leq d \right\} \quad \nabla_{1,d} = \left\{ \bigvee_{\ell \in L} \ell \mid L \subseteq \mathcal{L}, |L| \leq d \right\}$$

- **Inductive step** ( $t \geq 2$ ):  $\Delta_{t,d}$  consists of conjunctions of formulas from  $\nabla_{t-1,d}$  while  $\nabla_{t,d}$  consists of disjunctions of formulas from  $\Delta_{t-1,d}$ :

$$\Delta_{t,d} = \left\{ \bigwedge_{\varphi \in F} \varphi \mid F \subseteq \nabla_{t-1,d} \right\} \quad \nabla_{t,d} = \left\{ \bigvee_{\varphi \in F} \varphi \mid F \subseteq \Delta_{t-1,d} \right\}$$

If we do not want to bound the number of literals at the innermost level, we simply write  $\Delta_t := \Delta_{t,\infty}$  and  $\nabla_t := \nabla_{t,\infty}$ .

The set  $\Delta_t$  denotes the set of  $t$ -normalized formulas. For example, the formula

$$(x_2 \vee \neg x_3 \vee \neg x_4) \wedge (x_1 \vee x_2 \vee \neg x_4) \wedge (x_1 \vee \neg x_2). \quad (5.1)$$

is 2-normalized. Observe that every formula in  $\Delta_t$  for  $t \geq 1$  is certainly in negation normal form. Moreover,  $\Delta_2$  represents the set of formulas in *conjunctive normal form* (CNF). For the reductions to switching problems, the choice of  $t$  imposes specific constraints on the structure of the union graph. In the case of (WEIGHTED)-GCON-SWITCHING, the described reduction requires that  $t \leq 2$ . Note that the formula in Eq. (9.1) also belongs to  $\Delta_{2,3}$ , the subset of CNF formulas in which each clause contains at most three literals, denoted 3-CNF.

For  $d \geq 2$ , let  $B_d$  denote the set of  $d$ -bounded formulas, consisting of all formulas in which each variable appears in at most  $d$  literals. Moreover, for  $c \geq 0$  and  $d \geq 0$ , let  $B_{c,d}$  denote the class of  $c$ - $d$ -bounded formulas, where each variable appears in at most  $c$  literals in one polarity (negated or unnegated) and at most  $d$  literals in the other polarity. For example, the formula in Eq. (5.1) is in  $B_3$  as well as in  $B_{2,1}$ . For the reductions to

switching problems, we model each variable by an on-off-switch, and these bounds on variable occurrences correlate with the number of edges in the positions, thus limiting the size of switches.

Finally, for any formula  $\varphi \in \Delta_2$  in CNF, let  $G(\varphi)$  denote the *associated graph* of  $\varphi$ . In this graph, the variables and clauses of  $\varphi$  are represented as vertices, with edges modeling their relationships. Additionally,  $G(\varphi)$  contains a simple cycle connecting all variable vertices. We are particularly interested in 3-CNF formulas whose associated graph is planar, denoted by  $\Delta_{2,3}^{\text{planar}}$ . The planarity of  $G(\varphi)$  plays a key role in our reduction to GCON-SWITCHING, as it ensures that the resulting switch graph is planar, while ensuring that each switch forms a star and each vertex participates in only a bounded number of switches. Formally, let  $\mathcal{V}(\varphi) := \{x_1, \dots, x_n\}$  be the set of variables, and let  $\mathcal{C}(\varphi)$  denote the set of clauses in  $\varphi$ , where each clause is a set of literals. Then,  $G(\varphi)$  is defined as follows:

$$\begin{aligned} V(G(\varphi)) &:= \mathcal{V}(\varphi) \cup \mathcal{C}(\varphi) \\ E(G(\varphi)) &:= \{x_1x_2, x_2x_3, \dots, x_nx_1\} \\ &\quad \cup \{xC \mid x \in C \text{ or } \neg x \in C, \text{ for } x \in \mathcal{V}(\varphi) \text{ and } C \in \mathcal{C}(\varphi)\} \end{aligned}$$

In this chapter, we first study the complexity of the unweighted satisfiability problem in Section 5.1, showing that it remains NP-hard for 3-CNF formulas that have planar associated graphs and in which each variable appears at most three times. We then show a related hardness result for satisfiability when restricted to 2-bounded 3-normalized formulas, trading conjunctive normal form for a stronger bound on variable occurrences. In Section 5.2, we turn to the weighted satisfiability problem and recapitulate that, for  $t$ -normalized formulas, it is  $W[t]$ -complete. Through a series of closely related reductions from PARTITIONED SUBGRAPH ISOMORPHISM to weighted satisfiability, we show that, for various combinations of formula depth and bounds on variable occurrences, the weighted satisfiability problem is  $W[1]$ -hard, and, unless the ETH fails, cannot be solved in subexponential time. In particular, we prove this for 3-normalized 1-1-bounded formulas and 2-normalized 2-1-bounded formulas. This also resolves an open question posed by Kanj and Szeider [10] concerning the parameterized complexity of  $\text{WSAT}(\Delta_3 \cap B_2)$ .

## 5.1 Unweighted Satisfiability

For our reductions to unweighted switching problems, we start from the unweighted satisfiability problem. It is well known that satisfiability remains NP-complete even when restricted to 3-CNF formulas. Lichtenstein [14] further showed that this hardness persists when the associated graphs of the formulas are planar. By reducing such formulas to switch graphs, we already obtain certain structural properties of the union graph. However, the size of the switches can still be unbounded. To ensure smaller switches, we have to restrict the number of variable occurrences. Tovey [22][Theorem 2.1] proved that satisfiability for 3-CNF formulas remains NP-hard even when each variable appears

at most three times. Manuch and Gaur [15] extended this result to the case where the associated graph is planar. We summarize these results in the following theorem:

**Theorem 7** ([15]).  $\text{SAT}(\Delta_{2,3}^{\text{PLANAR}} \cap B_{2,1})$  is NP-hard.

*Proof Sketch.* We reduce from an instance of  $\text{SAT}(\Delta_{2,3}^{\text{PLANAR}})$ . Given a formula  $\varphi \in \Delta_{2,3}^{\text{PLANAR}}$ , the transformation to an equisatisfiable formula with bounded variable occurrences is done as follows: First, for each variable  $x \in \mathcal{V}(\varphi)$ , every occurrence of  $x$  in  $\varphi$  is replaced by a unique new variable  $x_i$  with  $i \in [k]$ , where  $k$  is the number of occurrences of  $x$  in  $\varphi$ . To ensure that the new variables  $x_1, \dots, x_k$  all have the same truth value in a satisfying interpretation, we add the following chain of implications to the formula:

$$(x_1 \Rightarrow x_2) \wedge (x_2 \Rightarrow x_3) \wedge \dots \wedge (x_{k-1} \Rightarrow x_k) \wedge (x_k \Rightarrow x_1).$$

We can rewrite each such implication as a simple disjunction, resulting in additional clauses with two literals per clause. Every new variable appears in both polarities exactly once in these additional clauses, ensuring that the formula is 2-1-bounded. Furthermore, Tippenhauer and Muzler [21, Figure 8.1] demonstrate that the associated graph remains planar after the transformation and show how we can add the new variables to the variable cycle.  $\square$

When attempting to further reduce the number of variable occurrences, we must give up the requirement that the formula is in conjunctive normal form. Indeed, Tovey [22, Lemma 3.2] showed that  $\text{SAT}(\Delta_2 \cap B_2)$  can be solved in polynomial time. Nevertheless, it is possible to transform a 3-CNF formula in which each variable appears at most three times into an equisatisfiable 3-normalized formula where each variable occurs exactly once in both polarities. This leads to the following theorem:

**Theorem 8.**  $\text{SAT}(\Delta_3 \cap B_{1,1})$  is NP-hard.

*Proof.* We reduce from an instance of  $\text{SAT}(\Delta_{2,3} \cap B_{2,1})$ . Suppose we are given a formula  $\varphi \in \Delta_{2,3} \cap B_{2,1}$ . Without loss of generality, we assume that each variable occurring three times appears twice negated and once unnegated; otherwise, we can invert its polarities without affecting satisfiability. Similarly, for each variable with exactly two occurrences, we can assume that it does not appear exclusively in one polarity; otherwise, it can be removed from the formula without affecting satisfiability. For each variable  $x \in \mathcal{V}(\varphi)$  with three occurrences, we replace the two negated occurrences of variable  $x$  by  $x'$  and  $x''$  (keeping the sign), and add the following conjunct to the formula:

$$x \Rightarrow (x' \wedge x'').$$

Observe that after rewriting the implications as disjunctions, that resulting formula  $\varphi'$  is clearly in  $\Delta_3 \cap B_{1,1}$ . It remains to show that  $\varphi$  is satisfiable if and only if  $\varphi'$  is satisfiable.

( $\Rightarrow$ ) Suppose that  $\varphi$  is satisfiable. Then, there exists a truth assignment  $I$  such that  $I(\varphi) = 1$ . We can extend  $I$  to a truth assignment  $I'$  for  $\varphi'$  by setting  $I'(x') = I'(x'') = I(x)$

for every variable  $x \in \mathcal{V}(\varphi)$  with three occurrences in  $\varphi$ . Since the new implications are satisfied by this assignment, we have  $I'(\varphi') = 1$ .

( $\Leftarrow$ ) Suppose that  $\varphi'$  is satisfiable. Then, there exists a truth assignment  $I'$  such that  $I'(\varphi') = 1$ . Define  $I := I'|_{\mathcal{V}(\varphi)}$  as the restriction of  $I'$  to the variables in  $\mathcal{V}(\varphi)$ . Consider some arbitrary clause of  $\varphi$ . The corresponding conjunct in  $\varphi'$  must contain a literal  $\ell$  that is satisfied by  $I'$ . If  $\ell$  is identical to a literal in the original clause, then the clause is clearly satisfied by  $I$ . Otherwise,  $\ell$  must be the negation of a new variable  $x'$  or  $x''$  introduced for some variable  $x \in \mathcal{V}(\varphi)$ . Suppose  $\ell := \neg x'$ . Then  $I'(x') = 0$ , and since  $I'(x \Rightarrow (x' \wedge x'')) = 1$ , it follows that  $I'(x) = 0$ . Thus, the original clause in  $\varphi$ , which contains  $\neg x$ , is satisfied by  $I$ . The case for  $\ell := \neg x''$  is analogous. Consequently, we have  $I(\varphi) = 1$ .  $\square$

## 5.2 Weighted Satisfiability

Unsurprisingly, for our reductions to weighted switching problems, we reduce from the weighted satisfiability problem. In this context, we are particularly interested in the parameterized complexity of the problem. A natural parameterization is the maximum weight of a satisfying truth assignment. In the reductions described in Chapters 6 to 8, this parameter translates to the maximum weight of a witnessing configuration. In general, it is well known that for every  $t \geq 2$ ,  $\text{WSAT}(\Delta_t)$  is  $W[t]$ -complete [6, Theorem 7.1].

Since this does not impose a bound on the number of variable occurrences, the size of switches in the corresponding switch graphs remains unrestricted. To address this, we proceed by analyzing the parameterized complexity of weighted satisfiability of  $t$ -normalized formulas with additional constraints on variable occurrences.

To this end, we consider a variant of **SUBGRAPH ISOMORPHISM**, which serves as the base for a reduction to weighted satisfiability. In the general problem, we are given two graphs  $G$  and  $H$  and ask whether there exists a subgraph of  $G$  that is isomorphic to  $H$ . Specifically, we seek an injective mapping  $\phi : V(H) \rightarrow V(G)$  such that for every edge  $xy \in E(H)$ , the edge  $\phi(x)\phi(y)$  exists in  $G$ . The specific version of the problem used in the parameterized reduction is known to be  $W[1]$ -hard and cannot be solved in subexponential time unless the **ETH** fails. These hardness results carry over to the weighted satisfiability problem via the reduction. Moreover, the provided reductions ensure that the resulting formulas are  $t$ -normalized for a fixed constant  $t$ , and that the number of variable occurrences is bounded by a constant.

The variant relevant to us is called **PARTITIONED SUBGRAPH ISOMORPHISM**. In this version of the problem, we are additionally given a partition  $\mathcal{P}$  of  $V(G)$  such that there is a non-empty set  $P_x \in \mathcal{P}$  for each vertex  $x \in V(H)$ . The subgraph isomorphism has to respect this partition, that is, for every  $x \in V(H)$ , we must have that  $\phi(x) \in P_x$ . Marx [16] has proven the following theorem:

**Theorem 9.** [16, Corollary 6.3] *If PARTITIONED SUBGRAPH ISOMORPHISM can be decided in time  $f(k)n^{o(\frac{k}{\log k})}$ , where  $f$  is some computable function,  $n$  the number of vertices in the host graph  $G$ , and  $k$  is the number of edges in the pattern graph  $H$ , then the ETH fails.*

A problem that is closely related to PARTITIONED SUBGRAPH ISOMORPHISM is MULTICOLORED CLIQUE. Here, the input consists of a graph  $G$ , an integer  $\kappa$ , and a vertex coloring function  $\chi : V(G) \rightarrow [\kappa]$ , and the aim is to decide whether there exists a subgraph of  $G$  isomorphic to  $K_\kappa$  with each vertex of a different color. It is known that MULTICOLORED CLIQUE is W[1]-hard parameterized by  $\kappa$  [4, Theorem 13.25].

Observe that MULTICOLORED CLIQUE is a special case of PARTITIONED SUBGRAPH ISOMORPHISM — any instance of MULTICOLORED CLIQUE can be reduced to an instance of PARTITIONED SUBGRAPH ISOMORPHISM by setting  $H := K_\kappa$  with  $V(H) := [\kappa]$ , keeping  $G$  unchanged, and defining the partition  $\mathcal{P}$  of  $V(G)$  based on the vertex coloring, that is, for each color  $i \in [\kappa]$  we have a set  $P_i := \chi^{-1}(i)$  in  $\mathcal{P}$ . Since  $H$  has  $k := \binom{\kappa}{2}$  edges, we obtain a correct parameterized reduction, from which Theorem [10] follows.

**Theorem 10.** PARTITIONED SUBGRAPH ISOMORPHISM is W[1]-hard parameterized by  $k$ , where  $k$  is the number of edges in the pattern graph  $H$ .

We now describe how to reduce an instance of the problem to an instance of weighted satisfiability. Starting with a construction of a 2-normalized formula — initially without a bound on the number of variable occurrences — we will later refine the construction to ensure that each variable appears only a constant number of times.

**Construction 5.** Given an instance  $(G, H, \mathcal{P})$  of PARTITIONED SUBGRAPH ISOMORPHISM, we construct a propositional formula  $\varphi$  whose variable set is defined as  $\mathcal{V}(\varphi) := V(G) \cup E(G)$ . In the context of the formula, each vertex  $v \in V(G)$  is referred to as a *vertex variable*, and each edge  $uv \in E(G)$  as an *edge variable*. The formula  $\varphi$  is then defined as follows:

$$\varphi := \bigwedge_{x \in V(H)} \left( \bigvee_{v \in P_x} v \right) \wedge \bigwedge_{\substack{x \in V(H) \\ y \in N_H(x) \\ v \in P_x}} \left( v \Rightarrow \bigvee_{u \in N_G(v) \cap P_y} uv \right)$$

Intuitively, setting a variable in  $\mathcal{V}(\varphi)$  to true indicates that the corresponding vertex or edge is selected as part of subgraph. The first part of the conjunction ensures that for every vertex  $x \in V(H)$ , at least one vertex  $v$  is selected from the set  $P_x$ . The second part of the conjunction guarantees that for every vertex  $x \in V(H)$  and each incident edge  $xy \in E(H)$ , the subgraph includes a corresponding edge from the selected vertex in  $P_x$  to some vertex in  $P_y$ . These conditions are necessary for the subgraph to be isomorphic

to  $H$ , but they are not sufficient on their own. This is where the weight constraint comes into play. The maximum weight of a satisfying truth assignment is bounded by:

$$k := |V(H)| + |E(H)|.$$

This bound effectively restricts the selection to at most  $|V(H)|$  vertices and  $|E(H)|$  edges and thus guarantees that the selected subgraph is isomorphic to  $H$ .

Observe that  $\varphi$  is in conjunctive normal form, as each implication can be rewritten as a disjunction with the left side negated. This yields the following lemma:

**Lemma 6.**  *$(G, H, \mathcal{P})$  is a yes-instance of PARTITIONED SUBGRAPH ISOMORPHISM if and only if  $(\varphi, k)$  is a yes-instance of WSAT( $\Delta_2$ ).*

*Proof.* We aim to show that there exists a subgraph of  $G$  isomorphic to  $H$  while respecting the partition  $\mathcal{P}$  if and only if there exists a truth assignment  $I$  such that  $I(\varphi) = 1$  and  $|I|_1 \leq k$ . We will prove both directions separately.

( $\Rightarrow$ ) Suppose there exists an injective mapping  $\phi : V(H) \rightarrow V(G)$  such that for every edge  $xy \in E(H)$ ,  $\phi(x)\phi(y) \in E(G)$ , and for every vertex  $x \in V(H)$ , we have  $\phi(x) \in P_x$ . We construct a truth assignment  $I$  where for every vertex  $x \in V(H)$ ,  $I(\phi(x)) = 1$  and for every edge  $xy \in E(H)$ ,  $I(\phi(x)\phi(y)) = 1$ . All other variables are set to false. By this construction, we have  $|I|_1 = |V(H)| + |E(H)| \leq k$ . It remains to be shown that  $I(\varphi) = 1$ .

For the first part of the conjunction, consider an arbitrary vertex  $x \in V(H)$ . For  $\phi(x) \in P_x$ , we have  $I(\phi(x)) = 1$  and, hence,  $I(\bigvee_{v \in P_x} v) = 1$ .

For the second part of the conjunction, consider an arbitrary vertex  $x \in V(H)$ , one of its neighbors  $y$  in  $H$ , and an arbitrary vertex  $v \in P_x$ . If  $v \neq \phi(x)$ , then  $I(v) = 0$  and the implication  $v \Rightarrow \bigvee_{u \in N_G(v) \cap P_y} uv$  is vacuously true. If  $v = \phi(x)$ , observe that for edge  $xy$ , we must have  $\phi(x)\phi(y) \in E(G)$  and  $\phi(y) \in P_y$ . By construction, we have  $I(\phi(x)\phi(y)) = 1$ , and, consequently,  $I(\bigvee_{u \in N_G(v) \cap P_y} uv) = 1$ . Thus, the implication  $v \Rightarrow \bigvee_{u \in N_G(v) \cap P_y} uv$  evaluates to true under  $I$ .

Therefore, all conjuncts of  $\varphi$  are satisfied, and we conclude that  $I(\varphi) = 1$ .

( $\Leftarrow$ ) Suppose there exists a truth assignment  $I$  such that  $I(\varphi) = 1$  and  $|I|_1 \leq k$ . We begin by analyzing the structure of  $I$ .

Formula  $\varphi$  is satisfied if every conjunct is true. The first part of the conjunction contains a conjunct for every  $x \in V(H)$ , which is a disjunction of all vertex variables in  $P_x$ . Hence, this conjunct is satisfied if  $P_x$  contains at least one vertex variable that evaluates to true. Therefore, at least  $|V(H)|$  vertex variables from  $V(G)$  are set to true.

The second part of the conjunction contains a conjunct for each triple  $(x, y, v)$  with  $x \in V(H)$ ,  $y \in N_H(x)$ , and  $v \in P_x$ . Each conjunct is an implication which is vacuously true if  $v$  is false. Otherwise, the disjunction on the right side must be satisfied, which is the case if at least one edge variable  $uv \in E(G)$  with  $u \in P_y$  evaluates to true. We

know that at least one vertex variable in  $P_x$  evaluates to true, resulting in a minimum of  $\sum_{x \in V(H)} \deg_H(x) = 2|E(H)|$  implications where the right side must be satisfied. As each edge variable  $uv \in E(G)$  appears in exactly two implications, at least  $|E(H)|$  edge variables from  $E(G)$  must be set to true.

This amounts to a minimum of  $|V(H)| + |E(H)|$  variables set to true in  $I$ . However, we know that  $|I|_1 \leq k = |V(H)| + |E(H)|$ . Thus, precisely  $|V(H)|$  vertex variables and  $|E(H)|$  edge variables are set to true. This implies that for each vertex  $x \in V(H)$ , exactly one vertex variable  $v \in P_x$  is set to true. Furthermore, for every vertex variable  $v \in V(G)$  where  $I(v) = 0$ , the edge variables in each corresponding implication must all be false. Thus, for each edge variable  $uv \in E(G)$  where  $I(uv) = 1$ , we have that  $I(v) = 1$  and  $I(u) = 1$ .

We now define a mapping  $\phi : V(H) \rightarrow V(G)$  such that for each  $x \in V(H)$ , we have  $\phi(x)$  mapped to the unique vertex  $v \in P_x$  where  $I(v) = 1$ . To prove that  $\phi$  constitutes a subgraph isomorphism, we must verify two conditions: (1) for every edge  $xy \in E(H)$ , the edge  $\phi(x)\phi(y)$  exists in  $E(G)$ , and (2) for every vertex  $x \in V(H)$ , we have  $\phi(x) \in P_x$ .

The second condition holds by construction. To show the first condition, consider an arbitrary edge  $xy \in E(H)$ . The second part of the conjunction contains an implication where  $v = \phi(x)$  forms the left side and the right side is a disjunction of all edge variables  $uv \in E(G)$  with  $u \in P_y$ . Since  $I(v) = 1$ , one such edge variable  $uv \in E(G)$  must be true because the right side of the implication has to be satisfied. By the previous argument, we know that  $I(u) = 1$ . According to the definition of  $\phi$ , it must be the case that  $u = \phi(y)$ . Thus,  $\phi(x)\phi(y) \in E(G)$ , proving the first condition as well. We conclude that  $\phi$  indeed constitutes a subgraph isomorphism.  $\square$

### 5.2.1 2-Bounded 3-Normalized Formulas

As previously mentioned, the definition of  $\varphi$  in Construction 5 does not impose a bound on the number of variable occurrences. To achieve this, we will present several modifications of Construction 5 which constrains the occurrences of each variable in a certain way. Recall that in our reductions in Chapters 6 to 8, the number of variable occurrences corresponds to the size of the switches. Therefore, limiting the number of variable occurrences results in smaller switches. The modifications are inspired by the techniques used in Section 5.1 to show NP-hardness of unweighted satisfiability under bounded variable occurrences. However, in the weighted setting, we need to carefully adapt these techniques to account for the weight constraints.

**Construction 6.** In this first modified version of Construction 5, we split each vertex variable  $v \in V(G)$  into multiple variables. Specifically, we define a propositional formula  $\varphi^\blacktriangle$  with variable set:

$$\mathcal{V}(\varphi^\blacktriangle) := \{v_y \mid x \in V(H), v \in P_x, y \in N_H(x)\} \cup E(G).$$

Observe that for every  $x \in V(H)$  and each  $v \in P_x$ , we create exactly  $\deg_H(x)$  copies of  $v$ . These variables will again be referred to as *vertex variables*. Then, the propositional

formula  $\varphi^\blacktriangle$  is defined as follows:

$$\varphi^\blacktriangle := \bigwedge_{x \in V(H)} \left( \bigvee_{v \in P_x} \left( \bigwedge_{y \in N_H(x)} v_y \right) \right) \wedge \bigwedge_{\substack{x \in V(H) \\ y \in N_H(x) \\ v \in P_x}} \left( v_y \Rightarrow \bigvee_{u \in N_G(v) \cap P_y} uv \right)$$

Intuitively, in the first part of the conjunction, each vertex variable  $v \in P_x$  for  $x \in V(H)$  is replaced with a conjunction of all its copies. Consequently, in order for vertex  $v$  to be selected from the set  $P_x$ , all  $\deg_H(x)$  corresponding vertex variables must be set to true. In the second part of the conjunction, each vertex variable  $v$  is simply replaced by the specific copy  $v_y$  corresponding to the neighbor  $y$  of  $x$  handled in that conjunct. Since all copies of  $v \in V(G)$  must evaluate to true in order for  $v$  to be selected, a total of  $\sum_{x \in V(H)} \deg_H(x) = 2|E(H)|$  vertex variables must be set to true to ensure that one vertex is chosen from each partition set in  $\mathcal{P}$ . Accordingly, we update the maximum allowed weight of a satisfying truth assignment to:

$$k^\blacktriangle := 3|E(H)|.$$

Observe that after replacing the implication by its disjunctive form, the formula  $\varphi^\blacktriangle$  is 3-normalized and each variable appears in at most two literals. This leads us to following lemma:

**Lemma 7.**  *$(G, H, \mathcal{P})$  is a yes-instance of PARTITIONED SUBGRAPH ISOMORPHISM if and only if  $(\varphi^\blacktriangle, k^\blacktriangle)$  is a yes-instance of WSAT( $\Delta_3 \cap B_2$ ).*

*Proof.* To prove this lemma, we show that  $(\varphi, k)$  admits a satisfying truth assignment  $I$  with weight  $|I|_1 \leq k$  if and only if  $(\varphi^\blacktriangle, k^\blacktriangle)$  from Construction 5 admits a satisfying truth assignment  $I^\blacktriangle$  with  $|I^\blacktriangle|_1 \leq k^\blacktriangle$ . Since Lemma 6 establishes that  $(\varphi, k)$  is a yes-instance of WSAT( $\Delta_2$ ) if and only if  $(G, H, \mathcal{P})$  is a yes-instance of PARTITIONED SUBGRAPH ISOMORPHISM, the desired result then follows. We prove both directions separately.

( $\Rightarrow$ ) Suppose there exists a truth assignment  $I$  of  $\varphi$  such that  $I(\varphi) = 1$  and  $|I|_1 \leq k$ . We construct a corresponding truth assignment  $I^\blacktriangle$  of  $\varphi^\blacktriangle$  as follows: for every  $x \in V(H)$ , every  $v \in P_x$ , and every neighbor  $y \in N_H(x)$ , we set  $I^\blacktriangle(v_y) := I(v)$ . For every edge  $uv \in E(G)$ , we set  $I^\blacktriangle(uv) := I(uv)$ . It is easy to see that  $I^\blacktriangle(\varphi^\blacktriangle) = I(\varphi) = 1$ . To determine the number of variables set to true in  $I^\blacktriangle$ , recall from the proof of Lemma 6 that precisely one vertex variable  $v \in P_x$  for each  $x \in V(H)$  and precisely  $|E(H)|$  edge variables are set to true in  $I$ . If  $v \in P_x$  is set true, then the corresponding  $\deg_H(x)$  vertex variables of  $v$  are set to true in  $I^\blacktriangle$ . Thus, we have  $|I^\blacktriangle|_1 = \sum_{x \in V(H)} \deg_H(x) + |E(H)| = 3|E(H)| \leq k^\blacktriangle$ . It follows that  $I^\blacktriangle$  is a yes-instance of WSAT( $\Delta_3 \cap B_2$ ).

( $\Leftarrow$ ) Suppose there exists a truth assignment  $I^\blacktriangle$  such that  $I^\blacktriangle(\varphi^\blacktriangle) = 1$  and  $|I^\blacktriangle|_1 \leq k^\blacktriangle$ . We begin by analyzing the structure of  $I^\blacktriangle$ .

For each conjunct in the first part of  $\varphi^\blacktriangle$  to be satisfied, it must hold that for each  $x \in V(H)$ , there exists at least one vertex  $v \in P_x$  where all  $\deg_H(x)$  corresponding vertex variables are set to true. This amounts to a minimum of  $\sum_{x \in V(H)} \deg_H(x) = 2|E(H)|$  vertex variables set to true in  $I^\blacktriangle$ . Since each vertex variable forms the left side of a unique implication in the second part of the conjunction, there are at least  $2|E(H)|$  implications where the right side must evaluate to true in order for the implication to be satisfied. As the right side is a disjunction of edge variables and each edge variable  $uv \in E(G)$  appears in exactly two implications, at least  $|E(H)|$  edge variables from  $E(G)$  must be set to true for all implications to be satisfied.

This amounts to a minimum of  $3|E(H)|$  variables set to true in  $I^\blacktriangle$ . However, we know that  $|I^\blacktriangle|_1 \leq k^\blacktriangle = 3|E(H)|$ , therefore, precisely  $2|E(H)|$  vertex variables and  $|E(H)|$  edge variables are set to true.

We now construct a truth assignment  $I$  for  $\varphi$  as follows: for every  $x \in V(H)$ , choose one vertex  $v \in P_x$  where all the corresponding vertex variables in  $\varphi^\blacktriangle$  are set to true in  $I^\blacktriangle$ , and set  $I(v) := 1$ . All other vertex variables are set to false. For every edge variable  $uv \in E(G)$ , we set  $I(uv) := I^\blacktriangle(uv)$ . Observe that  $I^\blacktriangle$  is a satisfying truth assignment of  $\varphi^\blacktriangle$ . Moreover, this amounts to exactly  $|V(H)|$  vertex variables and  $|E(H)|$  edge variables set to true in  $I$ , thus  $|I|_1 = |V(H)| + |E(H)| \leq k$ . Thus,  $I$  is a yes-instance of  $\text{WSAT}(\Delta_2)$ .  $\square$

From Lemma 7 we obtain the following Theorem 11, thereby resolving an open problem posed by Kanj and Szeider [10] regarding the  $W$ -hardness of  $\text{WSAT}(\Delta_3 \cap B_2)$ . Note that the authors work with normalized circuits of bounded depth, rather than  $t$ -normalized formulas. Nevertheless, our result on formulas translates directly to their circuit setting.

**Theorem 11.**  *$\text{WSAT}(\Delta_3 \cap B_2)$  is  $W[1]$ -hard parameterized by  $k$ , and, assuming the ETH, cannot be solved in time  $f(k)n^{o(\frac{k}{\log k})}$ , where  $f$  is some computable function,  $k$  is the maximum weight of a satisfying truth assignment, and  $n$  is the number of variables in the formula.*

*Proof.* Given an instance  $(G, H, \mathcal{P})$  of PARTITIONED SUBGRAPH ISOMORPHISM, Construction 6 defines a corresponding instance  $(\varphi^\blacktriangle, k^\blacktriangle)$  of  $\text{WSAT}(\Delta_3 \cap B_2)$ . Let  $m := |V(G)|$ ,  $\ell := |E(H)|$ ,  $n := |\mathcal{V}(\varphi^\blacktriangle)|$  and  $k := k^\blacktriangle$ .

We start by showing that this construction yields a parameterized reduction from PARTITIONED SUBGRAPH ISOMORPHISM, parameterized by  $\ell$ , to  $\text{WSAT}(\Delta_3 \cap B_2)$ , parameterized by  $k$ . The correctness of the reduction follows from Lemma 7. We can bound the parameter  $k$  by some computable function of  $\ell$  since  $k = 3\ell$ . To analyze the running time of the reduction, let us first analyze the number of variables in  $\varphi^\blacktriangle$ : Observe that we have exactly  $\sum_{x \in V(H)} \sum_{v \in P_x} \deg_H(x)$  vertex variables and  $|E(G)|$  edge variables.

Therefore, we can bound  $n$  in terms of  $m$  and  $\ell$  as follows:

$$\begin{aligned} n &= |E(G)| + \sum_{x \in V(H)} \sum_{v \in P_x} \deg_H(x) \\ &\leq m^2 + \sum_{x \in V(H)} \sum_{v \in P_x} \ell \\ &= m^2 + \ell m \\ &\leq 2\ell m^2 \end{aligned}$$

It is easy to see that the construction of  $\varphi^\blacktriangle$  is achievable in time  $\mathcal{O}(n)$ . Consequently,  $\varphi^\blacktriangle$  can be computed in time  $f(\ell) \cdot m^{\mathcal{O}(1)}$  for some computable function  $f$ . Hence, all conditions for a parameterized reduction are satisfied. Since PARTITIONED SUBGRAPH ISOMORPHISM is  $\mathsf{W}[1]$ -hard parameterized by  $\ell$  according to Theorem 10, we conclude that  $\mathsf{WSAT}(\Delta_3 \cap B_2)$  is  $\mathsf{W}[1]$ -hard parameterized by  $k$ .

To show that  $\mathsf{WSAT}(\Delta_3 \cap B_2)$  cannot be solved in time  $f(k)n^{o(\frac{k}{\log k})}$ , we will assume the contrary and derive a contradiction. Suppose there exists an algorithm that solves  $\mathsf{WSAT}(\Delta_3 \cap B_2)$  in time  $f(k)n^{o(\frac{k}{\log k})}$ , where  $f$  is some computable function. Thus, we can solve an instance of PARTITIONED SUBGRAPH ISOMORPHISM by first reducing it to an instance of  $\mathsf{WSAT}(\Delta_3 \cap B_2)$  and then using this algorithm to solve the latter. From the previous analysis, we conclude that the reduction can be performed in time  $f'(\ell) \cdot m^2$ , where  $f'$  is some computable function. Thus, the overall running time of this routine is:

$$f'(\ell) \cdot m^2 + f(k) \cdot n^{o(\frac{k}{\log k})}.$$

We can upper bound the running time in terms of  $\ell$ ,  $m$ , and a new computable function  $f''$  as follows:

$$\begin{aligned} f'(\ell) \cdot m^2 + f(k) \cdot n^{o(\frac{k}{\log k})} &\leq f'(\ell) \cdot m^2 + f(3\ell) \cdot (2\ell m^2)^{o(\frac{3\ell}{\log 3\ell})} \\ &\leq \left( f'(\ell) + f(3\ell) \cdot (2\ell)^{o(\frac{3\ell}{\log 3\ell})} \right) \cdot m^{2 \cdot o(\frac{3\ell}{\log 3\ell})} \\ &\leq f''(\ell) \cdot m^{o(\frac{\ell}{\log \ell})} \end{aligned}$$

This implies that PARTITIONED SUBGRAPH ISOMORPHISM can be decided in time  $f''(\ell) \cdot m^{o(\frac{\ell}{\log \ell})}$ , where  $f''$  is some computable function. However, assuming the ETH, this contradicts Theorem 9, which states that PARTITIONED SUBGRAPH ISOMORPHISM cannot be solved in this time unless the ETH fails. Hence,  $\mathsf{WSAT}(\Delta_3 \cap B_2)$  cannot be decided in time  $f(k)n^{o(\frac{k}{\log k})}$ .  $\square$

### 5.2.2 1-1-Bounded 3-Normalized Formulas

In Construction 6, each vertex variable in the formula  $\varphi^\blacktriangle$  occurs once in negated form and once in unnegated form, whereas each edge variable appears twice in unnegated form.

We can further refine the structure of  $\varphi^\blacktriangle$  to ensure that every variable appears at most once with each polarity (negated or unnegated). This further reduces the size of switches in the reductions presented in Chapters 6 and 8.

**Construction 7.** Given an instance  $(G, H, \mathcal{P})$  of PARTITIONED SUBGRAPH ISOMORPHISM, we define a propositional formula  $\varphi^\bullet$  based on Construction 6, where for each edge  $uv \in E(G)$ , we introduce two additional *arc variables*  $\vec{u}_v$  and  $\vec{v}_u$ , alongside the existing edge variable  $uv$ . These are called arc variables because, unlike edges, the order of the vertices matters, in a way analogous to arcs. Formally, we define the variable set of  $\varphi^\bullet$  as follows:

$$\mathcal{V}(\varphi^\bullet) := \{v_y \mid x \in V(H), v \in P_x, y \in N_H(x)\} \cup \{\vec{u}_v, \vec{v}_u, uv \mid uv \in E(G)\}.$$

Then, the propositional formula  $\varphi^\bullet$  is defined as follows:

$$\varphi := \bigwedge_{x \in V(H)} \left( \bigvee_{v \in P_x} \left( \bigwedge_{y \in N_H(x)} v_y \right) \right) \quad (5.2)$$

$$\wedge \bigwedge_{\substack{x \in V(H) \\ y \in N_H(x) \\ v \in P_x}} \left( v_y \Rightarrow \bigvee_{u \in N_G(v) \cap P_y} \vec{v}_u \right) \quad (5.3)$$

$$\wedge \bigwedge_{uv \in E(G)} ((\vec{v}_u \vee \vec{u}_v) \Rightarrow uv) \quad (5.4)$$

Observe that, compared to the formula  $\varphi^\blacktriangle$  from Construction 6, each edge variable in the second part of  $\varphi^\bullet$  is replaced by a corresponding arc variable. As a result, every arc variable appears exactly once in unnegated form. The new third part of  $\varphi^\bullet$  ensures that, for each edge  $uv \in E(G)$ , if one of the arc variables  $\vec{u}_v$  or  $\vec{v}_u$  is set to true, then the corresponding edge variable  $uv$  must also be set to true. Intuitively, these modifications guarantee the same edge selection behavior as before, however, it increases the weight of a satisfying truth assignment. Specifically, for each selected edge, we additionally have the two corresponding arc variables set to true. Accordingly, we update the maximum allowed weight of a satisfying truth assignment to:

$$k^\bullet := 5|E(H)|.$$

For each edge  $uv \in E(G)$ , the implication in the third part of  $\varphi^\bullet$  can be rewritten as follows:

$$(\vec{v}_u \vee \vec{u}_v) \Rightarrow uv \quad \Leftrightarrow \quad \neg(\vec{v}_u \vee \vec{u}_v) \vee uv \quad \Leftrightarrow \quad (\neg\vec{v}_u \wedge \neg\vec{u}_v) \vee uv$$

Thus, each arc variable appears once in negated form, and each edge variable appears once in unnegated form in the third part of  $\varphi^\bullet$ . Moreover, the formula  $\varphi^\bullet$  remains 3-normalized. This leads us to the following lemma:

**Lemma 8.**  $(G, H, \mathcal{P})$  is a yes-instance of PARTITIONED SUBGRAPH ISOMORPHISM if and only if  $(\varphi^\bullet, k^\bullet)$  is a yes-instance of WSAT( $\Delta_3 \cap B_{1,1}$ ).

*Proof.* To prove this lemma, we show that  $(\varphi^\blacktriangle, k^\blacktriangle)$  from Construction 6 admits a satisfying truth assignment  $I^\blacktriangle$  with weight  $|I^\blacktriangle|_1 \leq k^\blacktriangle$  if and only if  $(\varphi^\bullet, k^\bullet)$  admits a satisfying truth assignment  $I^\bullet$  with  $|I^\bullet|_1 \leq k^\bullet$ . Due to Lemma 7, the desired result then follows. We prove both directions separately.

( $\Rightarrow$ ) Suppose there exists a truth assignment  $I^\blacktriangle$  of  $\varphi^\blacktriangle$  such that  $I^\blacktriangle(\varphi^\blacktriangle) = 1$  and  $|I^\blacktriangle|_1 \leq k^\blacktriangle$ . We define a truth assignment  $I^\bullet$  for  $\varphi^\bullet$  that agrees with  $\varphi^\blacktriangle$  on all common variables. Additionally, for each edge  $uv \in E(G)$ , we set  $I^\bullet(\vec{u}_v) = I^\bullet(\vec{v}_u) = I^\blacktriangle(uv)$  for the arc variables. It is easy to see that  $I^\bullet$  satisfies  $\varphi^\bullet$ . Furthermore, since we have established in Lemma 7 that exactly  $|E(H)|$  edge variables are set to true in  $I^\blacktriangle$ , it follows that  $2|E(H)|$  arc variables must be set to true in  $I^\bullet$ . Consequently, we have  $|I^\bullet|_1 = |I^\blacktriangle|_1 + 2|E(H)| \leq k^\blacktriangle + 2|E(H)| = 5|E(H)| \leq k^\bullet$ . As a result,  $I^\bullet$  is a yes-instance of WSAT( $\Delta_3 \cap B_{1,1}$ ).

( $\Leftarrow$ ) Suppose there exists a truth assignment  $I^\bullet$  such that  $I^\bullet(\varphi^\bullet) = 1$  and  $|I^\bullet|_1 \leq k^\bullet$ . We begin by analyzing the structure of  $I^\bullet$ .

Just as established in Lemma 7 for  $\varphi^\blacktriangle$ , at least  $2|E(H)|$  vertex variables must be set to true in  $I^\bullet$ . Therefore, for  $2|E(H)|$  implications in the second part of  $\varphi^\bullet$ , the left side evaluates to true. In order for these implications to be satisfied, the right side must evaluate to true as well, and since the right side is a disjunction of arc variables, and each arc variable appears in exactly one of those implications, at least  $2|E(H)|$  arc variables must be set to true. It follows that the left sides of at least  $|E(H)|$  implications in the third part of  $\varphi^\bullet$  are satisfied. To satisfy these implications, the right side must evaluate to true, entailing that a minimum of  $|E(H)|$  edge variables must be set to true in  $I^\bullet$ .

This amounts to a minimum of  $5|E(H)|$  variables set to true in  $I^\bullet$ . However, we know that  $|I^\bullet|_1 \leq k^\bullet = 5|E(H)|$ , therefore, precisely  $2|E(H)|$  vertex variables,  $2|E(H)|$  arc variables, and  $|E(H)|$  edge variables are set to true. Consequently, for each edge variable  $uv \in E(G)$  set to true, both corresponding arc variables  $\vec{u}_v$  and  $\vec{v}_u$  must also be set to true.

We now define a truth assignment  $I^\blacktriangle$  for  $\varphi^\blacktriangle$  that is equivalent to  $I^\bullet$  restricted to the variables in  $\varphi^\blacktriangle$ . It is easy to see that  $I^\blacktriangle$  satisfies  $\varphi^\blacktriangle$ . Furthermore, exactly  $3|E(G)|$  variables are set to true in  $I^\blacktriangle$ , which is less than or equal to  $k^\blacktriangle$ . Thus,  $I^\blacktriangle$  is a yes-instance of WSAT( $\Delta_3 \cap B_2$ ).  $\square$

From Lemma 8, we obtain a parameterized reduction from PARTITIONED SUBGRAPH ISOMORPHISM to WSAT( $\Delta_3 \cap B_{1,1}$ ), analogously to Theorem 11, resulting in W[1]-hardness of WSAT( $\Delta_3 \cap B_{1,1}$ ). Moreover, since the size of the parameter, the number of variables, and consequently the time needed to construct formula  $\varphi^\bullet$  only increases by a constant factor, we can use the same arguments as in Theorem 11 to show that

WSAT( $\Delta_3 \cap B_{1,1}$ ) cannot be decided in subexponential time unless the ETH fails. Thus, we directly obtain the following theorem:

**Theorem 12.** WSAT( $\Delta_3 \cap B_{1,1}$ ) is  $W[1]$ -hard parameterized by  $k$ , and, assuming the ETH, cannot be solved in time  $f(k)n^{o(\frac{k}{\log k})}$ , where  $f$  is some computable function,  $k$  is the maximum weight of a satisfying truth assignment, and  $n$  is the number of variables in the formula.

### 5.2.3 2-1-Bounded 2-Normalized Formulas

As previously hinted, to reduce from weighted satisfiability to WEIGHTED-GCON-SWITCHING, we require that the formulas are 2-normalized. Observe that the original formula  $\varphi$  in Construction 5 is 2-normalized, but the two refinements presented in Construction 6 and Construction 7 are not. In the following, we will construct a propositional formula  $\varphi^*$  that is 2-normalized and ensures that each variable occurs at most three times.

**Construction 8.** Given an instance  $(G, H, \mathcal{P})$  of PARTITIONED SUBGRAPH ISOMORPHISM, we split each vertex variable  $v \in V(G)$  into multiple variables similarly to the modifications presented in Construction 6. Specifically, we define a propositional formula  $\varphi^*$  with variable set:

$$\mathcal{V}(\varphi^*) := \{v_i \mid x \in V(H), v \in P_x, i \in [0, \deg_H(x)]\} \cup \{uv \mid uv \in E(G)\}.$$

These new variables will again be referred to as *vertex variables*. For each  $x \in V(H)$ , let  $g_x : N_H(x) \rightarrow [\deg_H(x)]$  be a bijection that maps each neighbor  $y$  of  $x$  to a unique index  $i \in [\deg_H(x)]$ . We can now define the propositional formula  $\varphi^*$  as follows:

$$\varphi^* := \bigwedge_{x \in V(H)} \left( \bigvee_{v \in P_x} v_0 \right) \quad (5.5)$$

$$\wedge \bigwedge_{\substack{x \in V(H) \\ y \in N_H(x) \\ v \in P_x}} \left( v_{g_x(y)} \Rightarrow \bigvee_{u \in N_G(v) \cap P_y} uv \right) \quad (5.6)$$

$$\wedge \bigwedge_{\substack{x \in V(H) \\ v \in P_x \\ i \in [\deg_H(x)]}} (v_{i-1} \Rightarrow v_i) \quad (5.7)$$

The first and second part of  $\varphi^*$  are identical to  $\varphi$  from Construction 5, except that each occurrence of vertex variable  $v$  is replaced by one of its unique copies. As a result, each vertex variable appears exactly once in the first and second part of  $\varphi^*$ . In the third part, the chain of implications for each vertex  $v \in V(G)$  ensure that, if  $v$  is selected from its partition set in the first part, then all its  $\deg_H(x) + 1$  corresponding vertex variables

must be set to true. Intuitively, this guarantees the same behavior as the original formula, but it increases the weight of a satisfying truth assignment. To select one vertex from each partition set in  $\mathcal{P}$ , we obtain a total of  $\sum_{x \in V(H)} (\deg_H(x) + 1) = |V(H)| + 2|E(H)|$  vertex variables set to true. The number of edge variables set to true remains unchanged. Accordingly, we update the maximum allowed weight of a satisfying truth assignment to:

$$k^* := |V(H)| + 3|E(H)|.$$

Each disjunction in the third part of  $\varphi^*$  translates to a disjunction where the variable on the left side is negated while the variable on the right side is not. Consequently, each vertex variable appears in each polarity exactly once in this part. In conclusion, the formula remains 2-normalized and each variable appears at most two times in one polarity, and at most once in the other polarity. This leads us to the following lemma:

**Lemma 9.**  *$(G, H, \mathcal{P})$  is a yes-instance of PARTITIONED SUBGRAPH ISOMORPHISM if and only if  $(\varphi^*, k^*)$  is a yes-instance of WSAT( $\Delta_2 \cap B_{2,1}$ ).*

*Proof.* To prove this lemma, we show that  $(\varphi, k)$  from Construction 5 admits a satisfying truth assignment  $I$  with weight  $|I|_1 \leq k$  if and only if  $(\varphi^*, k^*)$  admits a satisfying truth assignment  $I^*$  with  $|I^*|_1 \leq k^*$ . Due to Lemma 6, the desired result then follows. We prove both directions separately.

( $\Rightarrow$ ) Suppose there exists a truth assignment  $I$  of  $\varphi$  such that  $I(\varphi) = 1$  and  $|I|_1 \leq k$ . We construct a corresponding truth assignment  $I^*$  of  $\varphi^*$  as follows: for each  $x \in V(H)$ , every  $v \in P_x$ , and every  $i \in [0, \deg_H(x)]$ , we set  $I^*(v_i) = I(v)$ . For every edge variable  $uv \in E(G)$ , we set  $I^*(uv) = I(uv)$ . It is easy to see that the conjuncts in the first and second part of  $\varphi^*$  are satisfied as they are identical to  $\varphi$  with only the vertex variables replaced by their copies. Furthermore, since for each  $x \in V(H)$ , every  $v \in P_x$ , and every  $i \in [\deg_H(x)]$ , we have that  $I(v_i) = 1$  if  $I(v_{i-1}) = 1$ , every conjunct in the third part of  $\varphi^*$  is satisfied as well. Thus,  $I^*(\varphi^*) = 1$ .

To determine the number of variables set to true in  $I^*$ , recall from the proof of Lemma 6 that for each vertex  $x \in P_x$ , exactly one vertex variable  $v \in P_x$  is set to true in  $I$ . For each such vertex variables  $v$  of  $\varphi$ , we have  $\deg_H(x) + 1$  vertex variables set to true in  $I^*$ . This amounts to an additional  $\sum_{x \in V(H)} \deg_H(x) = 2|E(H)|$  variables set to true in  $I^*$ , resulting in a total of  $|I^*|_1 = |I|_1 + 2|E(H)| = |V(G)| + 3|E(H)|$  variables set to true, which is less than or equal to  $k^*$ . Thus,  $I^*$  is a yes-instance of WSAT( $\Delta_2 \cap B_{2,1}$ ).

( $\Leftarrow$ ) Suppose there exists a truth assignment  $I^*$  such that  $I^*(\varphi^*) = 1$  and  $|I^*|_1 \leq k^*$ . We begin by analyzing the structure of  $I^*$ .

For each conjunct in the first part of  $\varphi^*$  to be satisfied, it must hold that for each  $x \in V(H)$ , there exists at least one vertex  $v \in P_x$  where  $v_0$  is set to true. Furthermore, for each vertex  $v \in V(G)$ , we have a chain of implications  $v_0 \Rightarrow v_1, v_1 \Rightarrow v_2, \dots, v_{\deg_H(x)-1} \Rightarrow v_{\deg_H(x)}$  in the third part of  $\varphi^*$ , including all  $\deg_H(x) + 1$  vertex variables corresponding to  $v$ . Given that  $v_0$  is set to true, all these variables must evaluate to true

for all these implications to be satisfied. This amounts to a minimum of  $|V(H)|$  vertex variables with index 0 set to true and a minimum of  $\sum_{x \in V(H)} \deg_H(x) = 2|E(H)|$  vertex variables with an index greater than 0 set to true. Each variable in the latter forms the left side of an implication in the second part of  $\varphi^*$ . To satisfy these implications, the right side must evaluate to true as well, and since the right side is a disjunction of edge variables, and each edge variable appears in exactly two implications of the second part, at least  $|E(H)|$  edge variables must be set to true.

This amounts to a minimum of  $|V(H)| + 3|E(H)|$  variables set to true in  $I^*$ . However, we know that  $|I^*|_1 \leq k^* = |V(H)| + 3|E(H)|$ , therefore, precisely  $|V(H)|$  vertex variables with index 0, and exactly  $|E(H)|$  edge variables to true.

We now construct a truth assignment  $I$  for  $\varphi$  such that  $I(v) := I^*(v_0)$  for every  $v \in V(G)$ , and  $I(uv) := I^*(uv)$  for every edge variable  $uv \in E(G)$ . It is easy to see that  $I$  satisfies  $\varphi$ . Furthermore, exactly  $|V(H)| + |E(H)|$  variables are set to true in  $I$ , which is less than or equal to  $k$ . Thus,  $I$  is a yes-instance of  $\text{WSAT}(\Delta_2 \cap B_{2,1})$ .  $\square$

Lemma 9 demonstrates how to reduce an instance of  $\text{PARTITIONED SUBGRAPH ISOMORPHISM}$  to an instance of  $\text{WSAT}(\Delta_2 \cap B_{2,1})$ . However, to obtain a parameterized reduction like in Theorem 11, we encounter a small problem:  $k^*$  is not strictly bounded by a function of  $|E(H)|$  due to its additional dependence on  $|V(H)|$ . To resolve this, we can utilize a kernelization procedure that removes every vertex  $x \in V(H)$  with  $\deg_H(x) = 0$  from  $H$ , as well as all vertices in  $P_x$  from  $G$ . This is possible, since  $P_x \neq \emptyset$  by definition, hence  $x$  can be mapped to some arbitrary vertex in  $P_x$ . Applying this kernelization procedure to an arbitrary instance, we can ensure that each vertex  $x \in V(H)$  has at least one incident edge, thus  $|V(H)| = \sum_{v \in V(H)} 1 \leq \sum_{v \in V(H)} \deg_H(v) = 2|E(H)|$ . It follows that  $k^*$  is indeed bounded by a function of  $|E(H)|$ , and analogously to Theorem 11,  $\text{W}[1]$ -hardness of  $\text{WSAT}(\Delta_2 \cap B_{2,1})$  follows. Moreover, since the size of the parameter, the number of variables, and consequently the time needed to construct formula  $\varphi^*$  only increases by a constant factor, we can use the same arguments as in Theorem 11 to show that  $\text{WSAT}(\Delta_2 \cap B_{2,1})$  cannot be decided in subexponential time unless the ETH fails. Consequently, we obtain the following theorem:

**Theorem 13.**  *$\text{WSAT}(\Delta_2 \cap B_{2,1})$  is  $\text{W}[1]$ -hard parameterized by  $k$ , and, assuming the ETH, cannot be solved in time  $f(k)n^{o(\frac{k}{\log k})}$ , where  $f$  is some computable function,  $k$  is the maximum weight of a satisfying truth assignment, and  $n$  is the number of variables in the formula.*

**Discussion** In summary, this chapter has established the computational hardness of both unweighted and weighted satisfiability problems under various structural restrictions on propositional formulas. We showed that unweighted satisfiability remains NP-hard even for planar 3-CNF formulas with at most three occurrences per variable, and extended this to 3-normalized formulas with only two occurrences per variable. For the weighted

case, we demonstrated  $W[1]$ -hardness for several classes of normalized formulas with bounded variable occurrences, including 3-normalized 1-1-bounded and 2-normalized 2-1-bounded formulas, with the former resolving an open question posed by Kanj and Szeider [10]. These results provide a solid foundation for reductions to switching problems on constrained graph families in subsequent chapters.



# $s$ - $t$ -Connectivity Switching and Satisfiability

In Chapter 4, we established that for every class  $\mathcal{A}$  of graphs closed under isomorphism and excluding isolated vertices, STCON-SWITCHING restricted to  $\{1\}^*$ - $\mathcal{A}$ -switches can be decided in linear time if  $\mathcal{A}$  contains only complete multipartite graphs; otherwise, the problem is NP-hard. This immediately implies that STCON-SWITCHING is NP-hard for  $\{1, 1\}$ -switches but becomes tractable for  $\{1\}^*$ -star switches. We also showed that STCON-SWITCHING remains NP-hard for  $\{2, 1\}$ -star switches (Theorem 4).

In this chapter, we extend these results through reductions from various restricted variants of the satisfiability problem introduced in Chapter 5. These restrictions impose additional structural constraints on the switch graph, particularly concerning the structure of the union graph. Moreover, by reducing from the weighted satisfiability problem, we derive intractability results for WEIGHTED-STCON-SWITCHING under similar constraints.

The reductions involve constructing a special type of switch graph, which we call *series-parallel switch graph*. Series-parallel graphs are well-studied in standard graph theory, and their definition naturally extends to switch graphs. A switch graph  $\hat{G}$  is called *series-parallel* if it has distinguished terminals  $s(\hat{G})$  and  $t(\hat{G})$ , and satisfies one of the following conditions:

- *Single-edge switch graph*:  $\hat{G}$  consists of exactly two vertices, a distinct source  $s(\hat{G})$  and a distinct target  $t(\hat{G})$ , and one switch whose edge set includes an edge connecting  $s(\hat{G})$  and  $t(\hat{G})$ .
- *Series-composition*:  $\hat{G}$  is obtained by taking the union of two series-parallel graphs  $\hat{G}_1$  and  $\hat{G}_2$  identifying  $s(\hat{G}) = s(\hat{G}_1)$ ,  $t(\hat{G}_1) = s(\hat{G}_2)$ , and  $t(\hat{G}_2) = t(\hat{G})$ .
- *Parallel-composition*:  $\hat{G}$  is obtained by taking the union of two series-parallel graphs  $\hat{G}_1$  and  $\hat{G}_2$  identifying  $s(\hat{G}) = s(\hat{G}_1) = s(\hat{G}_2)$  and  $t(\hat{G}) = t(\hat{G}_1) = t(\hat{G}_2)$ .

Observe that the union graph of a series-parallel switch graph is a series-parallel graph without multiedges<sup>1</sup>. As a result, the union graph of the constructed switch graph is inherently constrained. In particular, series-parallel graphs are naturally planar and have a treewidth of at most two.

In Section 6.1, we describe the construction that transforms an arbitrary propositional formula  $\varphi$  into a series-parallel switch graph  $\widehat{G}_\varphi$ . Using this construction, we establish in Sections 6.2 and 6.3 that STCON-SWITCHING and ONOFF-STCON-SWITCHING, parameterized by the activation budget, are intractable for  $\{2, 1\}$ -switches even when the union graph is a path, and for  $\{1, 1\}$ -switches even when the union graph is series-parallel. For ONOFF-STCON-SWITCHING, we further present a method to transform  $\{1, 1\}$ -switches into  $\{2, 1\}$ -star switches, thereby extending the intractability results to this class of switches as well.

## 6.1 Reduction from Satisfiability

Given a propositional formula  $\varphi$  in negation normal form, we aim to construct a series-parallel on-off switch graph  $\widehat{G}_\varphi$  such that there exists an  $s$ - $t$ -path in  $\widehat{G}_\varphi$  if and only if  $\varphi$  is satisfiable. The series-parallel composition of  $\widehat{G}_\varphi$  is defined by an inductive process based on the structure of  $\varphi$ . In this composition, each literal is modeled by a single-edge switch graph, with the corresponding edge part of an on-off-switch representing the respective variable. The edge appears in the off-position if the variable is negated and in the on-position otherwise. If we define a configuration such that this switch is in the on-position precisely when the corresponding variable is set to true, then the edge is present in the configured switch graph if and only if the literal evaluates to true.

Intuitively, a series-composition within  $\widehat{G}_\varphi$  can model a conjunction, as any  $s$ - $t$ -path in a series-composition of two switch graphs must traverse both of them. Conversely, a parallel-composition within  $\widehat{G}_\varphi$  can model a disjunction, since any  $s$ - $t$ -path in a parallel-composition of two switch graphs must traverse only one of them. Thus, the series-parallel structure of the switch graph  $\widehat{G}_\varphi$  is a direct representation of the logical structure of the formula  $\varphi$ . Fig. 6.1 illustrates the construction on a simple example. Formally, the inductive construction process is defined as follows:

**Construction 9.** For every formula  $\varphi$  in negation normal form, we construct a corresponding series-parallel on-off switch graph  $\widehat{G}_\varphi$  as follows:

- If  $\varphi := \neg x$  for some variable  $x$ , then  $\widehat{G}_\varphi$  is a single-edge switch graph with a single switch  $S_x$ , where the edge is contained in the off-position  $S_x[1]$ .
- If  $\varphi := x$  for some variable  $x$ , then  $\widehat{G}_\varphi$  is a single-edge switch graph with a single switch  $S_x$ , where the edge is contained in the on-position  $S_x[2]$ .
- If  $\varphi := \psi \wedge \omega$ , then  $\widehat{G}_\varphi$  is formed by the series-composition of  $\widehat{G}_\psi$  and  $\widehat{G}_\omega$ .

<sup>1</sup>In the union graph, only one copy of an parallel edge is used.

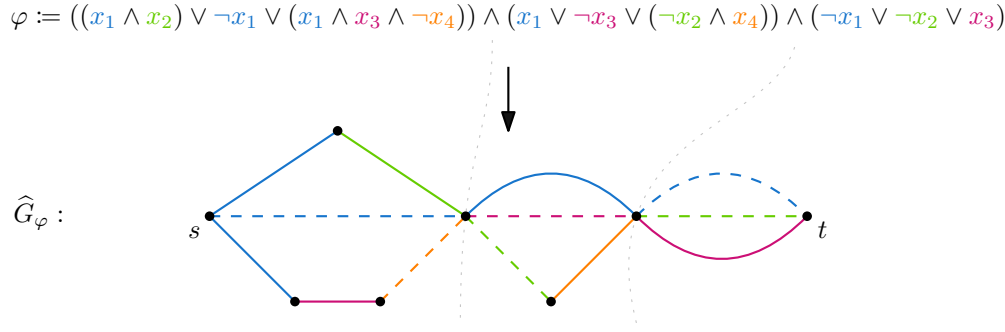


Figure 6.1: Example switch graph  $\widehat{G}_\varphi$  for a propositional formula  $\varphi$  in negation normal form, constructed as described in Construction 9. Each color corresponds to a variable and its associated switch; solid edges represent edges of the on-position, while dashed edges represent edges of the off-position.

- If  $\varphi := \psi \vee \omega$ , then  $\widehat{G}_\varphi$  is formed by the parallel-composition of  $\widehat{G}_\psi$  and  $\widehat{G}_\omega$ .

Throughout this process, switches corresponding to the same variable retain their identity and are merged accordingly. As a result,  $\widehat{G}_\varphi$  contains exactly one switch  $S_x$  for each variable  $x \in \mathcal{V}(\varphi)$ .

As mentioned previously, we can map each interpretation  $I$  of  $\varphi$  to a configuration  $c$  of  $\widehat{G}_\varphi$  by setting each switch  $S_x \in \mathcal{S}(\widehat{G}_\varphi)$  to the on-position if and only if the corresponding variable  $x$  is set to true in  $I$ . This correspondence is captured by the bijection  $f_\varphi : \mathcal{I}(\varphi) \rightarrow \mathcal{C}(\widehat{G}_\varphi)$ , formally defined for each interpretation  $I \in \mathcal{I}(\varphi)$  and switch  $S_x \in \mathcal{S}(\widehat{G}_\varphi)$  as follows:

$$f_\varphi(I)(S_x) := I(x) + 1.$$

The following lemma proves the correctness of this mapping:

**Lemma 10.** *Let  $\varphi$  be a propositional formula in negation normal form and  $I$  an interpretation of  $\varphi$ . Then,  $\widehat{G}_\varphi$  configured by  $f_\varphi(I)$  admits an  $s$ - $t$ -path if and only if  $I(\varphi) = 1$ .*

*Proof.* Let  $c := f_\varphi(I)$ . The proof is by induction on the structure of the formula  $\varphi$ .

**Base case:** If  $\varphi$  is a literal, then either  $\varphi := x$  or  $\varphi := \neg x$  for some variable  $x \in \mathcal{V}(\varphi)$ . This entails that  $\widehat{G}_\varphi$  is a single-edge switch graph with a single on-off switch  $S_x$ , where an edge  $e$  connecting the source  $s(\widehat{G}_\varphi)$  and the target  $t(\widehat{G}_\varphi)$  is either in the off- or on-position. We now distinguish between the two shapes of the literal:

- $\varphi := \neg x$ : By construction, edge  $e \in S_x[1]$ . Consequently,  $\widehat{G}_\varphi \circ c$  admits an  $s$ - $t$ -path if and only if  $c(S_x) = 1$ . As  $c(S_x) = f(I)(S_x) = I(x) + 1 = (1 - I(\neg x)) + 1 = I(\varphi)$ ,  $c(S_x) = 1$  if and only if  $I(\varphi) = 1$ .

- $\varphi := x$ : By construction, edge  $e \in S_x[2]$ . Consequently,  $\widehat{G}_\varphi \circ c$  admits an  $s$ - $t$ -path if and only if  $c(S_x) = 2$ . As  $c(S_x) = f(I)(S_x) = I(x) + 1 = I(\varphi) + 1$ ,  $c(S_x) = 2$  if and only if  $I(\varphi) = 1$ .

**Inductive step:** Assume that the statement holds for all subformulas of  $\varphi$ . Since  $\varphi$  is not a literal, it must be either be a conjunction or a disjunction of two subformulas  $\psi$  and  $\omega$ . This entails that  $\widehat{G}_\varphi$  is formed by either a series-composition or parallel-composition of  $\widehat{G}_\psi$  and  $\widehat{G}_\omega$ . We now distinguish between the two possible shapes of  $\varphi$ :

- $\varphi = \psi \wedge \omega$ : By construction,  $\widehat{G}_\varphi$  is formed by a series-composition of  $\widehat{G}_\psi$  and  $\widehat{G}_\omega$ . Observe that  $\widehat{G}_\varphi \circ c$  admits an  $s$ - $t$ -path if and only if both  $\widehat{G}_\psi \circ c$  and  $\widehat{G}_\omega \circ c$  admit an  $s$ - $t$ -path. By the inductive hypothesis, this is precisely the case when  $I(\psi) = 1$  and  $I(\omega) = 1$ . Finally,  $I(\psi) = 1$  and  $I(\omega) = 1$  if and only if  $I(\varphi) = I(\psi \wedge \omega) = 1$ .
- $\varphi = \psi \vee \omega$ : By construction,  $\widehat{G}_\varphi$  is formed by a parallel-composition of  $\widehat{G}_\psi$  and  $\widehat{G}_\omega$ . Observe that  $\widehat{G}_\varphi \circ c$  admits an  $s$ - $t$ -path if and only if  $\widehat{G}_\psi \circ c$  or  $\widehat{G}_\omega \circ c$  admit an  $s$ - $t$ -path. By the inductive hypothesis, this is precisely the case when  $I(\psi) = 1$  or  $I(\omega) = 1$ . Finally,  $I(\psi) = 1$  or  $I(\omega) = 1$  if and only if  $I(\varphi) = I(\psi \vee \omega) = 1$ .  $\square$

As a result of Lemma [10](#), we derive following corollary, from which the correctness of the above-described reduction from the satisfiability problem to STCON-SWITCHING follows:

**Corollary 2.** *Let  $\varphi$  be a propositional formula in negation normal form. Then  $\varphi$  is satisfiable if and only if there exists a configuration of  $\widehat{G}_\varphi$  such that the configured switch graph admits an  $s$ - $t$ -path.*

Concerning the weighted variant of the switching problem, consider an arbitrary propositional formula  $\varphi$  in negation normal form and an interpretation  $I \in \mathcal{I}(\varphi)$ . Observe that the number of variables assigned to true in  $I$  match the number of switches in the on-position in configuration  $f_\varphi(I)$ , i.e.,  $|I|_1 = |f_\varphi(I)|_2$ . With this in mind, we derive the following corollary, from which the correctness of the above described reduction from the weighted satisfiability problem to ONOFF-STCON-SWITCHING follows:

**Corollary 3.** *Let  $\varphi$  be a propositional formula in negation normal form and  $k$  some integer. Then there exists an interpretation  $I \in \mathcal{I}(\varphi)$  such that  $I(\varphi) = 1$  and  $|I|_1 \leq k$  if and only if there exists a configuration  $c$  of  $\widehat{G}_\varphi$  such that  $\widehat{G}_\varphi \circ c$  admits an  $s$ - $t$ -path and  $|c|_2 \leq k$ .*

## 6.2 Unweighted Switching

By selecting appropriate variants of the satisfiability problem from Chapter [5](#) and applying the reduction described in Section [6.1](#), we obtain a variety of intractability results for

STCON-SWITCHING under different structural restrictions on the switch graph. Given a propositional formula  $\varphi$  in negation normal form, along with additional syntactical constraints, we now analyze how these conditions translate into structural properties of the resulting switch graph  $\widehat{G}_\varphi$ .

Firstly, as previously discussed, the union graph of  $\widehat{G}_\varphi$  is a series-parallel graph, which is inherently planar and has a treewidth of at most two. In the special case where  $\varphi$  is 2-normalized, the switch graph  $\widehat{G}_\varphi$  is constructed as a series-composition of parallel-compositions of single-edge switch graphs. As the union graph  $G(\widehat{G}_\varphi)$  contains no multiedges, it follows that  $G(\widehat{G}_\varphi)$  forms a simple path.

Secondly, consider an arbitrary variable  $x$  that appears  $a$  times negated and  $b$  times unnegated in  $\varphi$ . By the construction of  $\widehat{G}_\varphi$ , the switch  $S_x$  contains  $a$  edges in the off-position and  $b$  edges in the on-position. Hence, if  $\varphi$  is  $a$ - $b$ -bounded for some  $a, b \in \mathbb{N}_{\geq 0}$ , then each switch in  $\widehat{G}_\varphi$  is an  $\{a, b\}$ -switch.

As established in Theorem 7, the satisfiability problem remains NP-hard even for 2-1-bounded 2-normalized formulas. By reducing from this restricted variant of the satisfiability problem to STCON-SWITCHING using Corollary 2, and applying the structural observations above, we derive the following intractability result under the corresponding switch graph constraints:

**Theorem 14.** *STCON-SWITCHING is NP-hard, even if all switches are  $\{2, 1\}$ -switches and the union graph is a path.*

On the other hand, Theorem 8 shows that the satisfiability problem remains NP-hard even if the formula is 3-normalized and 1-1-bounded. By using Corollary 2 to obtain a reduction to STCON-SWITCHING, we derive the following intractability result, extending Theorem 3 by adding additional restrictions to the union graph:

**Theorem 15.** *STCON-SWITCHING is NP-hard, even if all switches are  $\{1, 1\}$ -switches and the union graph is a series-parallel graph.*

## 6.3 Weighted Switching

We now turn to the weighted switching problem. In particular, we analyze the complexity of ONOFF-STCON-SWITCHING, parameterized by the activation budget, by employing the same reduction as for the unweighted variant in Section 6.1. Notice that the restrictions imposed on the propositional formulas in the previous section result in identical structural constraints on the switch graph  $\widehat{G}_\varphi$  as before.

As established in Chapter 2, WSAT( $\Phi$ ) is W[SAT]-hard when parameterized by the maximum weight of the truth assignment. Since any propositional formula can be converted into an equisatisfiable formula in negation normal form in linear time, we can reduce WSAT( $\Phi$ ) to ONOFF-STCON-SWITCHING via Corollary 3, yielding the following intractability result:

**Theorem 16.**  $\text{ONOFF-STCON-SWITCHING}$  is  $W[\text{SAT}]$ -hard parameterized by the activation budget, even if the union graph is a series-parallel graph.

To limit the size of switches, we turn to Theorem 13, which establishes that weighted satisfiability is  $W[1]$ -hard parameterized by the maximum weight  $k$  of a truth assignment, even for 2-1-bounded 2-normalized formulas. Moreover, the theorem shows that, unless the ETH fails, this problem cannot be solved in time  $g(k)n^{o(\frac{k}{\log k})}$ , where  $g$  is some computable function and  $n$  is the number of variables in the formula.

In the parameterized reduction to  $\text{WEIGHTED-STCON-SWITCHING}$  described in Corollary 3, the parameter  $k$  is preserved, and  $n$  corresponds to the number of switches in  $\widehat{G}_\varphi$ . Since the size of the formula is in  $\mathcal{O}(n)$ , the construction of  $\widehat{G}_\varphi$  can be carried out in  $\mathcal{O}(n)$  time. Therefore, if  $\text{WEIGHTED-STCON-SWITCHING}$  could be decided in time  $f(k)n^{o(\frac{k}{\log k})}$  for some computable function  $f$ , it would imply that  $\text{WSAT}(\Delta_2 \cap B_{2,1})$  admits an algorithm that contradicts the ETH.

Furthermore, observe that for each formula  $\varphi \in \Delta_2 \cap B_{2,1}$ , the union graph of  $\widehat{G}_\varphi$  is a path and each switch is a  $\{2, 1\}$ -switch. Hence, we obtain the following intractability result for  $\text{ONOFF-STCON-SWITCHING}$ :

**Theorem 17.**  $\text{ONOFF-STCON-SWITCHING}$  is  $W[1]$ -hard parameterized by the activation budget  $k$ , even if all switches are  $\{2, 1\}$ -switches and the union graph is a path. Furthermore, assuming the ETH, the problem cannot be solved in time  $f(k)n^{o(\frac{k}{\log k})}$ , where  $f$  is an arbitrary computable function and  $n$  is the number of switches in the switch graph.

Similarly, applying the reduction from Corollary 3 to the instances used in Theorem 12 — which establishes the  $W[1]$ -hardness parameterized by the maximum weight of a truth assignment and rules out subexponential-time algorithms (under the ETH) for weighted satisfiability of 1-1-bounded 3-normalized formulas — yields the following intractability result for  $\text{ONOFF-STCON-SWITCHING}$ :

**Theorem 18.**  $\text{ONOFF-STCON-SWITCHING}$  is  $W[1]$ -hard parameterized by the activation budget  $k$ , even if all switches are  $\{1, 1\}$ -switches and the union graph is a series-parallel graph. Furthermore, assuming the ETH, the problem cannot be solved in time  $f(k)n^{o(\frac{k}{\log k})}$ , where  $f$  is an arbitrary computable function and  $n$  is the number of switches in the switch graph.

### 6.3.1 Intractability for Star Switches

In Chapter 4, we established that  $\text{STCON-SWITCHING}$  is NP-hard even when restricted to  $\{2, 1\}$ -star switches (Theorem 4) via a reduction from  $\text{TRANSITION-COMPATIBLE-}s$ - $t$ - $\text{PATH}$ . In this section, we aim to extend this result to the weighted variant. Specifically, we show that  $\text{ONOFF-STCON-SWITCHING}$  is  $W[1]$ -hard even when all switches are  $\{2, 1\}$ -star-switches. To achieve this, we present a transformation from  $\{1, 1\}$ -switches to

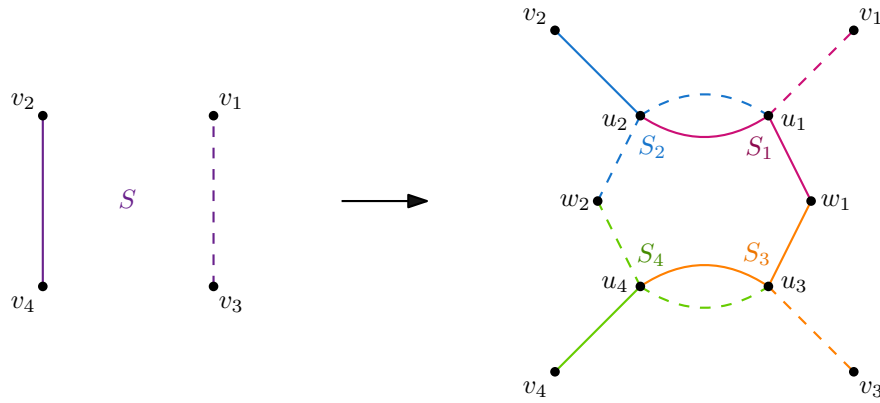


Figure 6.2: Example of the transformation from a  $\{1, 1\}$ -switch to four  $\{2, 1\}$ -star switches according to Construction 10. Each switch is represented by a different color, with solid edges indicating the on-position and dashed edges indicating the off-position.

$\{2, 1\}$ -star-switches, without affecting the existence of a configuration that admits an  $s$ - $t$ -path. Since  $\text{STCON-SWITCHING}$  is already  $W[1]$ -hard parameterized by the activation budget when restricted to  $\{1, 1\}$ -switches (Theorem 15), this transformation yields the same  $W[1]$ -hardness when restricted to  $\{2, 1\}$ -star-switches.

The transformation builds on a construction introduced by Szeider [20] used in proofs for  $\text{TRANSITION-COMPATIBLE-}s$ - $t$ - $\text{PATH}$ . In particular, they show how to transform a vertex with a transition graph isomorphic to  $K_2 + K_2$  into one with a transition graph isomorphic to  $L_4$ , without effect to the existence of a compatible  $s$ - $t$ -path.

Due to the connection between  $\text{STCON-SWITCHING}$  and  $\text{TRANSITION-COMPATIBLE-}s$ - $t$ - $\text{PATH}$  illustrated in Chapter 4, we can adopt a similar strategy to convert  $\{1, 1\}$ -switches into  $\{1\}^*$ -star switches with switch-induced graphs isomorphic to  $L_4$ . Then, as already exploited in Theorem 4, we apply Construction 3 to convert these into  $\{2, 1\}$ -star switches.

However, to make this transformation work in the weighted setting, we must carefully choose the on- and off-positions of the switches, a distinction that was irrelevant in the unweighted case. Specifically, the transformation must ensure that if a  $\{1, 1\}$ -switch  $S$  is set to the off-position and  $uv \in S[1]$ , then the new switches enable a path between  $u$  and  $v$  only when set to the off-position. Likewise, if  $uv \in S[2]$  and  $S$  is set to the on-position, then the path between  $u$  and  $v$  should exist only when the new switches are set to the on-position.

The resulting construction transforming a  $\{1, 1\}$ -switch to a set of  $\{2, 1\}$ -star switches, illustrated in Fig. 6.2, is defined as follows:

**Construction 10.** Let  $\hat{G}$  be a switch graph and  $S \in \mathcal{S}(\hat{G})$  a  $\{1, 1\}$ -switch with  $S[1] = \{v_1v_3\}$  and  $S[2] = \{v_2v_4\}$ . We construct a new switch graph  $\hat{G}'$  from  $\hat{G}$  by introducing six fresh vertices  $\{u_1, u_2, u_3, u_4, w_1, w_2\}$  and replacing  $S$  with four new  $\{2, 1\}$ -star switches

$\{S_1, S_2, S_3, S_4\}$ , such that for each  $i \in [2]$  and  $j \in \{0, 2\}$ , we have:

$$S_{i+j}[i] := \{v_{i+j}u_{i+j}\} \quad \text{and} \quad S_{i+j}[3-i] := \{u_{1+j}u_{2+j}, u_{i+j}w_i\}.$$

With the help of Fig. 6.2, it is easy to see that  $v_1$  and  $v_3$  are connected if all new switches are set to the off-position, while  $v_2$  and  $v_4$  are connected if all new switches are set to the on-position. Crucially, all other configurations of the new switches do not enable any new paths between pairs of vertices in  $\{v_1, v_2, v_3, v_4\}$ . Consequently, the new switches mirror the behavior of the original  $\{1, 1\}$ -switch  $S$  in terms of connectivity.

After applying this transformation to every  $\{1, 1\}$ -switch in a switch graph  $\widehat{G}$ , we obtain a new switch graph  $\widehat{G}'$  consisting exclusively of  $\{2, 1\}$ -star switches. Based on the arguments above, every configuration of  $\widehat{G}$  can then be mapped to a configuration of  $\widehat{G}'$  with four times as many switches set to the on-position while preserving the connectivity of the original vertices. Conversely, every configuration of  $\widehat{G}'$  can be mapped to a configuration of  $\widehat{G}$  with at most one switch set to the on-position for every four switches in  $\widehat{G}'$ , again preserving the connectivity of the original vertices.

By quadrupling the activation budget, we thus obtain a valid parameterized reduction from ONOFF-STCON-SWITCHING restricted to  $\{1, 1\}$ -switches. It is important to note, however, that this transformation does not preserve the structure of the union graph. In particular, the resulting union graph is not necessarily series-parallel, and we lose any guarantees regarding planarity or bounded treewidth. Nevertheless, we obtain the following intractability result for ONOFF-STCON-SWITCHING:

**Theorem 19.** ONOFF-STCON-SWITCHING is  $W[1]$ -hard parameterized by the activation budget  $k$ , even if all switches are  $\{2, 1\}$ -star switches. Furthermore, assuming the ETH, the problem cannot be solved in time  $f(k)n^{o(\frac{k}{\log k})}$ , where  $f$  is an arbitrary computable function and  $n$  is the number of switches in the switch graph.

*Proof.* Let  $\widehat{G}$  be a switch graph consisting of  $\{1, 1\}$ -switches and  $k$  an integer representing the activation budget. Without loss of generality, we may assume that  $\widehat{G}$  contains no static edges, as these can be contracted without affecting connectivity. Moreover, we may assume that each switch position contains one edge. Otherwise, we introduce two fresh vertices  $u$  and  $v$ , and insert the edge  $uv$  into all empty positions. Since  $u$  and  $v$  are isolated from the rest of the graph, this modification does not affect the connectivity between the original vertices.

To obtain a valid parameterized reduction from STCON-SWITCHING, parameterized by the activation budget  $k$  and restricted to  $\{1, 1\}$ -switches, to the same parameterized problem restricted to  $\{2, 1\}$ -star switches, we apply Construction 10 to every switch in  $\widehat{G}$ , resulting in a new switch graph  $\widehat{G}'$ , and setting the new activation budget to  $k' := 4k$ . Clearly, the reduction can be achieved in polynomial time, and  $k'$  is bounded by a computable function of  $k$ .

To show correctness of the reduction, we aim to show that there exists a configuration  $c \in \mathcal{C}(\widehat{G})$  such that  $\widehat{G} \circ c$  admits an  $s$ - $t$ -path and  $|c|_2 \leq k$  if and only if there exists a configuration  $c' \in \mathcal{C}(\widehat{G}')$  such that  $\widehat{G}' \circ c'$  admits an  $s$ - $t$ -path and  $|c'|_2 \leq k'$ .

( $\Rightarrow$ ): Let  $c$  be a configuration of  $\widehat{G}$  such that  $\widehat{G} \circ c$  admits an  $s$ - $t$ -path and  $|c|_2 \leq k$ . We now construct a configuration  $c' \in \mathcal{C}(\widehat{G}')$  such that, for each switch  $S \in \mathcal{S}(\widehat{G})$  and  $i \in [4]$ , we have  $c'(S_i) := c(S)$ . Clearly,  $|c'|_2 = 4|c|_2 \leq 4k = k'$ . What remains to be shown is that  $\widehat{G}' \circ c'$  admits an  $s$ - $t$ -path.

Let  $e$  be an arbitrary edge in  $\widehat{G} \circ c$ , and let  $S \in \mathcal{S}(\widehat{G})$  be the switch such that  $e \in S[i]$  for  $i := c(S)$ . Denote the endpoints of  $e$  as  $v_i$  and  $v_{i+2}$ . For each  $j \in \{0, 2\}$ , we have  $v_{i+j}u_{i+j} \in S_{i+j}[i]$  and  $u_{1+j}u_{2+j}, u_{(3-i)+j}w_{3-i} \in S_{(3-i)+j}[i]$ , resulting in a path from  $v_{i+j}$  to  $w_{3-i}$  in  $\widehat{G}' \circ c'$ , where  $w_{3-i}$  and all other inner vertices of the path are auxiliary vertices introduced by the transformation of switch  $S$ . Combining these paths for  $j = 0$  and  $j = 2$  yields a path from  $v_i$  and  $v_{i+2}$ .

Consequently, each edge  $uv$  in the  $s$ - $t$ -path in  $\widehat{G} \circ c$  can be replaced by a corresponding path between  $u$  and  $v$  in  $\widehat{G}' \circ c'$ , thus resulting in an  $s$ - $t$ -path in  $\widehat{G}' \circ c'$ .

( $\Leftarrow$ ): Let  $c'$  be a configuration of  $\widehat{G}'$  such that  $\widehat{G}' \circ c'$  admits an  $s$ - $t$ -path  $P'$  and  $|c'|_2 \leq k'$ . We construct a configuration  $c \in \mathcal{C}(\widehat{G})$  such that, for each switch  $S \in \mathcal{S}(\widehat{G})$ , we have  $c(S) := \min_{i \in [4]} \{c'(S_i)\}$ . Thus,  $c(S) \leq \frac{1}{4} \sum_{i \in [4]} c'(S_i)$ , and consequently  $|c|_2 \leq \frac{1}{4} |c'|_2 \leq \frac{1}{4} k' = k$ . It remains to show that  $\widehat{G} \circ c$  admits an  $s$ - $t$ -path.

To do so, we process the vertices in  $P'$  sequentially and construct an  $s$ - $t$ -path  $P$  in  $\widehat{G} \circ c$  in parallel. The path  $P'$  starts and ends with original vertices of  $\widehat{G}$  (specifically,  $s(\widehat{G})$  and  $t(\widehat{G})$ ), so any auxiliary vertices introduced by the transformation must be internal and followed by further vertices.

Initially, let  $v$  be the first vertex in  $P'$ , and initialize  $P := (v)$ .

Assume  $v$  is not the last vertex in  $P'$ . Then, by construction, it must be followed by an auxiliary vertex introduced by the transformation of some  $\{1, 1\}$ -switch  $S$  in  $\widehat{G}$ . In particular, there exists a switch  $S_{i+j}$  with  $i = c(S_{i+j})$  and  $j \in \{0, 2\}$  such that  $S_{i+j}[i] = \{v_{i+j}u_{i+j}\}$ , where  $v = v_{i+j}$  and  $u_{i+j}$  is the next vertex following  $v$  in  $P'$ .

The only other switch incident to  $u_{i+j}$  is  $S_{(3-i)+j}$  via an edge  $u_{1+j}u_{2+j}$  contained in position  $i$ . Thus, we have  $c(S_{(3-i)+j}) = i$  and  $u_{i+j}$  is followed by the auxiliary vertex  $u_{(3-i)+j}$  in  $P'$ .

Next, since  $u_{(3-i)+j}$  is not incident to any switches other than  $S_{(3-i)+j}$ , and  $S_{(3-i)+j}[i]$  contains exactly one additional edge  $u_{(3-i)+j}w_{3-i}$ , it follows that  $u_{(3-i)+j}$  is followed by the auxiliary vertex  $w_{3-i}$  in  $P'$ .

The vertex  $w_{3-i}$  has only one additional incident switch, namely  $S_{(3-i)+(2-j)}$  via the edge  $w_{3-i}u_{(3-i)+(2-j)}$  contained in position  $i$ . Hence, we have  $c(S_{(3-i)+(2-j)}) = i$  and  $w_{3-i}$  is followed by the auxiliary vertex  $u_{(3-i)+(2-j)}$  in  $P'$ .

Although  $u_{(3-i)+(2-j)}$  is incident to switch  $S_{(i)+(2-j)}$ , it is through the same edge  $u_{1+(2-j)}u_{2+(2-j)}$  already contained in  $S_{(3-i)+(2-j)}[i]$ . As there are no other switches

or edges in  $S_{(3-i)+(2-j)}[i]$  incident to  $u_{(3-i)+(2-j)}$ , this vertex must be followed by the auxiliary vertex  $u_{i+(2-j)}$  in  $P'$ .

At this point, the only other switch incident to  $u_{i+(2-j)}$  is  $S_{i+(2-j)}$ . Suppose this switch is configured to position  $i - 3$ , consisting of edges  $u_{1+(2-j)}u_{2+(2-j)}$  and  $u_{i+(2-j)}w_i$ . Since  $P'$  is a path, we cannot return to the previously visited vertex  $u_{(3-i)+(2-j)}$ , so the only possible successor of  $u_{i+(2-j)}$  would be the auxiliary vertex  $w_i$ . However,  $w_i$  is only incident to switch  $S_{i+j}$  via an edge in position  $i - 3$ , while  $S_{i+j}$  is already configured to position  $i$ . This creates a contradiction, as the path cannot continue through  $w_i$ . Therefore, switch  $S_{i+(2-j)}$  must be configured to position  $i$ , which contains a single edge  $u_{i+(2-j)}v_{i+(2-j)}$ . Hence,  $u_{i+(2-j)}$  is followed by the original vertex  $v_{i+(2-j)}$  in  $P'$ .

In summary, for all  $\ell \in [4]$ , we have  $c'(S_\ell) = i$ , resulting in  $c(S) = i$ . By construction, edge  $v_{i+j}v_{i+(2-j)}$  is contained in position  $i$  of switch  $S$ , and thus part of  $\widehat{G} \circ c$ . Since the last vertex added to  $P$  was  $v_{i+j}$ , we extend  $P$  by appending  $v_{i+(2-j)}$ .

We repeat the procedure for  $v = v_{i+(2-j)}$ . Eventually, this constructs a full  $s$ - $t$ -path  $P$  in  $\widehat{G} \circ c$ .

In Theorem 15, we established that STCON-SWITCHING is W[1]-hard parametrized by the activation budget  $k$ , even if all switches are  $\{1, 1\}$ -switches. The parameterized reduction described above thus shows that the same problem remains W[1]-hard when restricted to  $\{2, 1\}$ -star-switches.

Moreover, Theorem 15 shows that, unless the ETH fails, STCON-SWITCHING for  $\{1, 1\}$ -switches cannot be decided in time  $f(k)n^{o(\frac{k}{\log k})}$ , where  $f$  is a computable function and  $n$  is the number of switches in the switch graph. In the reduced instance, the number of switches  $m$  as well as the activation budget  $k'$  is a factor of four larger than in the original instance, i.e.,  $m' = 4n$  and  $k' = 4k$ .

For the sake of contradiction, suppose STCON-SWITCHING restricted to  $\{2, 1\}$ -star switches admits an algorithm running in time  $g(k')m^{o(\frac{k'}{\log k'})}$  for some computable function  $g$ . Then we can transform an instance restricted to  $\{1, 1\}$ -switches into an instance restricted to  $\{2, 1\}$ -star switches in time  $\mathcal{O}(n)$  and solving the instance using the suggested algorithm in time  $g(4k)(4n)^{o(\frac{4k}{\log(4k)})}$ , which is equivalent to  $f(k)n^{o(\frac{k}{\log k})}$  for some computable function  $f$ , contradicting the ETH.  $\square$

**Discussion** In summary, this chapter extends the results of Chapter 4 by demonstrating the intractability of STCON-SWITCHING for general  $\{1\}^*$ -switches, even when the union graph is series-parallel. We further address switches with multiple edges per position, establishing intractability for  $\{2, 1\}$ -switches, even when all switch-induced graphs are star graphs or the union graph is a path. Future research may further investigate the

complexity dichotomy for these more general switches, analogous to the one established for  $\{1\}^*$ -switches in Theorem 6.

Additionally, we analyze the weighted variant, proving  $W[1]$ -hardness for `ONOFF-STCON-SWITCHING` under the same structural restrictions as for the unweighted variant. Notably, tractable cases for `WEIGHTED-STCON-SWITCHING`, particularly when restricted to complete multipartite switches, remain open and present an interesting direction for future work.



# Global Connectivity Switching and Satisfiability

In Chapter 3, we presented a polynomial-time procedure for solving GCON-SWITCHING when restricted to matroid switches (Theorem 1), and extended this approach to the weighted variant for  $\{1\}^*$ -matroid switches (Theorem 2). However, the intractability of GCON-SWITCHING and ONOFF-GCON-SWITCHING remains unresolved.

In this chapter, we address this gap by providing reductions from (weighted) satisfiability problems, using techniques similar to those in Chapter 6. Unlike the reductions for STCON-SWITCHING, these reductions do not work on arbitrary formulas. Instead, it requires the input formula to be 2-normalized. Despite this restriction, we are still able to establish intractability results for GCON-SWITCHING and ONOFF-GCON-SWITCHING under quite constrained settings — particularly with respect to switch size and the structure of the union graph.

In Section 7.1, we reuse the reduction from Chapter 6 to establish the intractability of GCON-SWITCHING and ONOFF-GCON-SWITCHING, parameterized by the activation budget, even when restricted to  $\{2, 1\}$ -switches and the union graph is a path. In Section 7.2, we slightly adapt this reduction to show that the problem remains intractable for  $\{2, 1\}$ -star switches, even when the union graph forms a star. In the unweighted case, we further refine the construction to bound the switch degree of each vertex. Specifically, in Section 7.3, we prove that GCON-SWITCHING is NP-hard for  $\{2, 1\}$ -star switches, even when the union graph is planar and the switch degree of every vertex is at most three.

## 7.1 Initial Intractability Results

Consider an arbitrary 2-normalized formula  $\varphi$ . For every configuration of the corresponding switch graph  $\widehat{G}_\varphi$  constructed as in Construction 9 that admits an  $s$ - $t$ -path,

all vertices of  $\widehat{G}_\varphi$  are included in the path. Consequently, the configured switch graph is globally connected. Conversely, every configuration of  $\widehat{G}_\varphi$  that does not admit an  $s$ - $t$ -path is trivially not globally connected. Hence, analogous to Theorem 14, we obtain the following intractability result for 2-1-bounded 2-normalized formulas:

**Theorem 20.** *GCON-SWITCHING is NP-hard, even if all switches are  $\{2, 1\}$ -switches and the union graph is a path.*

The above observations immediately extend to the weighted variant of the problem as well. Thus, analogous to Theorem 17, we obtain the following intractability result for 2-1-bounded 2-normalized formulas:

**Theorem 21.** *ONOFF-GCON-SWITCHING is W[1]-hard, even if all switches are  $\{2, 1\}$ -switches and the union graph is a path. Furthermore, assuming the ETH, the problem cannot be solved in time  $f(k)n^{o(\frac{k}{\log k})}$ , where  $f$  is an arbitrary computable function and  $n$  is the number of switches in the switch graph.*

## 7.2 Intractability for Star Switches

If we restrict ourselves to  $\{1, 1\}$ -switches, Theorem 1 and Theorem 2 show that both GCON-SWITCHING and ONOFF-GCON-SWITCHING are decidable in polynomial time. This rules out the use of the construction similar to Construction 10 to demonstrate intractability for star switches. However, a slight modification of Construction 9 allows us to enforce connectivity of the switch-induced subgraph for each switch.

Instead of modeling conjunction using a series-composition of switch graphs in the construction of  $\widehat{G}_\varphi$  as presented in Construction 9, we introduce a new operation called *star-composition* in which only the source vertices are merged into a single vertex. Fig. 7.1 illustrates the updated construction process on a simple example. Observe that in any configuration where the resulting graph is connected, both component graphs of a star-composed switch graph must themselves be connected. Intuitively, this models a conjunction for global connectivity.

To formally define this new construction, we begin by introducing a new type of switch graph. A switch graph  $\widehat{G}$  is a *series-parallel-star switch graph* if it has a distinguished source  $s(\widehat{G})$  and satisfies the following conditions:

- *Series-parallel switch graph:*  $\widehat{G}$  is a series-parallel switch graph.
- *Star-composition:*  $\widehat{G}$  is obtained by taking the union of two series-parallel-star switch graphs  $\widehat{G}_1$  and  $\widehat{G}_2$  identifying  $s(\widehat{G}) = s(\widehat{G}_1) = s(\widehat{G}_2)$ .

Unlike series-parallel switch graphs, series-parallel-star switch graphs have no distinct target vertex. Hence, a series-composition of star-composed switch graphs is not possible. Consequently, by replacing the series-composition with a star-composition in the

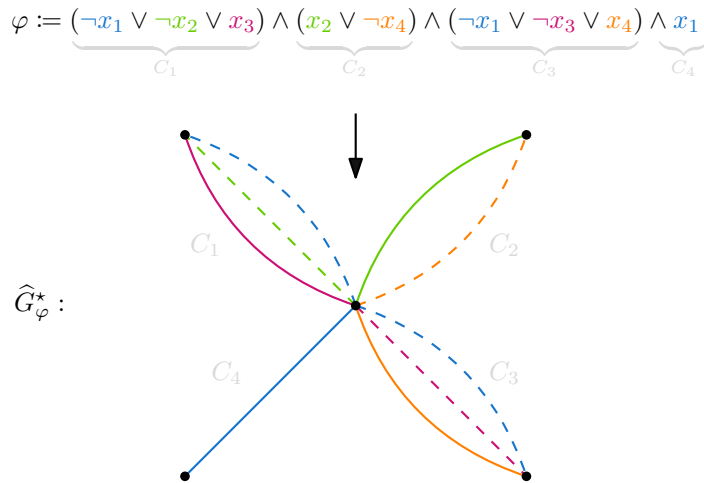


Figure 7.1: Example switch graph  $\widehat{G}_\varphi^*$  for a 2-normalized formula  $\varphi$ , constructed as described in Construction [11](#). Each color corresponds to a variable and its associated switch; solid edges represent edges of the on-position, while dashed edges represent edges of the off-position.

construction of  $\widehat{G}_\varphi^*$ , we are limited to 2-normalized formulas. Formally, we update the inductive construction process as follows:

**Construction 11.** For every 2-normalized formula  $\varphi$ , we construct a corresponding series-parallel-star switch graph  $\widehat{G}_\varphi^*$  as follows:

- If  $\varphi := \psi \wedge \omega$ , then  $\widehat{G}_\varphi^*$  is formed by the star-composition of  $\widehat{G}_\psi^*$  and  $\widehat{G}_\omega^*$ .
- Otherwise,  $\widehat{G}_\varphi^*$  is equal to  $\widehat{G}_\varphi$  as described in Construction [9](#).

Throughout this process, switches corresponding to the same variable retain their identity and are merged accordingly. As a result,  $\widehat{G}_\varphi^*$  contains exactly one switch  $S_x$  for each variable  $x \in \mathcal{V}(\varphi)$ .

Since  $\varphi$  is in conjunctive normal form, the series-parallel switch graph corresponding to each clause in  $\varphi$  is constructed by a sequence of parallel-compositions of single-edge switch graphs. Each such construction yields a switch graph consisting of two vertices, with each involved switch containing a single edge between them. Applying the star-composition to these clause-specific switch graphs produces a series-parallel-star switch graph with the structure of a star (see Fig. [7.1](#) for an illustration). Specifically, the union graph of  $\widehat{G}_\varphi^*$  forms a star graph, implying that every switch  $S_x \in \mathcal{S}(\widehat{G}_\varphi^*)$  is a star switch. Furthermore, as in Construction [9](#), each  $a$ - $b$ -bounded variable in  $\varphi$  corresponds to an  $\{a, b\}$ -switch in  $\widehat{G}_\varphi^*$ .

Analogous to Lemma [10](#), we map each interpretation  $I$  of  $\varphi$  to a configuration  $c$  of  $\widehat{G}_\varphi^*$  by setting each switch  $S_x \in \mathcal{S}(\widehat{G}_\varphi^*)$  to the on-position if and only if the corresponding variable  $x$  is set to true in  $I$ . This correspondence is captured by the bijection  $f_\varphi : \mathcal{I}(\varphi) \rightarrow \mathcal{C}(\widehat{G}_\varphi^*)$ , formally defined for each interpretation  $I \in \mathcal{I}(\varphi)$  and switch  $S_x \in \mathcal{S}(\widehat{G}_\varphi^*)$  as follows:

$$f_\varphi(I)(S_x) := I(x) + 1.$$

The following lemma proves the correctness of this mapping:

**Lemma 11.** *Let  $\varphi$  be a 2-normalized propositional formula and  $I$  an interpretation of  $\varphi$ . Then,  $\widehat{G}_\varphi^*$  configured by  $f_\varphi(I)$  is globally connected if and only if  $I(\varphi) = 1$ .*

*Proof.* Let  $c := f_\varphi(I)$ . The proof is by induction on the structure of the formula  $\varphi$ .

**Base case:** If  $\varphi$  is not a conjunction of two subformulas, then  $\widehat{G}_\varphi^*$  is equal to  $\widehat{G}_\varphi$  as defined in Construction [9](#). Since  $\varphi$  is 2-normalized, it must be composed solely of literals and disjunctions. Consequently,  $\widehat{G}_\varphi$  is constructed from single-edge switch graphs and parallel-compositions. It follows that  $\widehat{G}_\varphi^*$  is composed of precisely two distinct vertices — a source vertex  $s(\widehat{G}_\varphi^*)$  and a target vertex  $t(\widehat{G}_\varphi^*)$ . As a consequence,  $c \circ \widehat{G}_\varphi^*$  is globally connected exactly when  $c \circ \widehat{G}_\varphi$  admits an  $s$ - $t$ -path. In Lemma [10](#), we have shown that  $c \circ \widehat{G}_\varphi$  admits an  $s$ - $t$ -path if and only if  $I(\varphi) = 1$ . Thus, we obtain the desired equivalence.

**Inductive step:** Assume that the statement holds for all subformulas of  $\varphi$ . Since the base case already covers all formulas that are not conjunctions, we can conclude that  $\varphi$  must be a conjunction of two subformulas  $\psi$  and  $\omega$ . By construction,  $\widehat{G}_\varphi^*$  is formed by the star-composition of  $\widehat{G}_\psi^*$  and  $\widehat{G}_\omega^*$ . Observe that  $c \circ \widehat{G}_\varphi^*$  is globally connected if and only if both  $c \circ \widehat{G}_\psi^*$  and  $c \circ \widehat{G}_\omega^*$  are globally connected. By the inductive hypothesis, this is precisely the case when  $I(\psi) = 1$  and  $I(\omega) = 1$ . Finally,  $I(\psi) = 1$  and  $I(\omega) = 1$  if and only if  $I(\varphi) = I(\psi \wedge \omega) = 1$ .  $\square$

Analogous to Corollary [2](#) and Corollary [3](#) obtained from Lemma [10](#) showing the correctness of the reduction to STCON-SWITCHING and WEIGHTED-STCON-SWITCHING, respectively, we can equivalently derive the following two corollaries from Lemma [11](#):

**Corollary 4.** *Let  $\varphi$  be a 2-normalized propositional formula. Then  $\varphi$  is satisfiable if and only if there exists a configuration of  $\widehat{G}_\varphi^*$  such that the configured switch graph is globally connected.*

**Corollary 5.** *Let  $\varphi$  be a 2-normalized propositional formula and  $k$  some integer. Then there exists an interpretation  $I \in \mathcal{I}(\varphi)$  such that  $I(\varphi) = 1$  and  $|I|_1 \leq k$  if and only if there exists a configuration  $c$  of  $\widehat{G}_\varphi^*$  such that  $\widehat{G}_\varphi^* \circ c$  is globally connected and  $|c|_2 \leq k$ .*

In Chapter [5](#), we have established that the satisfiability problem remains NP-hard even for 2-1-bounded 2-normalized formulas (Theorem [7](#)). By reducing this variant

of the satisfiability problem to GCON-SWITCHING using Corollary 4 and applying the structural observations from Construction 11, we derive the following intractability result for GCON-SWITCHING:

**Theorem 22.** GCON-SWITCHING is NP-hard, even if all switches are  $\{2, 1\}$ -star switches and the union graph is a star graph.

Moving on to the weighted variant, recall that Theorem 13 established that weighted satisfiability is W[1]-hard even for 2-1-bounded 2-normalized formulas. Moreover, the theorem shows that, unless the ETH fails, this problem cannot be solved in time  $g(k)n^{o(\frac{k}{\log k})}$ , where  $g$  is some computable function,  $k$  is the maximum weight of a satisfying truth assignment, and  $n$  is the number of variables in the formula. Similar to Theorem 17, we reduce this restricted variant of weighted satisfiability to ONOFF-GCON-SWITCHING via Corollary 5, yielding the following intractability result:

**Theorem 23.** ONOFF-GCON-SWITCHING is W[1]-hard, even if all switches are  $\{2, 1\}$ -star switches and the union graph is a star graph. Furthermore, assuming the ETH, the problem cannot be solved in time  $f(k)n^{o(\frac{k}{\log k})}$ , where  $f$  is an arbitrary computable function and  $n$  is the number of switches in the switch graph.

### 7.3 Intractability for Vertices of Bounded Switch Degree

Although the theorems above establish the intractability of GCON-SWITCHING and ONOFF-GCON-SWITCHING restricted to  $\{2, 1\}$ -star switches with a planar union graph, they do not impose any constraints on the switch degree of the vertices. In fact, in Construction 11, the source vertex of the series-parallel-star switch graph is incident to all switches.

To address this, we replace the source vertex with multiple new vertices — one for each switch — and update the endpoint of the affected edges accordingly. We then add static edges connecting these new vertices to form a single connected component. Notice that, after contracting these static edges, we recover the original switch graph. Since edge contractions preserve the existence of a globally connected configuration, this yields a valid transformation.

In the transformed switch graph, each new vertex has a switch degree equal to one plus the number of incident static edges. Importantly, while the union graph of the resulting switch graph may no longer be a star graph, each individual switch remains a  $\{2, 1\}$ -star switch.

Building on this transformation, we show in Theorem 24 that GCON-SWITCHING remains intractable even when the switch degree of every vertex is bounded by three and the union graph remains planar. To formally prove this result, we begin by presenting a modified construction of the switch graph, based on the above transformation, for a given 2-normalized formula:

$$\varphi := \underbrace{(\neg x_1 \vee \neg x_2 \vee x_3)}_{C_1} \wedge \underbrace{(x_2 \vee \neg x_4)}_{C_2} \wedge \underbrace{(\neg x_1 \vee \neg x_3 \vee x_4)}_{C_3} \wedge \underbrace{x_1}_{C_4}$$

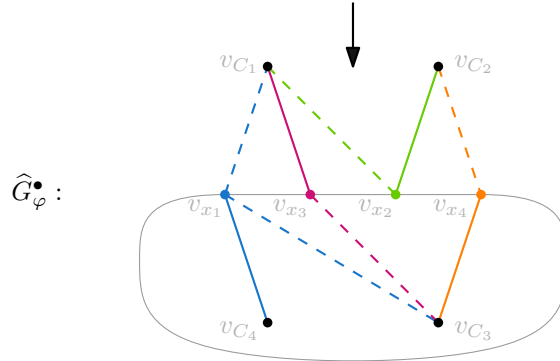


Figure 7.2: Example switch graph  $\widehat{G}_\varphi^\bullet$  for a propositional formula  $\varphi \in \Delta_{2,3}^{\text{planar}}$ , constructed as described in Construction 12. Each color corresponds to a variable and its associated vertex and switch; solid edges represent edges of the on-position, while dashed edges represent edges of the off-position. The grey edges represent static edges that connect the variable vertices in a cycle.

**Construction 12.** Given a formula  $\varphi \in \Delta_2$ , we construct a corresponding switch graph  $\widehat{G}_\varphi^\bullet$ . The vertex set  $V(\widehat{G}_\varphi^\bullet)$  consists of *variable vertex*  $v_x$  for each variable  $x \in \mathcal{V}(\varphi)$  and a *clause vertex*  $v_C$  for each clause  $C \in \mathcal{C}(\varphi)$ . The switch set  $\mathcal{S}(\widehat{G}_\varphi^\bullet)$  consists of a switch  $S_x$  for each variable  $x \in \mathcal{V}(\varphi)$  such that  $S_x[1] = \{v_x v_C \mid C \in \mathcal{C}(\varphi), \neg x \in C\}$  and  $S_x[2] = \{v_x v_C \mid C \in \mathcal{C}(\varphi), x \in C\}$ . Furthermore, it contains a static edge  $S_e = (\{e\})$  for each edge  $e$  part of the variable cycle in the associated graph  $G(\varphi)$ . Fig. 7.2 illustrates the construction of  $\widehat{G}_\varphi^\bullet$  for a simple example.

By connecting the variable vertices using static edges according to the variable cycle in the associated graph  $G(\varphi)$ , we ensure that the union graph of  $\widehat{G}_\varphi^\bullet$  is isomorphic to  $G(\varphi)$ . This construction guarantees that the union graph of  $\widehat{G}_\varphi^\bullet$  is planar whenever  $G(\varphi)$  is planar. Moreover, each clause vertex  $v_C$  has switch degree equal to the number of literals in clause  $C$ , while each variable vertex  $v_x$  has switch degree three — two from the static edges and one from the corresponding switch  $S_x$ . As a result, the constructed switch graph  $\widehat{G}_\varphi^\bullet$  derived from a formula  $\varphi \in \Delta_{2,3}^{\text{planar}}$  fulfills both desired properties: the union graph is planar, and every vertex has switch degree at most three.

For our purposes, we identify switches in both  $\widehat{G}_\varphi^*$  (from Construction 11) and  $\widehat{G}_\varphi^\bullet$  if they correspond to the same variable  $x \in \mathcal{V}(\varphi)$ . This allows us to extend any configuration  $c$  of  $\widehat{G}_\varphi^*$  to a configuration  $c'$  of  $\widehat{G}_\varphi^\bullet$  by additionally setting all static edges in  $\widehat{G}_\varphi^\bullet$  to the off-position. This defines a bijection  $f_\varphi : \mathcal{C}(\widehat{G}_\varphi^*) \rightarrow \mathcal{C}(\widehat{G}_\varphi^\bullet)$  such that for each  $S \in \mathcal{S}(\widehat{G}_\varphi^\bullet)$ ,

we have:

$$f_\varphi(c)(S) := \begin{cases} c(S) & \text{if } S \in \mathcal{S}(\widehat{G}_\varphi^*) \\ 1 & \text{otherwise} \end{cases}.$$

Note that for every configuration  $c \in \mathcal{C}(\widehat{G}_\varphi^*)$ ,  $\widehat{G}_\varphi^* \circ c$  is isomorphic to  $\widehat{G}_\varphi^\bullet \circ f_\varphi(c)$  after contracting the static edges. Since edge contractions preserve the existence of a globally connected configuration, we derive the following lemma:

**Lemma 12.** *Let  $\varphi \in \Delta_2$  be some formula and  $c$  be a configuration of  $\widehat{G}_\varphi^*$  constructed as described in Construction 11. Then  $\widehat{G}_\varphi^* \circ c$  is globally connected if and only if  $\widehat{G}_\varphi^\bullet \circ f_\varphi(c)$  is globally connected.*

Since Theorem 7 establishes that satisfiability remains NP-hard even for 2-1-bounded formulas in 3-CNF and a planar associated graph, we can combine this result with Lemma 11 and Lemma 12 to obtain a reduction demonstrating the intractability of GCON-SWITCHING. Like in Construction 11, the 2-1-boundedness of the formula ensures that each switch is a  $\{2, 1\}$ -star switch. Combined with the above observations that the union graph is planar and each vertex has switch degree at most three, we obtain the following intractability result for GCON-SWITCHING:

**Theorem 24.** *GCON-SWITCHING is NP-hard, even if all switches are  $\{2, 1\}$ -star switches, the union graph is planar, and the switch degree of each vertex is at most three.*

Unfortunately, for the weighted variant of the problem, we do not know whether  $\text{WSAT}(\Delta_{2,3} \cap B_{2,1})$  is  $\text{W}[1]$ -hard, regardless of whether the associated graph is planar or not. Thus, we cannot conclude that  $\text{ONOFF-GCON-SWITCHING}$  is  $\text{W}[1]$ -hard for small star switches whilst keeping the switch degree of each vertex bounded by a constant. We leave this as an open question.

**Discussion** Generally, for the unweighted variant of GCON-SWITCHING, we have shown that the problem is polynomial-time solvable when restricted to matroid switches (Theorem 1). However, the problem becomes intractable for  $\{2, 1\}$ -switches, which are the simplest non-matroid switches. Despite this, it is not straightforward to establish a dichotomy result analogous to Theorem 6. Specifically, given a class of switches  $\mathcal{A}$  without isolated vertices, can we assert that GCON-SWITCHING is polynomial-time solvable for all switches from  $\mathcal{A}$  if and only if  $\mathcal{A}$  consists exclusively of matroid switches?

Many of the intractability results in this chapter also carry over to the weighted variant, except in the case where the union graph is planar and each vertex has switch degree at most three. This leaves open the question of whether  $\text{ONOFF-GCON-SWITCHING}$  is  $\text{W}[1]$ -hard under these constraints.



# $s$ - $t$ -Disconnectivity Switching and Satisfiability

In this chapter, we investigate the complexity of `STDISCON-SWITCHING` and `WEIGHTED-STDISCON-SWITCHING` — two variants of the switching problem not addressed by our previous results. In both cases, the goal is to find a configuration of a given switch graph such that the resulting graph *does not* admit an  $s$ - $t$ -path. In other words, we seek a configuration that disconnects the source from the target vertex.

Due to their similarity to `STCON-SWITCHING`, we can leverage many of the tools developed in Chapter 6 to analyze the complexity of these two variants. In Section 8.1, we explore this relationship in detail, establishing hardness for both `STDISCON-SWITCHING` and `ONOFF-STDISCON-SWITCHING`, parameterized by the activation budget, even for  $\{1, 1\}$ -switches when the union graph is series-parallel. Building on these results, we present in Section 8.2 a transformation that reduces  $\{1, 1\}$ -switches to  $\{2, 1\}$ -star switches, thereby proving the intractability of both `STDISCON-SWITCHING` and `ONOFF-STDISCON-SWITCHING` under this restriction as well.

## 8.1 Connection to $s$ - $t$ -Connectivity Switching

Given a propositional formula  $\varphi$  in negation normal form, Construction 9 demonstrates how we can construct a corresponding series-parallel switch graph  $\hat{G}_\varphi$ . As already described in Section 6.1, we can map an interpretation  $I \in \mathcal{I}(\varphi)$  to a configuration  $c$  of the corresponding switch graph  $\hat{G}_\varphi$  by setting each switch  $S_x \in \mathcal{S}(\hat{G}_\varphi)$  to the on-position if and only if the corresponding variable  $x$  is set to true in  $I$ . This mapping is captured by the bijection  $f_\varphi : \mathcal{I}(\varphi) \rightarrow \mathcal{C}(\hat{G}_\varphi)$ , defined as  $f_\varphi(I)(S_x) := I(x) + 1$  for each switch  $S_x \in \mathcal{S}(\hat{G}_\varphi)$ .

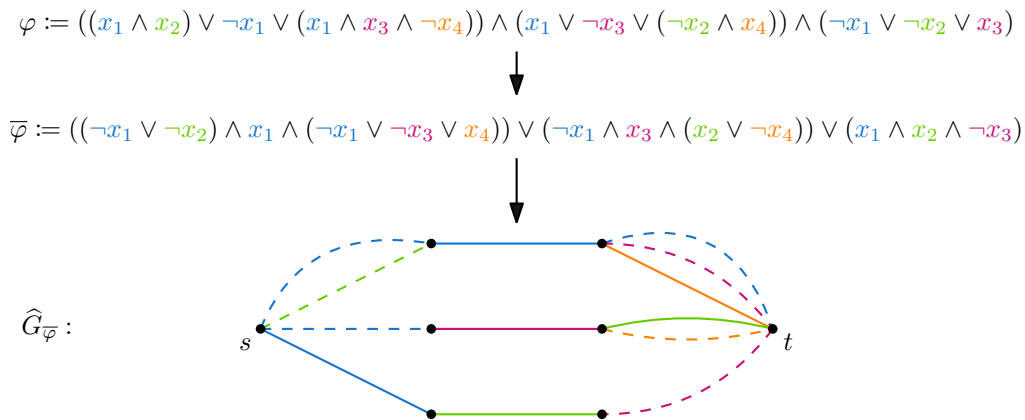


Figure 8.1: Example switch graph  $\widehat{G}_{\bar{\varphi}}$  of the negated formula  $\bar{\varphi}$  in NNF, constructed as described in Construction 9, derived from a propositional formula  $\varphi$  in NNF. Each color corresponds to a variable and its associated switch; solid edges represent edges of the on-position, while dashed edges represent edges of the off-position.

Given an interpretation  $I \in \mathcal{I}(\varphi)$ , Lemma 10 established that the configured switch graph  $\widehat{G}_{\varphi} \circ f_{\varphi}(I)$  admits an  $s$ - $t$ -path if and only if  $I(\varphi) = 1$ . Conversely,  $\widehat{G}_{\varphi} \circ f_{\varphi}(I)$  does not admit an  $s$ - $t$ -path if and only if  $I(\varphi) = 0$ . This yields a reduction from the falsifiability problem for propositional formulas to  $\text{STDISCON-SWITCHING}$ .

Moreover, since  $I(\varphi) = 1$  if and only if  $I(\neg\varphi) = 0$ , we can equivalently reduce from the satisfiability problem by negating the formula before constructing the corresponding switch graph. Although  $\neg\varphi$  may not be in negation normal form (NNF), as required by Construction 9, we can transform it into an equisatisfiable formula  $\bar{\varphi}$  in NNF in linear time without altering the set of variables. This transformation inverts the polarity of each literal and switches conjunctions and disjunctions. Combined with the equivalences established above, we obtain that

$$I(\varphi) = 1 \iff I(\neg\varphi) = I(\bar{\varphi}) = 0 \iff \widehat{G}_{\bar{\varphi}} \circ f_{\bar{\varphi}}(I) \text{ does not admit an } s\text{-}t\text{-path.}$$

This chain of equivalences, illustrated in Fig. 8.1, establishes the necessary reduction to prove the following corollary:

**Corollary 6.** *Let  $\varphi$  be a propositional formula in negation normal form. Then  $\varphi$  is satisfiable if and only if there exists a configuration of  $\widehat{G}_{\bar{\varphi}}$  such that the configured switch graph does not admit an  $s$ - $t$ -path.*

Concerning the weighted variant of the problem, note that the weight of an interpretation  $I \in \mathcal{I}(\varphi)$  corresponds to the number of switches set to the on-position in  $f_{\varphi}(I)$ , that is  $|I|_1 = |f_{\varphi}(I)|_2$ . This directly yields the correctness of the parameterized reduction from weighted satisfiability to  $\text{ONOFF-STDISCON-SWITCHING}$ :

**Corollary 7.** *Let  $\varphi$  be a propositional formula in negation normal form and  $k$  an integer. Then there exists an interpretation  $I \in \mathcal{I}(\varphi)$  such that  $I(\varphi) = 1$  and  $|I|_1 \leq k$  if and only if there exists a configuration  $c$  of  $\widehat{G}_{\bar{\varphi}}$  such that the configured switch graph does not admit an  $s$ - $t$ -path and  $|c|_2 \leq k$ .*

To establish intractability, we apply the above reductions to constrained variants of the satisfiability problem discussed in Chapter 5, thereby obtaining hardness results for restricted classes of switch graphs. As in Section 6.1, we rely on structural correspondences between formulas and switch graphs. For instance, we observed that an  $a$ - $b$ -bounded formula translates into a switch graph in which each switch is an  $a, b$ -switch, which does not change when negating the formula. Furthermore, since the reduction involves constructing a series-parallel switch graph, the union graph must be series-parallel as well.

Analogous to Theorem 15, we obtain intractability for  $\text{STDISCON-SWITCHING}$  even when restricted to  $\{1, 1\}$ -switches, by reducing from the satisfiability problem constrained to 1-1-bounded formulas, which is  $\text{NP-hard}$  by Theorem 8, via Corollary 6:

**Theorem 25.**  *$\text{STDISCON-SWITCHING}$  is  $\text{NP-hard}$ , even if all switches are  $\{1, 1\}$ -switches and the union graph is series-parallel.*

Contrary to Theorem 15, where an analogous reduction for  $\text{STCON-SWITCHING}$  ensured that the union graph is a path, we must account for the fact that if  $\varphi \in \Delta_t$ , then  $\bar{\varphi} \in \nabla_t \subseteq \Delta_{t+1}$ . As a result, the necessary structural properties of the propositional formula may no longer hold, and we cannot guarantee that the union graph remains a path.

Turning to the weighted variant of the problem, we obtain a result analogous to Theorem 16 by applying a parameterized reduction via Corollary 7 from the weighted satisfiability problem, which is  $\text{W[SAT]}$ -hard when parameterized by the maximum weight of the truth assignment:

**Theorem 26.**  *$\text{ONOFF-STDISCON-SWITCHING}$  is  $\text{W[SAT]}$ -hard parameterized by  $k$ , even if the union graph is series-parallel.*

Furthermore, we can establish an analogous result to Theorem 18, which shows that  $\text{ONOFF-STCON-SWITCHING}$  is  $\text{W[1]}$ -hard and can not be solved in subexponential time unless the  $\text{ETH}$  fails, even if all switches are  $\{1, 1\}$ -switches. Since the negation of  $\varphi$  can be achieved in linear time, we derive a similar reduction from weighted satisfiability restricted to 1-1-bounded formulas, which is  $\text{W[1]}$ -hard by Theorem 12, via Corollary 7:

**Theorem 27.**  *$\text{ONOFF-STDISCON-SWITCHING}$  is  $\text{W[1]}$ -hard, even if all switches are  $\{1, 1\}$ -switches and the union graph is a series-parallel graph. Furthermore, assuming the  $\text{ETH}$ , the problem cannot be solved in time  $f(k)n^{o\left(\frac{k}{\log k}\right)}$ , where  $f$  is an arbitrary computable function and  $n$  is the number of switches in the switch graph.*

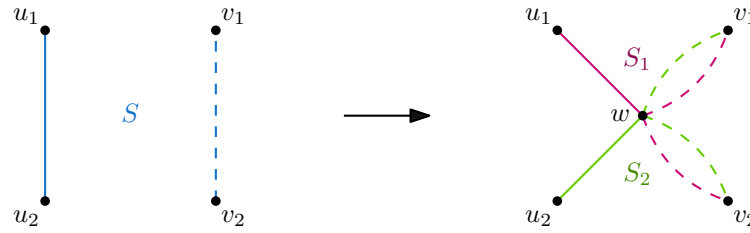


Figure 8.2: Example of the transformation from a  $\{1, 1\}$ -switch to two  $\{2, 1\}$ -star switches according to Construction 13. Each switch is represented by a different color, with solid edges indicating the on-position and dashed edges indicating the off-position.

## 8.2 Intractability for Star Switches

Similar to the approach in Section 6.3.1, where we transformed  $\{1, 1\}$ -switches into  $\{2, 1\}$ -star switches without effect to the existence of a compatible configuration with an  $s$ - $t$ -path, we can apply a comparable transformation in the context of  $s$ - $t$ -disconnectivity. However, in this case, the transformation is significantly simpler.

The following construction shows how a  $\{1, 1\}$ -switch can be replaced with two  $\{2, 1\}$ -star switches through the addition of a fresh vertex. We then establish the correctness of the transformation with respect to  $s$ - $t$ -disconnectivity in Lemma 13 and use it to derive intractability results for  $\text{STDISCON-SWITCHING}$  and  $\text{ONOFF-STDISCON-SWITCHING}$  restricted to  $\{2, 1\}$ -star switches in Theorem 28 and Theorem 29.

**Construction 13.** Let  $\widehat{G}$  be a switch graph and  $S \in \mathcal{S}(\widehat{G})$  a  $\{1, 1\}$ -switch with  $S[1] := \{v_1v_2\}$  and  $S[2] := \{u_1u_2\}$ . We construct a new switch graph  $\widehat{G}'$  by introducing a fresh vertex  $w$  and replacing  $S$  with two new  $\{2, 1\}$ -switches,  $S_1$  and  $S_2$ . For each  $i \in [2]$ , the positions of  $S_i$  are:

$$S_i[1] := \{v_1w, v_2w\} \quad \text{and} \quad S_i[2] := \{u_iw\}.$$

An illustration of this transformation is shown in Fig. 8.2.

Intuitively, if both new switches are set to the off-position, then  $v_1$  and  $v_2$  are connected via  $w$ , while  $u_1$  and  $u_2$  remain disconnected. Conversely, if both switches are set to the on position, then  $u_1$  and  $u_2$  are connected via  $w$ , while  $v_1$  and  $v_2$  remain disconnected. This replicates the behavior of the original switch  $S$ .

However, we must also consider the intermediate cases where one switch is set to the on-position and the other to the off-position. In such cases,  $v_1$ ,  $v_2$ , and either  $u_1$  or  $u_2$  become connected through  $w$ . If such a configuration does not admit an  $s$ - $t$ -path, we can always replace it with the configuration in which both switches are set to the off-position, which still blocks the  $s$ - $t$ -path while not increasing the number of switches in the on-position. Thus, in the context of  $s$ - $t$ -disconnectivity, these intermediate states are never more favorable.

**Lemma 13.** *Let  $\widehat{G}$  be a switch graph and  $S \in \mathcal{S}(\widehat{G})$  a  $\{1, 1\}$ -switch with two edges. Furthermore, let  $\widehat{G}'$  be the switch graph obtained from the transformation described in Construction 13. Then, there exists a configuration  $c$  of  $\widehat{G}$  such that  $\widehat{G} \circ c$  does not admit an  $s$ - $t$ -path if and only if there exists a configuration  $c'$  of  $\widehat{G}'$  such that  $\widehat{G}' \circ c'$  does not admit an  $s$ - $t$ -path.*

*Proof.* ( $\Rightarrow$ ): Let  $c$  be a configuration of  $\widehat{G}$  such that  $\widehat{G} \circ c$  does not admit an  $s$ - $t$ -path. We define  $c'$  to be an extension of  $c$  to  $\widehat{G}'$ , where for each  $i \in [2]$ , we set  $c'(S_i) := c(S)$ .

To obtain a contradiction, suppose that  $\widehat{G}' \circ c'$  admits an  $s$ - $t$ -path. Observe that every path in  $\widehat{G}' \circ c'$  which does not include the new vertex  $w$  must also exist in  $\widehat{G} \circ c$ . Hence, any  $s$ - $t$ -path in  $\widehat{G}' \circ c'$  must pass through vertex  $w$ .

If  $c(S) = 1$ , then both  $c'(S_1)$  and  $c'(S_2)$  are in the off-position, so the subpath  $v_1$ - $w$ - $v_2$  must be part of the  $s$ - $t$ -path in  $\widehat{G}' \circ c'$ . However, since  $c(S) = 1$ , the edge  $v_1v_2$  is present in  $\widehat{G} \circ c$ , allowing us to replace the subpath  $v_1$ - $w$ - $v_2$  with the edge  $v_1v_2$ , resulting in an  $s$ - $t$ -path in  $\widehat{G} \circ c$ , a contradiction.

The case for  $c(S) = 2$  is analogous: we obtain an  $s$ - $t$ -path in  $\widehat{G} \circ c$  by replacing the subpath  $u_1$ - $w$ - $u_2$  with the edge  $u_1u_2$ , again contradicting the assumption. Hence,  $\widehat{G}' \circ c'$  does not admit an  $s$ - $t$  path.

( $\Leftarrow$ ): Let  $c'$  be a configuration of  $\widehat{G}'$  such that  $\widehat{G}' \circ c'$  does not admit an  $s$ - $t$ -path. We define  $c$  to be an extension of  $c'$  to  $\widehat{G}$ , where switch  $S$  evaluates to  $c(S) := \min_{i \in [2]} \{c'(S_i)\}$ .

Suppose for contradiction that  $\widehat{G} \circ c$  admits an  $s$ - $t$ -path. Observe that any path not using an edge contributed by switch  $S$  must also exist in  $\widehat{G}' \circ c'$ . Hence, any  $s$ - $t$ -path in  $\widehat{G} \circ c$  must pass through an edge provided by switch  $S$ .

If  $c(S) = 1$ , then the  $s$ - $t$ -path in  $\widehat{G} \circ c$  must include the edge  $v_1v_2$ . However, since  $c(S) = 1$ , either  $c'(S_1)$  or  $c'(S_2)$  are in the off-position, adding edges  $v_1w$  and  $v_2w$  to  $\widehat{G}' \circ c'$ . Hence, edge  $v_1v_2$  can be replaced by the subpath  $v_1$ - $w$ - $v_2$ , resulting in an  $s$ - $t$ -path in  $\widehat{G}' \circ c'$ , a contradiction.

If  $c(S) = 2$ , then the  $s$ - $t$ -path in  $\widehat{G} \circ c$  includes edge  $u_1u_2$ . In that case, both  $c'(S_1)$  and  $c'(S_2)$  are in the on-position, adding edges  $u_1w$  and  $u_2w$  to  $\widehat{G}' \circ c'$ . Therefore, replacing edge  $u_1u_2$  by the subpath  $u_1$ - $w$ - $u_2$  results in an  $s$ - $t$ -path in  $\widehat{G}' \circ c'$ , again contradicting the assumption. Hence,  $\widehat{G} \circ c$  does not admit an  $s$ - $t$ -path.  $\square$

In Theorem 25, we established that  $\text{STDISCON-SWITCHING}$  is NP-hard even when all switches are  $\{1, 1\}$ -switches. By applying the transformation from Construction 13 to each switch in the switch graph consisting of two edges, and invoking Lemma 13, we obtain a correct reduction to  $\text{STDISCON-SWITCHING}$  restricted to  $\{2, 1\}$ -star switches.

It is important to note, however, that this transformation no longer preserves the structure of the union graph: in particular, the resulting union graph is not necessarily series-parallel, and we lose any guarantees regarding planarity or bounded treewidth. Nevertheless, we obtain the following intractability result for  $\text{STDISCON-SWITCHING}$ :

**Theorem 28.**  $\text{STDISCON-SWITCHING}$  is  $NP$ -hard, even if all switches are  $\{2, 1\}$ -star switches.

For the weighted variant of the problem, we need to carefully analyze the impact of the transformation described in Construction 13 on the number of switches set to the on-position. When the  $\{1, 1\}$ -switch  $S$  in  $\widehat{G}$  is set to the on-position, it contributes a weight of one. As previously discussed, this corresponds in  $\widehat{G}'$  to setting both  $S_1$  and  $S_2$  to the on-position, which collectively contribute a weight of two. Consequently, when applying the transformation, the activation budget must be increased by 1 in the attempt to preserve correctness.

However, if  $S$  is set to the off-position in  $\widehat{G}$ , then in  $\widehat{G}'$  one could potentially activate more switches elsewhere due to the increased budget, invalidating the reduction. To circumvent this issue, we need to ensure that the transformation is applied to every single switch. This ensures that each switch set to the on-position in  $\widehat{G}$  corresponds to two switches set to the on-position in  $\widehat{G}'$ . By doubling the activation budget, we obtain a correct parameterized reduction to  $\text{ONOFF-STDISCON-SWITCHING}$  restricted to  $\{2, 1\}$ -star switches.

Note, however, that Construction 13 applies only to  $\{1, 1\}$ -switches that contain two edges. Nonetheless, static edges can safely be contracted without affecting the existence of an  $s$ - $t$ -path, and empty switch positions can be filled with an edge between two fresh vertices that remain disconnected from the rest of the switch graph. Thus, we obtain the following hardness result:

**Theorem 29.**  $\text{ONOFF-STDISCON-SWITCHING}$  is  $W[1]$ -hard parameterized by  $k$ , even if all switches are  $\{2, 1\}$ -star switches. Furthermore, assuming the  $ETH$ , the problem cannot be solved in time  $f(k)n^{o(\frac{k}{\log k})}$ , where  $f$  is an arbitrary computable function and  $n$  is the number of switches in the switch graph.

*Proof.* We begin the proof by providing a parameterized reduction, with the activation budget as the parameter, from an instance  $(\widehat{H}, \ell)$  of  $\text{ONOFF-STDISCON-SWITCHING}$  restricted to  $\{1, 1\}$ -switches to an instance  $(\widehat{G}, k)$  restricted to  $\{2, 1\}$ -star switches.

Similar to the proof of Theorem 19, we may assume that  $\widehat{H}$  contains no static edges, as these can be contracted without affecting connectivity. Moreover, we may assume that each switch position contains one edge. Otherwise, we introduce two fresh vertices  $u$  and  $v$ , and insert the edge  $uv$  into all empty positions. Since  $u$  and  $v$  are isolated from the rest of the graph, this modification does not affect the connectivity between the original vertices.

The reduction itself is as follows:  $\widehat{G}$  is constructed by applying the transformation described in Construction 13 to each switch in  $\widehat{H}$ . Furthermore, we set the new activation budget  $k$  to  $2\ell$ .

Observe that the reduction is computable in polynomial time and the parameter  $k$  can be bounded by some computable function of  $\ell$ . To prove that the reduction is correct, we need to show that there exists a configuration  $c'$  of  $\widehat{H}$  such that  $\widehat{H} \circ c'$  does not admit an  $s$ - $t$ -path and  $|c'|_2 \leq \ell$  if and only if there exists a configuration  $c$  of  $\widehat{G}$  such that  $\widehat{G} \circ c$  does not admit an  $s$ - $t$ -path and  $|c|_2 \leq k$ . We show both directions independently:

( $\Rightarrow$ ): Let  $c'$  be a configuration of  $\widehat{H}$  such that  $\widehat{H} \circ c'$  does not admit an  $s$ - $t$ -path and  $|c'|_2 \leq \ell$ . We construct a configuration  $c$  of  $\widehat{G}$  such that for each switch  $S \in \mathcal{S}(\widehat{H})$  and  $i \in [2]$ , we have  $c(S_i) := c'(S)$ .

By construction, we have  $|c|_2 = 2|c'|_2$  and thus  $|c|_2 \leq 2\ell = k$ . We show that  $\widehat{G} \circ c$  does not admit an  $s$ - $t$ -path. Suppose, for contradiction, that such a path exists in  $\widehat{G} \circ c$ .

Since original vertices of  $\widehat{H}$  in  $\widehat{G}$  are exclusively adjacent to the auxiliary vertices introduced in the transformations and vice versa, any  $s$ - $t$ -path in  $\widehat{G} \circ c$  must alternate between original vertices of  $\widehat{H}$  and fresh vertices. Additionally, the path must start and finish with an original vertex of  $\widehat{H}$  (the source and target vertex, respectively).

Now, take some fresh vertex  $w$  — introduced by the transformation in Construction 13 of some switch  $S \in \mathcal{S}(\widehat{H})$  — which the  $s$ - $t$ -path passes through. By the argument in Lemma 13 ( $\Rightarrow$ ),  $w$  is preceded and followed by two original vertices  $u$  and  $v$  where  $S[c'(S)] = \{uv\}$ . Since the edge  $uv$  is present in  $\widehat{H} \circ c'$ , we can replace the subpath  $u$ - $w$ - $v$  in the  $s$ - $t$ -path with the edge  $uv$ . We repeat this procedure for all fresh vertices in the path to obtain an  $s$ - $t$ -path in  $\widehat{H} \circ c'$ , resulting in a contradiction.

Thus,  $\widehat{G} \circ c$  does not admit an  $s$ - $t$  path.

( $\Leftarrow$ ): Let  $c$  be a configuration of  $\widehat{G}$  such that  $\widehat{G} \circ c$  does not admit an  $s$ - $t$ -path and  $|c|_2 \leq k$ . We construct a configuration  $c'$  of  $\widehat{H}$  such that for each switch  $S \in \mathcal{S}(\widehat{H})$ , we have  $c'(S) := \min_{i \in [2]} \{c(S_i)\}$ .

By construction, we have  $|c'|_2 \leq \frac{1}{2}|c|_2 \leq \frac{1}{2}k = \ell$ . We show that  $\widehat{H} \circ c'$  does not admit an  $s$ - $t$ -path. Suppose, for contradiction, that such a path exists in  $\widehat{H} \circ c'$ . Now, let  $uv$  be some arbitrary edge on this  $s$ - $t$ -path activated by a switch  $S \in \mathcal{S}(\widehat{H})$ .

By the argument in Lemma 13 ( $\Leftarrow$ ), we know that the subpath  $u$ - $w$ - $v$  must exist in  $\widehat{G} \circ c$ , where  $w$  is the fresh vertex introduced in the transformation of  $S$ . Hence, we can replace  $uv$  with the subpath  $u$ - $w$ - $v$  in the  $s$ - $t$ -path. If we repeat this procedure for all edges in the  $s$ - $t$ -path, we obtain an  $s$ - $t$ -path in  $\widehat{G} \circ c$ , contradicting the assumption that  $\widehat{G} \circ c$  does not admit an  $s$ - $t$ -path.

Thus,  $\widehat{H} \circ c'$  does not admit an  $s$ - $t$ -path.

Since Theorem 27 establishes W[1]-hardness parameterized by  $\ell$  restricted to  $\{1, 1\}$ -switches, we obtain W[1]-hardness parameterized by  $k$  restricted to  $\{2, 1\}$ -star switches as well.

Furthermore, let  $(\widehat{H}, \ell)$  be an arbitrary instance of ONOFF-STDISCON-SWITCHING restricted to  $\{1, 1\}$ -switches, and let  $(\widehat{G}, k)$  be the instance restricted to  $\{2, 1\}$ -star switches produced by the reduction described above. Suppose that ONOFF-STDISCON-SWITCHING can be solved in time  $f(k)n^{o(\frac{k}{\log k})}$  restricted to  $\{2, 1\}$ -star switches, where  $f$  is an arbitrary computable function and  $n$  is the number of switches in the switch graph. Thus, to decide whether  $(\widehat{H}, \ell)$  is a yes-instance, we can first reduce it to the instance  $(\widehat{G}, k)$  and solve it in the aforementioned time.

Let  $m$  be the number of switches in  $\widehat{H}$ . By construction, we have that  $n = 2m$  and  $k = 2\ell$ . Furthermore, the reduction can be computed in linear time, that is, in time  $\mathcal{O}(m)$ . In combination, we can therefore decide whether  $(\widehat{H}, \ell)$  is a yes-instance in time  $\mathcal{O}(m) + f(2\ell) \cdot (2m)^{o(\frac{2\ell}{\log(2\ell)})}$ , which is less than or equal to  $g(\ell)m^{o(\frac{\ell}{\log \ell})}$  for some computable function  $g$ .

According to Theorem 27, it is impossible to solve ONOFF-STDISCON-SWITCHING restricted to  $\{1, 1\}$ -switches in time  $g(\ell)m^{o(\frac{\ell}{\log \ell})}$  unless the ETH fails. Hence, we conclude that ONOFF-STDISCON-SWITCHING restricted to  $\{2, 1\}$ -star switches cannot be solved in time  $f(k)n^{o(\frac{k}{\log k})}$  unless the ETH fails.  $\square$

**Discussion** In this chapter, we have established the intractability of STDISCON-SWITCHING and its weighted variant, even under strong structural restrictions such as  $\{1, 1\}$ - or  $\{2, 1\}$ -star switches. Our reductions highlight the close relationship between the complexity of  $s$ - $t$ -connectivity and disconnectivity switching.

A central open question is whether STDISCON-SWITCHING admits a dichotomy result analogous to Theorem 6 for STCON-SWITCHING. In particular: Is STDISCON-SWITCHING solvable in polynomial time for  $\{1\}^*$ -switches whose switch-induced graphs are complete multipartite? Does the problem become NP-hard as soon as the admissible class of switch-induced graphs contains a non-complete multipartite graph? And can such a dichotomy be extended beyond switches with at most one edge per position?

# FPT Algorithms via Dynamic Programming

So far, we have established that the switching problem is computationally hard in many cases, even under strong structural restrictions on the switches and the union graph. For GCON-SWITCHING, in particular, we showed that the problem remains NP-hard for  $\{2, 1\}$ -star switches on a planar union graph of maximum switch degree three (i.e., every vertex participates in at most three switches; see Theorem 24). This shows that parameterizing by switch size and switch degree alone does not yield a FPT algorithm, even when restricting to planar union graphs with connected switches.

A more promising direction is to move from planarity to tree-likeness, measured by the treewidth of the union graph. Treewidth has proven to be a powerful parameter in graph algorithms, often turning otherwise intractable problems into tractable ones via dynamic programming over tree decompositions. We provide the necessary preliminaries in Section 9.1. In Section 9.2, we develop a FPT algorithm for WEIGHTED-GCON-SWITCHING, parameterized jointly by the treewidth of the union graph, the maximum switch degree, and the maximum number of switch positions. In Section 9.3, we show how the same approach can be adapted to WEIGHTED-STCON-SWITCHING and WEIGHTED-STDISCON-SWITCHING.

Although we shifted our focus from planarity to treewidth, planarity of the union graph might still be leveraged in a win-win approach to drop the treewidth parameter. The idea is that if the treewidth is large, we can immediately conclude that the instance is a no-instance, while if the treewidth is small, we can apply a dynamic programming algorithm as above. Inspired by the potential applications for ROBOT CONFIGURATION 18, we explore in Section 9.4 a potential application of this idea for a version of WEIGHTED-GCON-SWITCHING with an additional parameter bounding the radius of the configuration graph.

## 9.1 Preliminaries

A *tree decomposition* of a simple graph  $G$  is a pair  $\mathcal{T} = (T, X)$  where  $T$  is a tree and  $X: V(T) \rightarrow 2^{V(G)}$  is a function assigning each node  $t \in V(T)$  a subset  $X(t) \subseteq V(G)$ , called a *bag*, such that the following three conditions hold:

1. **Vertex coverage:** Every vertex  $v \in V(G)$  is contained in at least one bag, i.e.,  $\bigcup_{t \in V(T)} X(t) = V(G)$ .
2. **Edge coverage:** For every edge  $uv \in E(G)$ , there exists a node  $t \in V(T)$  such that both  $\{u, v\} \subseteq X(t)$ .
3. **Coherence:** For every vertex  $v \in V(G)$ , the set of nodes  $\{t \in V(T) \mid v \in X(t)\}$  induces a connected subtree of  $T$ .

The *width* of the tree decomposition is  $\max_{t \in V(T)} \{|X(t)| - 1\}$ , and the *treewidth* of  $G$  is the minimum width over all tree decompositions of  $G$ , denoted  $\text{tw}(G)$ . Given an integer  $k$ , there exists an algorithm with running time  $2^{\mathcal{O}(k)}|V(G)|$  that either constructs a tree decomposition of  $G$  of width at most  $2k + 1$  and at most  $|V(G)|$  nodes, or concludes that  $\text{tw}(G) > k$  [12]. As a consequence, by running this algorithm for increasing values of  $k$ , we can obtain a tree decomposition of width at most  $2 \cdot \text{tw}(G) + 1$  in FPT time.

A tree decomposition  $\mathcal{T} = (T, X)$  is *nice* if  $T$  is a rooted binary tree and each node  $t \in V(T)$  is one of the following types:

- **Leaf node:**  $t$  is a leaf, meaning  $\deg(t) \leq 1$ , and  $X(t) = \emptyset$ .
- **Introduce node:**  $t$  has exactly one child  $t'$  such that  $X(t) = X(t') \cup \{v\}$  for some vertex  $v \in V(G) \setminus X(t')$ . In other words,  $t$  introduces a new vertex  $v$  into the bag.
- **Forget node:**  $t$  has exactly one child  $t'$  such that  $X(t) = X(t') \setminus \{v\}$  for some vertex  $v \in X(t')$ . In other words,  $t$  forgets a vertex  $v$  from the bag.
- **Join node:**  $t$  has two children  $t_1$  and  $t_2$  such that  $X(t) = X(t_1) = X(t_2)$ .

Furthermore, for node  $t \in V(T)$ , let  $V(t)$  be the union of all bags present in the subtree of  $T$  rooted at  $t$ .

Given an arbitrary tree decomposition  $\mathcal{T} = (T, X)$  of some graph  $G$  with a width of at most  $k$ , we can construct a nice tree decomposition of  $G$  with a width of at most  $k$  in time  $\mathcal{O}(k^2 \cdot \max\{|V(T)|, |V(G)|\})$  containing at most  $\mathcal{O}(k|V(G)|)$  nodes [4, Lemma 7.4]. Consequently, since a tree decomposition of  $G$  of width  $\mathcal{O}(\text{tw}(G))$  can be computed in FPT time, we can also compute a nice tree decomposition within the same time bounds.

*Dynamic programming* is a method for solving optimization or decision problems by recursively breaking them down into simpler subproblems. It exploits the fact that

many subproblems arise multiple times during the computation. Instead of recomputing their solutions, dynamic programming stores the results of solved subproblems in a table and reuses them when needed. This technique can significantly reduce the overall time complexity of the algorithm.

Applying dynamic programming on a nice tree decomposition is a widely used technique to design FPT algorithms for graph problems parameterized by treewidth. It takes advantage of the fact that bags act as separators of the input graph. In particular, given a nice tree decomposition  $\mathcal{T} = (T, X)$  of width  $k$  of a graph  $G$ , and a node  $t \in V(T)$ , there are no edges between the vertices in  $V(t) \setminus X(t)$  and those in  $V(G) \setminus V(t)$ . This means that the vertices in  $V(t)$  are connected to the rest of the graph only through the vertices in  $X(t)$ .

This separator property allows us to compute partial solutions for the subgraph  $G[V(t)]$  at each node  $t \in V(T)$ . Because the interaction between  $G[V(t)]$  and the remainder of the graph exclusively occurs via the bag  $X(t)$ , it is sufficient to retain the behavior of the partial solution on  $X(t)$  and discarding all internal details. Therefore, each partial solution can be stored in a dynamic programming table indexed by  $t$  and the *state* of the partial solution restricted to vertices from bag  $X(t)$ . For example, in the context of our problem, this state includes the set of vertex pairs in  $X(t)$  reachable from one another in subgraph  $G[V(t)]$ , as well as the configuration of the switches incident to vertices in  $X(t)$ .

The algorithm processes the tree decomposition in a bottom-up fashion. At each node  $t \in V(T)$ , it computes the partial solutions based on the node type and the table entries of its child(ren). Each entry must be computable in time  $g(k) \cdot |V(G)|^{\mathcal{O}(1)}$  for some computable function  $g$ . Since the number of possible states per bag depends only on  $k$ , and the total number of nodes in the decomposition is  $\mathcal{O}(k \cdot |V(G)|)$ , the overall runtime becomes  $f(k) \cdot |V(G)|^{\mathcal{O}(1)}$ , where  $f$  is a computable function.

The solution to the original problem can be derived from the partial solutions computed at the root  $r(T)$  of the tree decomposition, as  $V(r(T)) = V(G)$ . By first computing a nice tree decomposition of width  $\mathcal{O}(\text{tw}(G))$  in FPT time and then solving the problem via dynamic programming, we obtain an FPT algorithm for the problem parameterized by treewidth.

In our dynamic-programming algorithm for switching problems, we will be working a lot with reachability relations, or in more general, equivalence relations. The following lemma states an important property of equivalence relations which we will use in the analysis of our algorithm.

**Lemma 14.** *Let  $X_1$  and  $X_2$  be sets, and let  $R_1 \subseteq X_1^2$  and  $R_2 \subseteq X_2^2$  be equivalence relations. Then for any  $Y \supseteq X_1 \cap X_2$ , the following equality holds:*

$$(R_1 \cup R_2)^+ \cap Y^2 = ((R_1 \cup R_2) \cap Y^2)^+$$

*Proof.* Let  $(x, y) \in (R_1 \cup R_2)^+ \cap Y^2$ . This implies that  $x, y \in Y$  and that there exists a shortest sequence of vertices  $x = x_1, x_2, \dots, x_k = y$  such that  $(x_i, x_{i+1}) \in R_1 \cup R_2$  for all  $i \in [k - 1]$ .

We claim that  $x_i \in Y$  for all  $i \in [k]$ . Suppose, for contradiction, that there exists some  $x_i \notin Y$  for  $i \in [2, k - 1]$ . Then  $x_i$  must belong to either  $X_1 \setminus Y$  or  $X_2 \setminus Y$ . Without loss of generality, assume  $x_i \in X_1 \setminus Y$ . Since  $x_i \notin X_2$ , it follows that both  $(x_{i-1}, x_i)$  and  $(x_i, x_{i+1})$  must be in  $R_1$ , as they cannot be in  $R_2$ . Because  $R_1$  is transitive, we have  $(x_{i-1}, x_{i+1}) \in R_1$ . Thus, we can remove  $x_i$  from the sequence while preserving all required conditions, contradicting the assumption that the sequence was shortest. Therefore,  $x_i \in Y$  for all  $i \in [k]$ .

Consequently, for all  $i \in [k - 1]$ , we have  $(x_i, x_{i+1}) \in (R_1 \cup R_2) \cap Y^2$ . By transitivity, this implies that  $(x, y) \in ((R_1 \cup R_2) \cap Y^2)^+$ . Since this holds for all  $(x, y) \in (R_1 \cup R_2)^+ \cap Y^2$ , we conclude that

$$(R_1 \cup R_2)^+ \cap Y^2 \subseteq ((R_1 \cup R_2) \cap Y^2)^+. \quad (9.1)$$

Conversely, since  $((R_1 \cup R_2) \cap Y^2)^+$  is a subset of both  $(R_1 \cup R_2)^+$  and  $(Y^2)^+ = Y^2$ , it follows that

$$((R_1 \cup R_2) \cap Y^2)^+ \subseteq (R_1 \cup R_2)^+ \cap Y^2. \quad (9.2)$$

Together, Eq. (9.1) and Eq. (9.2) establish the desired equality.  $\square$

In Chapter 2, we have already defined  $\mathcal{S}_{\hat{G}}(v)$  as the neighborhood of switches of vertex  $v$  in switch graph  $\hat{G}$ . For the analysis of the algorithm, we omit the subscript for better readability and always assume that the switch graph in question is the one from the input. Moreover, we extend the definition to vertex sets, allowing us to obtain all incident switches of a given set of vertices.

## 9.2 Algorithm for Global Connectivity

In this section, we show that WEIGHTED-GCON-SWITCHING is in FPT for switch graphs where each switch induced graph is connected. Our algorithm is parameterized by the treewidth of the union graph, allowing us to define a dynamic programming routine on a nice tree decomposition of this graph. Additionally, it can also be parameterized by the maximum switch degree over all vertices and the maximum number of positions over all switches.

To illustrate the core idea of the algorithm, consider a switch graph  $\hat{G}$  where for each switch  $S \in \mathcal{S}(\hat{G})$ , the switch induced graph  $G(S)$  is connected, along with a nice tree decomposition  $\mathcal{T} = (T, X)$  of the union graph  $G(\hat{G})$ . For some node  $t \in V(T)$ , recall that the bag  $X(t)$  separates  $V(t) \setminus X(t)$  from the rest of the graph. Since each switch

induces a connected subgraph, every switch incident to vertices in  $V(t)$  but not in  $X(t)$  cannot be incident to vertices outside of  $V(t)$ . This property is crucial for our dynamic programming algorithm, enabling us to compute partial solutions for subgraph  $\widehat{G}[V(t)]$  at each node  $t \in V(T)$ .

The intuition behind this becomes clearer when we consider the following: let  $c_1$  and  $c_2$  be two configurations of  $\widehat{G}$  that are identical on all switches incident to vertices in  $X(t)$ , i.e.,  $c_1|_{\mathcal{S}(X(t))} = c_2|_{\mathcal{S}(X(t))}$ , and assume that both  $\widehat{G} \circ c_1$  and  $\widehat{G} \circ c_2$  are globally connected. Furthermore, suppose that the pairs of reachable vertices within  $X(t)$  are the same in both  $\widehat{G}[V(t)] \circ c_1$  and  $\widehat{G}[V(t)] \circ c_2$ , that is,  $R(\widehat{G}[V(t)] \circ c_1) \cap X(t)^2 = R(\widehat{G}[V(t)] \circ c_2) \cap X(t)^2$ .

Let  $w : \mathcal{S}(\widehat{G}) \times \mathbb{N}_{>0} \rightarrow \mathbb{R}$  be a weight function over switch positions. Suppose that the restricted weight of  $c_1$  is strictly smaller than that of  $c_2$  on the switches incident to vertices in  $V(t)$ , that is,  $w(c_1|_{\mathcal{S}(V(t))}) < w(c_2|_{\mathcal{S}(V(t))})$ . We claim that  $c_2$  is suboptimal.

If this claim holds, then every optimal configuration of  $\widehat{G}$  must also be optimal for the subgraph  $\widehat{G}[V(t)]$ , when restricted to configurations that agree on the switches incident to  $X(t)$ , and induce the same reachability relation on  $X(t)$ . This observation is crucial for our dynamic programming algorithm: it ensures that we can compute optimal partial solutions for  $\widehat{G}[V(t)]$  at node  $t$  independently of the rest of the graph.

To prove the above claim, we construct a new configuration  $c'_2$  from  $c_2$  by modifying the positions of switches incident to vertices in  $V(t)$  to match those in  $c_1$ . Formally, we define  $c'_2$  as follows:

$$c'_2 := c_2|_{\mathcal{S}(V(\widehat{G}) \setminus V(t))} \cup c_1|_{\mathcal{S}(V(t))}.$$

This means that  $\widehat{G}[V(t)] \circ c'_2$  is identical to  $\widehat{G}[V(t)] \circ c_1$ , and, by assumption, the reachability relations on  $X(t)$  coincide with those of  $\widehat{G}[V(t)] \circ c_2$ .

Observe that within  $\widehat{G}[V(t)] \circ c_1$ , every vertex in  $V(t) \setminus X(t)$  must be reachable from some vertex in  $X(t)$ . Any unreachable vertex would be disconnected from the rest of the graph, as there are no edges between  $V(t) \setminus X(t)$  and vertices outside of  $V(t)$  in any configuration of  $\widehat{G}$ , contradicting the assumption that  $\widehat{G} \circ c_1$  is globally connected. By construction of  $c'_2$ , the same can be said about  $\widehat{G}[V(t)] \circ c'_2$ .

Let us now consider the connectivity of  $\widehat{G} \circ c_2$ . Since  $\widehat{G} \circ c_1$  is globally connected and  $X(t)$  separates  $V(t)$  from the rest of the graph, every vertex in  $V(t) \setminus X(t)$  must be reachable by some vertex in  $X(t)$  within  $\widehat{G}[V(t)] \circ c_1$ . The same must hold for  $\widehat{G}[V(t)] \circ c'_2$ , as it is identical.

Since  $\widehat{G} \circ c_2$  is globally connected, the edges outside  $\widehat{G}[V(t)] \circ c_2$  must provide the missing connections among the pairs in  $X(t)$  and also link the vertices outside  $V(t)$  to those in  $X(t)$ . These edges remain unchanged in  $\widehat{G}[V(t)] \circ c'_2$ , implying that  $\widehat{G} \circ c'_2$  is also globally connected. Moreover,  $w(c_1|_{\mathcal{S}(V(t))}) < w(c_2|_{\mathcal{S}(V(t))})$  implies that  $w(c'_2) < w(c_2)$ , proving that  $c_2$  is not optimal.

What this shows is that, given a partial configuration  $c_t \in \mathcal{C}(\widehat{G}[X(t)])$  of switches incident to vertices in  $X(t)$ , and an equivalence relation  $R_t \subseteq X(t)^2$ , every globally

connected configuration  $c$  of  $\widehat{G}$  satisfying  $c|_{\mathcal{S}(X(t))} = c_t$  and  $R(\widehat{G}[V(t)] \circ c) \cap X(t)^2 = R_t$  must also ensure that every vertex in  $V(t) \setminus X(t)$  is reachable from some vertex in  $X(t)$  in  $\widehat{G}[V(t)] \circ c$ . Among such configurations, we can safely replace the settings of switches incident to vertices in  $V(t)$  with those from another configuration satisfying the same  $c_t$  and  $R_t$ , without affecting global connectivity. Moreover, every minimum-weight configuration under these constraints must also minimize the weight over the switches in  $V(t)$  among all such valid configurations.

### 9.2.1 Formal Description of the Algorithm

The above observations are the key to our dynamic programming algorithm. Given a switch graph  $\widehat{G}$  and a nice tree decomposition  $\mathcal{T} = (T, X)$  of the union graph  $G(\widehat{G})$ , we define a dynamic programming table  $W$  with following entries:

**Definition 1.** For each node  $t \in V(T)$ , partial configuration  $c_t \in \mathcal{C}(\widehat{G}[X(t)])$  of switches incident to  $X(t)$ , and equivalence relation  $R_t \subseteq X(t)^2$ , let  $W[t, c_t, R_t]$  be the minimum weight over all configurations  $c \in \mathcal{C}(\widehat{G}[V(t)])$  such that:

- (1) configuration  $c$  restricted to switches in  $X(t)$  is equal to  $c_t$ , that is,  $c|_{\mathcal{S}(X(t))} = c_t$ ,
- (2) the reachability relation of  $\widehat{G}[V(t)] \circ c$  restricted to vertices in  $X(t)$  is equal to  $R_t$ , that is,  $R(\widehat{G}[V(t)] \circ c) \cap X(t)^2 = R_t$ , and
- (3) every vertex in  $V(t)$  is reachable from some vertex in  $X(t)$  in  $\widehat{G}[V(t)] \circ c$ , that is,  $\forall v \in V(t) : R(\widehat{G}[V(t)] \circ c)[v] \cap X(t) \neq \emptyset$ .

If no such configuration exists, we assign the value  $\infty$  to  $W[t, c_t, R_t]$ , following the standard convention of taking the minimum over an empty set. As a consequence, for any non-empty switch graph  $\widehat{G}$ , we have  $W[r(T), \emptyset, \emptyset] = \infty$  at the root node of the nice tree decomposition, since condition (3) cannot be satisfied.

Thus, to determine the minimum weight over all connected configurations of  $\widehat{G}$ , we cannot use the entries of the root node  $r(T)$  directly. Instead, we consider the child node  $r'$ . Since  $r(T)$  must be a forget node, we have  $V(r') = V(\widehat{G})$  and  $X(r') = \{v\}$  for some vertex  $v \in V(\widehat{G})$ . For every configuration  $c_v \in \mathcal{C}(\widehat{G}[\{v\}])$  of the switches incident to  $v$ , the value  $W[r', c_v, \{(v, v)\}]$  corresponds to the minimum weight of configuration  $c \in \mathcal{C}(\widehat{G})$  such that  $c|_{\mathcal{S}(v)} = c_v$  and every vertex in  $\widehat{G} \circ c$  is reachable from  $v$ , thereby ensuring global connectivity. By iterating over all configurations of the switches incident to  $v$ , we can find the minimum weight over all globally connected configurations of  $\widehat{G}$ , formally stated in the following lemma:

**Lemma 15.** *The minimum weight over all connected configurations of  $\widehat{G}$  is given by*

$$\min_{c_v \in \mathcal{C}(\widehat{G}[\{v\}])} W[r', c_v, \{(v, v)\}],$$

where  $r'$  is the child of the root node  $r(T)$  in the nice tree decomposition such that  $X(r') = \{v\}$ .

We now describe how to compute the entries of the dynamic programming table  $W$  for each node  $t \in V(T)$ , depending on its type. For all node types except leaf nodes, the values are defined recursively based on the entries at the children of  $t$ . In each case, we prove that the resulting value satisfies the invariant defined in Definition 1.

**Leaf node.** If  $t$  is a leaf node, then its bag  $X(t)$  is empty. Consequently, the vertex-induced subswitch graph  $\widehat{G}[X(t)]$  is the empty switch graph, which only admits the empty configuration  $\emptyset$ . Furthermore, the only equivalence relation on  $X(t)$  is the empty relation. Since  $\widehat{G}[V(t)]$  is also the empty switch graph, all three conditions of Definition 1 are trivially satisfied.

**Lemma 16.** *If  $t$  is a leaf node, then the only entry in the dynamic programming table for  $t$  is*

$$W[t, \emptyset, \emptyset] := 0,$$

which satisfies the invariant specified in Definition 1.

**Introduce node.** If  $t$  is an introduce node with child  $t'$ , then a new vertex  $v$  is added to the bag. This may introduce new switches and add new edges to existing switches, all of which are incident to  $v$ . Crucially, all such edges lie between  $v$  and vertices in bag  $X(t')$ .

Intuitively, to compute the value of  $W[t, c_t, R_t]$ , we consider entries at child node  $t'$  whose configuration on switches incident to  $X(t')$  matches  $c_t$ . Among these, we select those whose reachability relation on  $X(t')$ , when extended by the new active edges incident to  $v$  in  $c_t$ , results in  $R_t$ . Taking the minimum of the corresponding values and adding the weight of all new switches incident to  $v$ , configured by  $c_t$ , yields the value of  $W[t, c_t, R_t]$ .

**Lemma 17.** *If  $t$  is an introduce node with child  $t'$ , where  $X(t) = X(t') \cup \{v\}$  for some vertex  $v \notin X(t')$ , then for every configuration  $c_t \in \mathcal{C}(\widehat{G}[X(t)])$  of switches incident to  $X(t)$  and equivalence relation  $R_t \subseteq X(t)^2$ , the following recursive definition of  $W[t, c_t, R_t]$  satisfies the invariant in Definition 1:*

$$W[t, c_t, R_t] := \min_{R_{t'} \subseteq X(t')^2 \text{ valid for } c_t, R_t} \left\{ W[t', c_t|_{S(X(t'))}, R_{t'}] \right\} + w(c_t|_{S(v)}),$$

where  $R_{t'}$  is an equivalence relation such that adding the active edges incident to  $v$  in  $\widehat{G}[X(t)] \circ c_t$  to  $R_{t'}$ , and taking the transitive closure, yields  $R_t$ , or equivalently,

$$R_t = (R_{t'} \cap R(\widehat{G}[X(t)] \circ c_t))^+.$$

*Proof.* Let  $\hat{c}$  be a configuration of  $\widehat{G}[V(t)]$  for which the minimum weight is attained in Definition [1](#) for  $W[t, c_t, R_t]$ . Then  $c := \hat{c}|_{\mathcal{S}(V(t'))}$  is a configuration of  $\widehat{G}[V(t')]$ . For  $c_{t'} = c|_{\mathcal{S}(X(t'))}$  and  $R_{t'} = R(\widehat{G}[V(t')] \circ c) \cap X(t')^2$ , configuration  $c$  clearly fulfills condition (1) and (2) of Definition [1](#) on  $W[t', c_{t'}, R_{t'}]$ . Condition (3) is also fulfilled since  $v$  is only adjacent to vertices of  $X(t')$  in  $\widehat{G}[V(t)] \circ \hat{c}$ , thus for any vertex  $u \in V(t')$  there still remains a path to a vertex of  $X(t')$  in  $\widehat{G}[V(t')] \circ c$ . Consequently,  $c$  is one of the configurations considered in the recursive definition of  $W[t', c_{t'}, R_{t'}]$  and we have that:

$$\begin{aligned} W[t', c_{t'}, R_{t'}] &\leq w(c) = w(\hat{c}|_{\mathcal{S}(V(t'))}) = w(\hat{c}) - w(\hat{c}|_{\mathcal{S}(V(t)) \setminus \mathcal{S}(V(t'))}) \\ &= W[t, c_t, R_t] - w(c_t|_{\mathcal{S}(v) \setminus \mathcal{S}(X(t'))}) \end{aligned} \quad (1)$$

Moreover, condition (1) of  $\hat{c}$  in the definition of  $W[t, c_t, R_t]$  proves that:

$$c_{t'} = c|_{\mathcal{S}(X(t'))} = \hat{c}|_{\mathcal{S}(V(t'))}|_{\mathcal{S}(X(t'))} = \hat{c}|_{\mathcal{S}(X(t))}|_{\mathcal{S}(X(t'))} = c_t|_{\mathcal{S}(X(t'))}$$

Utilizing Lemma [14](#), condition (2) has the following implication:

$$\begin{aligned} R_t &= R(\widehat{G}[V(t)] \circ \hat{c}) \cap X(t)^2 = (R(\widehat{G}[V(t')] \circ c) \cup R(\widehat{G}[X(t)] \circ c_t))^+ \cap X(t)^2 \\ &= ((R(\widehat{G}[V(t')] \circ c) \cup R(\widehat{G}[X(t)] \circ c_t)) \cap X(t)^2)^+ \\ &= (R_{t'} \cup R(\widehat{G}[X(t)] \circ c_t))^+ \end{aligned}$$

Hence,  $c_{t'}$  together with  $R_{t'}$  is considered in the recursive definition of  $W[t, c_t, R_t]$ . In combination with Eq. [\(1\)](#), we then have that:

$$W[t, c_t, R_t] \geq \min_{R_{t'} \subseteq X(t')^2 \text{ valid for } c_t, R_t} \{W[t', c_t|_{\mathcal{S}(X(t'))}, R_{t'}] + w(c_t|_{\mathcal{S}(v) \setminus \mathcal{S}(X(t'))})\} \quad (2)$$

On the other hand, consider an arbitrary equivalence relation  $R_{t'} \subseteq X(t')^2$  such that  $(R_{t'} \cup R(\widehat{G}[X(t)] \circ c_t))^+ = R_t$ . Let  $\hat{c}$  be a configuration of  $\widehat{G}[V(t')]$  for which the minimum weight is attained in Definition [1](#) for  $W[t', c_t|_{\mathcal{S}(X(t'))}, R_{t'}]$ . We construct a configuration  $c$  of  $\widehat{G}[V(t)]$  such that for each switch  $S \in \mathcal{S}(V(t))$ :

$$c(S) := \begin{cases} c_t(S) & \text{if } S \in \mathcal{S}(X(t)) \\ \hat{c}(S) & \text{otherwise} \end{cases}$$

Observe that, due to the properties of  $\hat{c}$  in Definition [1](#) for  $W[t', c_t|_{\mathcal{S}(X(t'))}, R_{t'}]$ ,  $c$  must also fulfill conditions (1) and (3) in the definition of  $W[t, c_t, R_t]$ . By utilizing Lemma [14](#), we can also confirm property (2):

$$\begin{aligned} R_t &= (R_{t'} \cup R(\widehat{G}[X(t)] \circ c_t))^+ = (R(\widehat{G}[V(t')] \circ \hat{c}) \cup R(\widehat{G}[X(t)] \circ c_t))^+ \cap X(t)^2 \\ &= R(\widehat{G}[V(t)] \circ c) \cap X(t)^2 \end{aligned}$$

Hence,  $c$  is one of the configurations considered in the recursive definition of  $W[t, c_t, R_t]$ . This implies that:

$$W[t, c_t, R_t] \leq W[t', c_t|_{\mathcal{S}(X(t'))}, R_{t'}] + w(c_t|_{\mathcal{S}(v) \setminus \mathcal{S}(X(t'))}) \quad (3)$$

Since Eq. (3) holds for every equivalence relation  $R_{t'}$  such that  $(R_{t'} \cup R(\widehat{G}[X(t)] \circ c_t))^+ = R_t$ , we conclude:

$$W[t, c_t, R_t] \leq \min_{\substack{R_{t'} \subseteq X(t')^2 \\ \text{valid for } c_t, R_t}} \{W[t', c_t|_{\mathcal{S}(X(t'))}, R_{t'}] + w(c_t|_{\mathcal{S}(v) \setminus \mathcal{S}(X(t'))})\} \quad (4)$$

Concluding, Eq. (2) and Eq. (4) together prove the recursive definition in the lemma statement.  $\square$

**Forget node.** If  $t$  is a forget node with child  $t'$ , then some vertex  $v$  is removed from the bag. Since this vertex has no edges to vertices outside of  $V_t$  in the union graph, we must ensure that  $v$  is reachable from at least one vertex in  $X(t)$  before it is removed. Otherwise, the vertex, and possibly other vertices in  $V(t) \setminus X(t)$  connected to  $v$  but not to any other vertices in  $X(t)$ , become disconnected from the rest of the configured graph, preventing global connectivity.

Intuitively, to compute the value of  $W[t, c_t, R_t]$ , we consider entries at child node  $t'$  whose configuration on switches incident to  $X(t)$  matches  $c_t$ , and whose equivalence relation restricted to pairs within  $X(t)$  is equal to  $R_t$ . Among these, we select those where  $v$  is in relation to at least one vertex in  $X(t)$ . The minimum of these values corresponds to the value of  $W[t, c_t, R_t]$ .

**Lemma 18.** *If  $t$  is a forget node with child  $t'$ , where  $X(t) \cup \{v\} = X(t')$  for some vertex  $v \notin X(t)$ , then for every configuration  $c_t \in \mathcal{C}(\widehat{G}[X(t)])$  of switches incident to  $X(t)$  and equivalence relation  $R_t \subseteq X(t)^2$ , the following recursive definition of  $W[t, c_t, R_t]$  satisfies the invariant in Definition 1:*

$$W[t, c_t, R_t] := \min_{\substack{c_{t'} \in \mathcal{C}(\widehat{G}[X[t']]) \\ R_{t'} \subseteq X_{t'}^2 \\ \text{valid for } c_t, R_t}} \{W[t', c_{t'}, R_{t'}]\},$$

where

- (1)  $c_{t'}$  is an extension of  $c_t$ , that is,  $c_{t'}|_{\mathcal{S}(X(t))} = c_t$ ,
- (2)  $R_{t'}$  is identical to  $R_t$  on  $X(t)$ , that is,  $R_{t'} \cap X(t)^2 = R_t$ , and
- (3)  $v$  is in relation to at least one vertex in  $X(t)$  in the equivalence relation  $R_{t'}$ , that is,  $R_{t'}[v] \cap X(t) \neq \emptyset$ .

*Proof.* Let  $\hat{c} \in \mathcal{C}(\widehat{G}[V(t)])$  be a configuration for which the minimum weight is attained in Definition 1 for  $W[t, c_t, R_t]$ . Since  $V(t) = V(t')$ ,  $\hat{c}$  is also a valid configuration in  $\mathcal{C}(\widehat{G}[V(t')])$ . Define  $c_{t'} := \hat{c}|_{\mathcal{S}(X(t'))}$  and  $R_{t'} := R(\widehat{G}[V(t')] \circ \hat{c}) \cap X(t')^2$ . By construction,

$\hat{c}$  satisfies all three conditions of Definition 1 for  $W[t', c_{t'}, R_{t'}]$ , and hence is among the considered configurations. Therefore:

$$W[t', c_{t'}, R_{t'}] \leq w(\hat{c}) = W[t, c_t, R_t]. \quad (1)$$

Moreover, the definition of  $\hat{c}$  implies the following:

$$\begin{aligned} c_{t'}|_{\mathcal{S}(X(t))} &= \hat{c}|_{\mathcal{S}(X(t'))}|_{\mathcal{S}(X(t))} = \hat{c}|_{\mathcal{S}(X(t))} = c_t, \\ R_{t'} \cap X(t)^2 &= R(\hat{G}[V(t')] \circ \hat{c}) \cap X(t')^2 \cap X(t)^2 = R(\hat{G}[V(t)] \circ \hat{c}) \cap X(t)^2 = R_t, \\ R_{t'}[v] \cap X(t) &= R(\hat{G}[V(t)] \circ \hat{c})[v] \cap X(t) \neq \emptyset. \end{aligned}$$

Thus,  $(c_{t'}, R_{t'})$  is considered in the minimum of the right-hand side of:

$$W[t, c_t, R_t] \geq \min_{\substack{c_{t'} \in \mathcal{C}(\hat{G}[X[t']]) \\ R_{t'} \subseteq X_{t'}^2}} \text{valid for } c_t, R_t} W[t', c_{t'}, R_{t'}] \quad (2)$$

Now, consider an arbitrary pair  $(c_{t'}, R_{t'})$  such that  $c_{t'}|_{\mathcal{S}(X(t))} = c_t$ ,  $R_{t'} \cap X(t)^2 = R_t$ , and  $R_{t'}[v] \cap X(t) \neq \emptyset$ . Let  $\hat{c} \in \mathcal{C}(\hat{G}[V(t')])$  be a configuration for which the minimum weight is attained in Definition 1 for  $W[t', c_{t'}, R_{t'}]$ . Since  $V(t') = V(t)$ , we have  $\hat{c} \in \mathcal{C}(\hat{G}[V(t)])$ . It then follows:

$$\begin{aligned} c_t &= c_{t'}|_{\mathcal{S}(X(t))} = \hat{c}|_{\mathcal{S}(X(t'))}|_{\mathcal{S}(X(t))} = \hat{c}|_{\mathcal{S}(X(t))}, \\ R_t &= R_{t'} \cap X(t)^2 = R(\hat{G}[V(t')] \circ \hat{c}) \cap X(t')^2 \cap X(t)^2 = R(\hat{G}[V(t)] \circ \hat{c}) \cap X(t)^2, \\ R_{t'}[v] \cap X(t) &= R(\hat{G}[V(t)] \circ \hat{c})[v] \cap X(t) \neq \emptyset. \end{aligned}$$

Hence,  $\hat{c}$  satisfies all three conditions in Definition 1 for  $W[t, c_t, R_t]$ , and is considered in the minimization, resulting in:

$$W[t, c_t, R_t] \leq w(\hat{c}) = W[t', c_{t'}, R_{t'}] \quad (3)$$

Since this holds for all valid  $(c_{t'}, R_{t'})$ , we obtain:

$$W[t, c_t, R_t] \leq \min_{\substack{c_{t'} \in \mathcal{C}(\hat{G}[X[t']]) \\ R_{t'} \subseteq X_{t'}^2}} \text{valid for } c_t, R_t} W[t', c_{t'}, R_{t'}] \quad (4)$$

Combining inequalities (2) and (4), we conclude that the entry  $W[t, c_t, R_t]$  in the dynamic programming table is defined as stated in the lemma.  $\square$

**Join node.** Finally, we consider the case where  $t$  is a join node, with children  $t_1$  and  $t_2$ . Intuitively, to compute the value of  $W[t, c_t, R_t]$ , we consider pairs of entries from  $t_1$  and  $t_2$  whose configuration on switches incident to  $X(t)$  matches  $c_t$ . This allows for their respective configuration graphs of  $\hat{G}[V(t_1)]$  and  $\hat{G}[V(t_2)]$  to be merged. In the process, it can occur that vertices in  $X(t)$  that were not connected before, may be reachable now.

Hence, we restrict ourselves to pairs of entries where the resulting reachability relation on vertices of  $X(t)$  is equal to  $R_t$ . Taking the minimum of the added values for each pair, and subtracting the weight of the common switches (which are exactly the switches incident to  $X(t)$ ) yields the value of  $W[t, c_t, R_t]$ .

**Lemma 19.** *If  $t$  is a join node with children  $t_1$  and  $t_2$ , then for every configuration  $c_t \in \mathcal{C}(\widehat{G}[X(t)])$  of switches incident to  $X(t)$  and equivalence relation  $R_t \subseteq X(t)^2$ , the following recursive definition of  $W[t, c_t, R_t]$  satisfies the invariant in Definition 1:*

$$W[t, c_t, R_t] = \min_{\substack{R_1, R_2 \subseteq X(t)^2 \\ \text{s.t. } (R_1 \cup R_2)^+ = R_t}} \{W[t_1, c_t, R_1] + W[t_2, c_t, R_2]\} - w(c_t)$$

*Proof.* Let  $\hat{c}$  be a configuration of  $\widehat{G}[V(t)]$  that attains the minimum in the definition of  $W[t, c_t, R_t]$  as specified in Definition 1. Then  $c_1 := \hat{c}|_{\mathcal{S}(V(t_1))}$  and  $c_2 := \hat{c}|_{\mathcal{S}(V(t_2))}$  are configurations of  $\widehat{G}[V(t_1)]$  and  $\widehat{G}[V(t_2)]$ , respectively. For  $c_t$  and  $R_1 := R(\widehat{G}[V(t_1)] \circ c_1) \cap X(t)^2$ , configuration  $c_1$  clearly fulfills conditions (1) and (2) of Definition 1 for  $W[t_1, c_t, R_1]$ . Condition (3) is also fulfilled since for every  $u \in V(t_1)$ , there exists a path to a vertex of  $X(t)$  without passing through any vertices of  $V(t_2) \setminus X(t)$  in  $\widehat{G}[V(t)] \circ c$ , hence the same path exists in  $\widehat{G}[V(t_1)] \circ c_1$ . Therefore,  $c_1$  is a configuration considered in Definition 1 for  $W[t_1, c_t, R_1]$ . Analogously, for  $c_t$  and  $R_2 := R(\widehat{G}[V(t_2)] \circ c_2) \cap X(t)^2$ , configuration  $c_2$  is considered in Definition 1 for  $W[t_2, c_t, R_2]$ . In combination with the fact that  $V(t_1) \cap V(t_2) = X(t)$ , this implies that:

$$\begin{aligned} W[t_1, c_t, R_1] + W[t_2, c_t, R_2] &\leq w(c_1) + w(c_2) \\ &= w(\hat{c}|_{\mathcal{S}(V(t_1))}) + w(\hat{c}|_{\mathcal{S}(V(t_2))}) \\ &= w(\hat{c}) + w(\hat{c}|_{\mathcal{S}(X(t))}) \\ &= W[t, c_t, R_t] + w(c_t) \end{aligned} \tag{1}$$

Employing Lemma 14, condition (2) of  $\hat{c}$  in Definition 1 for  $W[t, c_t, R_t]$  proves that:

$$\begin{aligned} (R_1 \cup R_2)^+ &= ((R(\widehat{G}[V(t_1)] \circ c_1) \cap X(t)^2) \cup (R(\widehat{G}[V(t_2)] \circ c_2) \cap X(t)^2))^+ \\ &= ((R(\widehat{G}[V(t_1)] \circ c_1) \cup R(\widehat{G}[V(t_2)] \circ c_2)) \cap X(t)^2)^+ \\ &= (R(\widehat{G}[V(t_1)] \circ c_1) \cup R(\widehat{G}[V(t_2)] \circ c_2))^+ \cap X(t)^2 \\ &= R(\widehat{G}[V(t)] \circ \hat{c}) \cap X(t)^2 = R_t. \end{aligned}$$

Thus,  $R_1$  together with  $R_2$  are considered in the recursive definition of the lemma. In combination with Eq. (1), we then have that:

$$W[t, c_t, R_t] \geq \min_{\substack{R_1, R_2 \subseteq X(t)^2 \\ (R_1 \cup R_2)^+ = R_t}} \{W[t_1, c_t, R_1] + W[t_2, c_t, R_2] - w(c_t)\} \tag{2}$$

On the other hand, consider arbitrary equivalence relations  $R_1, R_2 \subseteq X(t)^2$  where  $(R_1 \cup R_2)^+ = R_t$ . Let  $\hat{c}_1$  and  $\hat{c}_2$  be configurations of  $\widehat{G}[V(t_1)]$  and  $\widehat{G}[V(t_2)]$  for which

the minimum weights are attained in  $W[t_1, c_t, R_1]$  and  $W[t_2, c_t, R_2]$ , respectively. We construct a configuration  $c$  of  $\widehat{G}[V(t)]$  such that for each switch  $S \in \mathcal{S}(V(t))$ :

$$c(S) := \begin{cases} c_t(S) & \text{if } S \in \mathcal{S}(X(t)) \\ \hat{c}_1(S) & \text{if } S \in \mathcal{S}(V(t_1)) \setminus \mathcal{S}(X(t)) \\ \hat{c}_2(S) & \text{if } S \in \mathcal{S}(V(t_2)) \setminus \mathcal{S}(X(t)) \end{cases}$$

Since  $\hat{c}_1$  and  $\hat{c}_2$  fulfill the conditions of Definition 1 for  $W[t_1, c_t, R_1]$  and  $W[t_2, c_t, R_2]$ , respectively, configuration  $c$  satisfies conditions (1) and (3) in Definition 1 for  $W[t, c_t, R_t]$  as well. By utilizing Lemma 14, we can also confirm that condition (2) is fulfilled:

$$\begin{aligned} R_t &= (R_1 \cup R_2)^+ = ((R(\widehat{G}[V(t_1)] \circ \hat{c}_1) \cap X(t)^2) \cup (R(\widehat{G}[V(t_2)] \circ \hat{c}_2) \cap X(t)^2))^+ \\ &= ((R(\widehat{G}[V(t_1)] \circ \hat{c}_1) \cup R(\widehat{G}[V(t_2)] \circ \hat{c}_2)) \cap X(t)^2)^+ \\ &= (R(\widehat{G}[V(t_1)] \circ \hat{c}_1) \cup R(\widehat{G}[V(t_2)] \circ \hat{c}_2))^+ \cap X(t)^2 \\ &= R(\widehat{G}[V(t)] \circ c) \cap X(t)^2 \end{aligned}$$

Hence,  $c$  is one of the configurations considered in Definition 1 for  $W[t, c_t, R_t]$ . In combination with the fact that  $V(t_1) \cap V(t_2) = X(t)$ , this implies that:

$$\begin{aligned} W[t, c_t, R_t] &\leq w(c) \\ &= w(c|_{\mathcal{S}(V(t_1))}) + w(c|_{\mathcal{S}(V(t_2))}) - w(c|_{\mathcal{S}(X(t))}) \\ &= w(\hat{c}_1) + w(\hat{c}_2) - w(c_t) \\ &= W[t_1, c_t, R_1] + W[t_2, c_t, R_2] - w(c_t) \end{aligned}$$

As this holds for all valid  $R_1, R_2$ , we obtain:

$$W[t, c_t, R_t] \leq \min_{\substack{R_1, R_2 \subseteq X(t)^2 \\ (R_1 \cup R_2)^+ = R_t}} \{W[t_1, c_t, R_1] + W[t_2, c_t, R_2] - w(c_t)\} \quad (3)$$

Combining (2) and (3), we conclude that the entry  $W[t, c_t, R_t]$  in the dynamic programming table is defined as stated in the lemma.  $\square$

We implement these recursive definitions in Algorithm 9.1, a dynamic programming algorithm that computes the minimum weight over all configurations  $\hat{c} \in \mathcal{C}(\widehat{G})$  such that  $\widehat{G} \circ \hat{c}$  is globally connected. The entries of the dynamic programming table  $W$  are computed in a bottom-up fashion by traversing  $T$  in post-order. For each non-leaf node  $t$ , the values  $W[t, \cdot, \cdot]$  are calculated gradually by iterating over the values of its child nodes.

The correctness of the algorithm follows directly from Lemmas 15 to 19. For the running time analysis, consider an arbitrary node  $t \in T$  of the tree decomposition.

The computational bottleneck lies in the nested loops of the main routine, which enumerate all possible configurations of switches incident to  $X(t)$  and all possible equivalence relations on  $X(t)$ . Both quantities grow exponentially with the relevant parameters,

---

**Algorithm 9.1:** Dynamic programming algorithm computing the minimum weight over all connected configurations  $\hat{c} \in \mathcal{C}(\hat{G})$

---

**Input** : A switch graph  $\hat{G}$  restricted to connected switches, a weight function  $w: \mathcal{S}(\hat{G}) \times \mathbb{N}_{>0} \rightarrow \mathbb{R}$ , and a nice tree decomposition  $\mathcal{T} = (T, X)$  of  $G(\hat{G})$

**Output**: Minimum weight over all connected configurations  $\hat{c} \in \mathcal{C}(\hat{G})$

```

1  $W[t, c_t, R_t] \leftarrow \infty$  for each  $t \in V(T)$ ,  $c_t \in \mathcal{C}(\hat{G}[X_t])$ , and equivalence relation
    $R_t \subseteq X(t)^2$ 
2 for  $t \in V(T)$  in post-order do
3   if  $t$  is a leaf node then
4      $W[t, \emptyset, \emptyset] \leftarrow 0$ 
5   else if  $t$  is an introduce node with child  $t'$  with  $X(t) = X(t') \cup \{v\}$  then
6     for configuration  $c_v$  of switches  $\mathcal{S}(v) \setminus \mathcal{S}(X(t))$  do
7        $w_v \leftarrow w(c_v)$ 
8       for configuration  $c_{t'} \in \mathcal{C}(\hat{G}[X(t')])$  do
9         for equivalence relation  $R_{t'} \subseteq X(t')^2$  do
10            $c_t \leftarrow c_{t'} \cup c_v$ 
11            $R_t \leftarrow (R_{t'} \cup R(\hat{G}[X(t)] \circ c_t))^+$ 
12            $W[t, c_t, R_t] \leftarrow \min\{W[t, c_t, R_t], W[t, c_{t'}, R_{t'}] + w_v\}$ 
13   else if  $t$  is a forget node with child  $t'$  with  $X(t) \cup \{v\} = X(t')$  then
14     for configuration  $c_{t'} \in \mathcal{C}(\hat{G}[X(t')])$  do
15       for equivalence relation  $R_{t'} \subseteq X_{t'}^2$  where  $R_{t'}[v] \cap X(t) \neq \emptyset$  do
16          $c_t \leftarrow c_{t'}|_{\mathcal{S}(X(t))}$ 
17          $R_t \leftarrow R_{t'} \cap X(t)^2$ 
18          $W[t, c_t, R_t] \leftarrow \min\{W[t, c_t, R_t], W[t', c_{t'}, R_{t'}]\}$ 
19   else if  $t$  is a join node with children  $t_1$  and  $t_2$  then
20     for configuration  $c_t \in \mathcal{C}(\hat{G}[X(t)])$  do
21        $w_t \leftarrow w(c_t)$ 
22       for equivalence relations  $R_1, R_2 \subseteq X(t)^2$  do
23          $R_t \leftarrow (R_1 \cup R_2)^+$ 
24          $W[t, c_t, R_t] \leftarrow \min\{W[t, c_t, R_t], W[t_1, c_t, R_{t_1}] + W[t_2, c_t, R_{t_2}] - w_t\}$ 
25  $r' \leftarrow$  child of  $r(T)$  with  $X(r') = \{v\}$ 
26  $w_{r'} \leftarrow \infty$ 
27 for configuration  $c_{r'} \in \mathcal{C}(\hat{G}[X(r')])$  do
28    $w_{r'} \leftarrow \min\{w_{r'}, W[r', c_{r'}, \{(v, v)\}]\}$ 
29 return  $w_{r'}$ 

```

---

namely the bag size, the switch degree of each vertex, and the number of switch positions per switch. All other operations within these loops can be performed in time that depends only on these parameters.

As a result, the entire algorithm runs in FPT time when parameterized by the treewidth of  $G(\widehat{G})$ , the maximum switch degree over all vertices, and the maximum number of switch positions over all switches. The following lemma presents this result in detail:

**Lemma 20.** *Given a switch graph  $\widehat{G}$  restricted to connected switches, a weight function  $w: \mathcal{S}(\widehat{G}) \times \mathbb{N}_{>0} \rightarrow \mathbb{R}$ , and a nice tree decomposition  $\mathcal{T} = (T, X)$ , we can compute the minimum weight over all connected configurations of  $\widehat{G}$  in FPT-time parameterized by  $k$ ,  $\Delta$ , and  $s$ , or more specifically in  $\mathcal{O}(s^{(k+1)\cdot\Delta} \cdot k^{k+2} \cdot (\Delta + k) \cdot n)$  time, where*

- $n$  is the number of vertices in the in  $T$ , that is  $m := |V(T)|$ ,
- $k$  is the width of the tree decomposition  $\mathcal{T}$ ,
- $\Delta$  is the maximum switch degree, that is  $\Delta := \max_{v \in V(\widehat{G})} \deg_{\widehat{G}}(v)$ ,
- $s$  is the maximum number of switch positions, that is  $s := \max_{S \in \mathcal{S}(\widehat{G})} |S|$ .

*Proof.* The correctness of Algorithm 9.1 follows from Lemmas 15 to 19. We proceed by analyzing the running time.

For each node  $t \in V(T)$ , we know that the size of bag  $X(t)$  is at most  $k + 1$ . Hence, the number of configurations of switches incident to  $X(t)$  can be upper bounded by  $s^{(k+1)\cdot\Delta}$  while the the number of equivalence relations on  $X(t)$  can be upper bounded by  $(k + 1)^{k+1}$ .

Each equivalence relation is represented by assigning to every element a reference to a representative of its equivalence class. By imposing a total order on the vertices of  $\widehat{G}$ , we ensure that the representative is always chosen as the smallest element in its equivalence class.

Given two equivalence relations  $R_1$  and  $R_2$  over a totally ordered set  $X$ , the transitive closure  $(R_1 \cup R_2)^+$  can be computed in  $\mathcal{O}(|X|^2)$  time. This is accomplished by iterating over all elements of  $X$  and comparing their representative elements. If two elements have different representatives, we select the smaller representative and update all elements previously associated with the larger representative to point to the smaller one.

The main loop executes a total of  $m$  iterations, with each iteration corresponding to one of the conditional branches. We now analyze the running time associated with each branch:

- If  $t$  is a leaf node, then the entry  $W[t, \emptyset, \emptyset]$  is set to zero in constant time.

- If  $t$  is an introduce node, then the algorithm essentially iterates over all possible configurations of switches incident to  $X(t)$  and all possible equivalence relations on bag  $X(t')$  of child node  $t'$ . Since the introduced vertex  $v$  has at most  $\Delta$  incident switches, computing the weight in Line 7 can be done in  $\mathcal{O}(\Delta)$  time. Combining the configurations in Line 10 is done implicitly when iterating over the configurations of switches incident to  $X(t)$ .

In Line 7, rather than computing  $R(\widehat{G}[X(t)] \circ c_t)$  in its entirety, we can improve efficiency by determining only the vertices in  $X(t)$  that are adjacent to  $v$  under the configuration  $c_t$ . This suffices, as all other edges are already accounted for in  $R_{t'}$ , resulting in a reachability relation focused on  $v$ . Since there are at most  $\Delta$  switches incident to  $v$  and at most  $k$  edges between  $v$  and vertices in  $X(t)$ , this computation requires  $\mathcal{O}(\Delta \cdot k)$  time. As previously noted, the transitive closure of the combined reachability relation and  $R_{t'}$  can be computed in  $\mathcal{O}(k^2)$  time. Therefore, Line 7 can be executed in  $\mathcal{O}(\Delta \cdot k + k^2)$  time.

Finally, in Line 17, two entries of the dynamic programming table must be accessed. Since each configuration of switches incident to  $X(t)$  can be encoded in  $\Delta \cdot (k + 1)$  space and each equivalence relation on  $X(t)$  in  $k + 1$  space, such access can be performed in  $\mathcal{O}(\Delta \cdot k + k)$  time. Therefore, this line can be executed in  $\mathcal{O}(\Delta \cdot k + k)$  time.

Overall, the total running time of this branch is  $\mathcal{O}(s^{(k+1) \cdot \Delta} \cdot k^{k+2} \cdot (\Delta + k))$ .

- If  $t$  is a forget node with child node  $t'$ , then the algorithm again iterates over all possible configurations of switches incident to  $X(t')$  and all possible equivalence relations on  $X(t')$ . Removing the forgotten vertex from the configuration and equivalence relation in Lines 16 and 17 is handled implicitly. In Line 18, accessing two entries of the dynamic programming table can be performed in  $\mathcal{O}(\Delta \cdot k + k)$  time. Consequently, the total running time for this branch is  $\mathcal{O}(s^{(k+1) \cdot \Delta} \cdot k^{k+2} \cdot \Delta)$ .
- If  $t$  is a join node, then the algorithm iterates over all possible configurations of switches incident to  $X(t)$  and all possible pairs of equivalence relations on  $X(t)$ . As we have at most  $\delta \cdot (k + 1)$  switches, computing the weight of the configuration in Line 21 can be achieved in  $\mathcal{O}(\delta \cdot k)$  time.

In Line 23, computing the transitive closure of the two combined equivalence relations on  $X(t)$  is possible in time  $\mathcal{O}(k^2)$ . Finally, in Line 24, accessing three entries of the dynamic programming table can be performed in  $\mathcal{O}(\Delta \cdot k + k)$  time.

All in all, the total running time of this branch amounts to  $\mathcal{O}(s^{(k+1) \cdot \Delta} \cdot k^{k+2} \cdot (\Delta + k))$  time.

Hence, each iteration of the main loop takes  $\mathcal{O}(s^{(k+1) \cdot \Delta} \cdot k^{k+2} \cdot (\Delta + k))$  time. Since the running time of Line 1 and Lines 25 to 29 is negligible in contrast, the running time of the entire algorithm is given by the main loop. As we have  $n$  iterations of the main loop, we obtain the running time as stated in the lemma.  $\square$

Note that the algorithm can be slightly modified to also produce a witnessing connected configuration of minimum weight by employing backtracking.

Based on the preliminaries from Section 9.1, a nice tree decomposition of the union graph  $G(\widehat{G})$ , with width  $2 \cdot \text{tw}(G(\widehat{G})) + 1$  and at most  $\mathcal{O}(\text{tw}(G(\widehat{G})) \cdot |V(\widehat{G})|)$  nodes, can be computed in FPT time, parameterized by the treewidth of  $G(\widehat{G})$ . Thus, WEIGHTED-GCON-SWITCHING restricted to connected switches can be decided in FPT time when parameterized by the treewidth of  $G(\widehat{G})$ , the maximum switch degree over all vertices, and the maximum number of switch positions over all switches. This results in the following theorem:

**Theorem 30.** WEIGHTED-GCON-SWITCHING restricted to connected switches can be solved in FPT time parameterized by  $k$ ,  $\Delta$ , and  $s$ , or more specifically in  $2^{\mathcal{O}(k)} \cdot s^{\mathcal{O}(k \cdot \Delta)} \cdot k^{\mathcal{O}(k)} \cdot n$  time, where

- $n$  is the number of vertices in  $\widehat{G}$ , that is  $n := |V(\widehat{G})|$
- $k$  is the treewidth of the union graph  $G(\widehat{G})$ , that is  $k := \text{tw}(G(\widehat{G}))$ ,
- $\Delta$  is the maximum switch degree, that is  $\Delta := \max_{v \in V(\widehat{G})} \deg_{\widehat{G}}(v)$ ,
- $s$  is the maximum number of switch positions, that is  $s := \max_{S \in \mathcal{S}(\widehat{G})} |S|$ .

### 9.3 Adaption for $s$ - $t$ -(Dis)Connectivity

In this section, we adapt the dynamic programming algorithm from Algorithm 9.1 to solve WEIGHTED-STCON-SWITCHING and WEIGHTED-STDISCON-SWITCHING as well. Let  $\widehat{G}$  be a switch graph restricted to connected switches,  $w: \mathcal{S}(\widehat{G}) \times \mathbb{N}_{>0} \rightarrow \mathbb{R}$  a weight function on the switch positions, and  $\mathcal{T} = (T, X)$  a nice tree decomposition of the union graph  $G(\widehat{G})$ .

First, we modify  $\mathcal{T}$  so that the source vertex  $s(\widehat{G})$  and the target vertex  $t(\widehat{G})$  are never forgotten. More precisely, for each forget node of  $s(\widehat{G})$  or  $t(\widehat{G})$ , we remove the node, add an edge between its parent and child, and include the corresponding terminal in the bags of all its ancestors. The modified tree decomposition is still nice, except that the root node now contains both terminals,  $s(\widehat{G})$  and  $t(\widehat{G})$ . Moreover, the width of the tree decomposition is increased by at most two.

Next, we adjust the dynamic programming table  $W$  by removing condition (3) from Definition 1, since it is no longer necessary for all vertices to be reachable from one another. This change does not affect the (recursive) definitions for leaf, introduce, or join nodes, but the definition for forget nodes must be updated. In particular, we drop condition (3) of Lemma 18 which required that the forgotten vertex must be reachable from the bag.

At the root node  $r(T)$ , the bag contains exactly the terminals, that is  $X(r(T)) = \{s(\widehat{G}), t(\widehat{G})\}$ . To compute the minimum weight of a configuration admitting an  $s$ - $t$  path,

we consider entries  $W[r(T), c_r, R_r]$  where  $R_r$  is an equivalence relation on  $\{s(\widehat{G}), t(\widehat{G})\}$  such that  $s(\widehat{G})$  and  $t(\widehat{G})$  are in the same equivalence class. These entries correspond to the weights of configurations of  $\widehat{G}$  that admit an  $s$ - $t$ -path. To determine the minimum weight of such configurations, we need to evaluate the following expression:

$$\min_{c_r \in \mathcal{C}(\widehat{G}[\{s(\widehat{G}), t(\widehat{G})\}])} W[r(T), c_r, \{s(\widehat{G}), t(\widehat{G})\}^2].$$

For  $s$ - $t$ -disconnectivity, we instead consider entries where  $s(\widehat{G})$  and  $t(\widehat{G})$  are in separate equivalence classes. To determine the minimum weight of configurations which do not admit an  $s$ - $t$ -path, we thus have to evaluate the following expression:

$$\min_{c_r \in \mathcal{C}(\widehat{G}[\{s(\widehat{G}), t(\widehat{G})\}])} W[r(T), c_r, \{(s(\widehat{G}), s(\widehat{G})), (t(\widehat{G}), t(\widehat{G}))\}].$$

These modifications yield the following results:

**Theorem 31.** WEIGHTED-STCON-SWITCHING restricted to connected switches can be solved in FPT time parameterized by  $k$ ,  $\Delta$ , and  $s$ , where

- $k$  is the treewidth of the union graph  $G(\widehat{G})$ , that is  $k := \text{tw}(G(\widehat{G}))$ ,
- $\Delta$  is the maximum switch degree, that is  $\Delta := \max_{v \in V(\widehat{G})} \deg_{\widehat{G}}(v)$ ,
- $s$  is the maximum number of switch positions, that is  $s := \max_{S \in \mathcal{S}(\widehat{G})} |S|$ .

**Theorem 32.** WEIGHTED-STDISCON-SWITCHING restricted to connected switches can be solved in FPT time parameterized by  $k$ ,  $\Delta$ , and  $s$ , where

- $k$  is the treewidth of the union graph  $G(\widehat{G})$ , that is  $k := \text{tw}(G(\widehat{G}))$ ,
- $\Delta$  is the maximum switch degree, that is  $\Delta := \max_{v \in V(\widehat{G})} \deg_{\widehat{G}}(v)$ ,
- $s$  is the maximum number of switch positions, that is  $s := \max_{S \in \mathcal{S}(\widehat{G})} |S|$ .

## 9.4 Global Connectivity with Bounded Radius

Returning to the result in Theorem 30 for WEIGHTED-GCON-SWITCHING, if treewidth is not included as a parameter, the problem becomes NP-hard even when the union graph is planar, since the unweighted version is already NP-hard for a constant switch degree, a constant switch size, connected switches, and a planar union graph (Theorem 24).

In an attempt to overcome this limitation, we consider an adapted version of WEIGHTED-GCON-SWITCHING with an additional restriction on the *radius* of the resulting configuration graph. In general, the radius of a graph  $G$ , denoted  $\text{rad}(G)$ , is defined as

$$\text{rad}(G) := \min_{v \in V(G)} \max_{u \in V(G)} \text{dist}_G(v, u),$$

where  $\text{dist}_G(v, u)$  is the length of the shortest path between vertices  $v$  and  $u$  in  $G$ .

To formally express this problem in our switching model, we define a family of graphs  $r$ -GCON containing all connected graphs with a radius of at most  $r$ . Thus, this adapted version of WEIGHTED-GCON-SWITCHING is simply WEIGHTED- $r$ -GCON-SWITCHING.

By restricting the union graph to be planar, we conjecture that WEIGHTED- $r$ -GCON-SWITCHING can be solved in FPT time when parameterized by  $\Delta$ ,  $s$ , and  $r$ , where  $\Delta$  denotes the maximum switch degree and  $s$  the maximum number of switch positions.

The approach follows a win/win strategy: either the union graph admits a tree decomposition of sufficiently small width, enabling the use of a dynamic programming algorithm similar to Algorithm 9.1 to solve the problem in FPT time, or the union graph contains a *grid minor* of such large size that no connected configuration of radius  $r$  can exist.

A graph  $H$  is a *minor* of a graph  $G$  if  $H$  can be formed from  $G$  by deleting edges, vertices, and by contracting edges. Graph  $H$  is a  $t \times t$  *grid* if there exists a bijection  $f : V(H) \rightarrow [t]^2$  from the vertices of  $H$  to two-dimensional integer coordinates such that two vertices  $u, v \in V(H)$  with coordinates  $f(u) = (x_u, y_u)$  and  $f(v) = (x_v, y_v)$  are adjacent if and only if  $|x_u - x_v| + |y_u - y_v| = 1$ .

When given a planar graph  $G$  and an integer  $t$ , there exists an  $\mathcal{O}(|V(G)|^2)$  algorithm that either outputs a tree decomposition of  $G$  of width at most  $5t$ , or provides a minor  $H$  of  $G$  that is a  $t \times t$  grid [4, Theorem 7.23].

Now, let  $\widehat{G}$  be a switch graph with connected switches and a planar union graph  $G(\widehat{G})$ , and  $w : \mathcal{S}(\widehat{G}) \times \mathbb{N}_{>0} \rightarrow \mathbb{R}$  a weight function. If  $G(\widehat{G})$  contains a  $5r \times 5r$  grid minor  $H$ , then for each vertex  $v \in V(H)$  with coordinates  $f(v) = (x_v, y_v)$ , there exists a vertex  $u \in V(H)$  with coordinates  $f(u) = (x_u, y_u)$  such that  $|x_u - x_v| > r$ ,  $|y_u - y_v| > r$ , and  $r < x_u, y_u < 4r$ , implying that  $v$  lies in the interior of the grid. It follows that the radius of  $H$  exceeds  $r$ . Moreover, we conjecture that reversing contractions and deletions in  $H$  preserves this property, as edges between interior vertices cannot span grid cells in a planar graph. Consequently,  $G(\widehat{G})$  also has radius greater than  $r$ , and therefore, every connected configuration of  $\widehat{G}$  must have radius larger than  $r$  as well. In this case, we can conclude that  $(\widehat{G}, w)$  is a no-instance of WEIGHTED- $r$ -GCON-SWITCHING.

Otherwise,  $G(\widehat{G})$  admits a tree decomposition of width at most  $25r$ . In this case, we can apply a dynamic programming algorithm similar to Algorithm 9.1. To ensure that we compute the minimum weight over all connected configurations with radius at most  $r$ , the algorithm must be modified accordingly. Specifically, we propose running the algorithm  $|V(G)|$  times, once for each vertex  $v \in V(G)$ , treating  $v$  as the center of the configuration graph. We conjecture that the dynamic programming routine can be adapted to enforce the radius constraint by maintaining distance information between vertices within the dynamic programming table. Specifically, for each node in the tree decomposition, it seems fruitful to extend the table beyond entries for equivalence relations on the corresponding bag, which capture the reachability between vertices in the bag

within the subgraph induced by the subtree rooted at that node, to also include distances between vertices in the bag. For each pair, one value records the current distance within the subgraph induced by the subtree rooted at the node, while another value tracks the additional distance required to connect these vertices using the remainder of the graph outside the bag and its subtree. This clearly increases the size of the dynamic programming table, but since the radius is bounded by  $r$ , we only need to consider distances up to  $r$ , ensuring that the number of entries for each bag depends only on  $r$ ,  $\Delta$ , and  $s$ . This approach should enable us to decide **WEIGHTED- $r$ -GCON-SWITCHING** in FPT time, parameterized by  $\Delta$ ,  $s$ , and  $r$ .

In summary, this leads to the following conjecture:

**Conjecture 1.** **WEIGHTED- $r$ -GCON-SWITCHING**, when restricted to connected switches and a planar union graph, can be solved in FPT time parameterized by  $\Delta$ ,  $s$ , and  $r$ , where

- $\Delta$  is the maximum switch degree, i.e.,  $\Delta := \max_{v \in V(\widehat{G})} \deg_{\widehat{G}}(v)$ ,
- $s$  is the maximum number of switch positions, i.e.,  $s := \max_{S \in \mathcal{S}(\widehat{G})} |S|$ ,

**Discussion.** In this chapter, we have shown that **WEIGHTED-II-SWITCHING** can be solved in FPT time for the three graph families studied in this thesis, when parameterized by the treewidth of the union graph, the maximum switch degree, and the maximum number of switch positions. To move beyond treewidth and instead restrict the union graph to be planar, we explored a win-win approach for a variant of global connectivity switching with a bounded radius on the configuration graph. Although a formal proof of Conjecture 1 is still pending, we believe this approach is promising and could lead to an FPT algorithm for **WEIGHTED- $r$ -GCON-SWITCHING**.

More broadly, several open questions remain regarding the parameterized complexity of **WEIGHTED-SWITCHING**. For example, is it possible to design a parameterized algorithm when switches are not necessarily connected? This appears unlikely with the current set of parameters. For unconnected switches, even when each switch has constant size, the problem is intractable for all considered  $\Pi$  (Theorems 15, 20 and 25), despite the union graph having treewidth of at most two. Although the switch degree is often high in these hardness results, it is easily possible to reduce instances to bounded switch degree by introducing static edges and possibly new vertices. Exploring alternative parameters not considered here may also be a fruitful direction for future research.



## Conclusion

In this thesis, we took the first steps toward formulating a unified switching framework capable of encompassing various variants of the switching problem studied in the literature, called  $\Pi$ -SWITCHING. In our analysis, we focused on three specific problem variants, all with the objective to find a configuration of switches such that the resulting graph admits a certain connectivity property. The analysis revolved around identifying restrictions on the instances, especially on the structure of the switches, that either allow for efficient algorithms or lead to computational hardness results. Additionally, we introduced two weighted variants of the switching problem, WEIGHTED- $\Pi$ -SWITCHING and ONOFF- $\Pi$ -SWITCHING, where we also analyzed the parameterized complexity. Fig. 10.1 gives an overview of the tractability results obtained in this thesis.

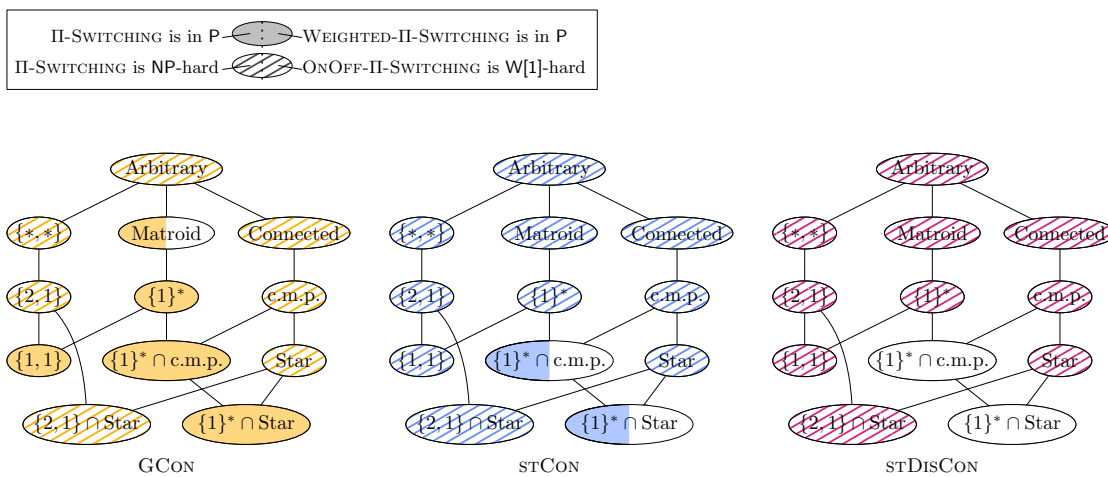


Figure 10.1: Overview of tractability for unweighted and weighted variants of the switching problem across graph families, depending on the allowed switch structure.

For the global connectivity variant, we extended the polynomial-time result of Katz et al. [11] to the weighted setting and showed that tractability depends less on the graph structure of the switches and more on whether their positions form a matroid-like system.

By contrast, for the  $s$ - $t$  connectivity and disconnection variants, the graph structure of switches becomes decisive: both remain NP-hard (and W[1]-hard in the weighted case) even for very simple  $\{1, 1\}$ -switches, which always satisfy the matroid property. However, for STCON-SWITCHING, restricting to  $\{1\}^*$ -switches where the switch-induced graphs are complete multipartite (including stars) makes the problem polynomial-time solvable. In fact, this yields a strict dichotomy: as soon as the set of admissible switch-induced graphs contains even a single graph that is not complete multipartite, the problem becomes NP-hard.

When moving beyond switches with at most one edge per position, the studied switching problems quickly become intractable. Already with simple  $\{2, 1\}$ -switches, all three variants are NP-hard (and W[1]-hard in the weighted case). This hardness persists even under strong additional restrictions, such as requiring all switch induced graphs to be stars; and for GCON-SWITCHING, it holds even when the union graph is planar with maximum switch degree three.

To tackle this intractability, we investigated parameterized complexity for WEIGHTED-II-SWITCHING to identify fixed-parameter tractable cases. We developed a dynamic programming algorithm that only requires switches to be connected and achieves an FPT running time when parameterized by the maximum switch degree, the maximum number of switch positions, and, most importantly, the treewidth of the union graph.

Looking ahead, there remain numerous directions for future research in the study of switching problems. Many open questions arise naturally from the overview in Fig. 10.1, where several combinations of graph families and switch restrictions are yet to be resolved for both II-SWITCHING and WEIGHTED-II-SWITCHING. For instance:

- Is WEIGHTED-GCON-SWITCHING solvable in polynomial time for matroid-switches, as is the case for GCON-SWITCHING?
- Is WEIGHTED-STCON-SWITCHING solvable in polynomial time for complete multipartite  $\{1\}^*$ -switches, as is the case for STCON-SWITCHING?
- Are STDISCON-SWITCHING and WEIGHTED-STDISCON-SWITCHING polynomial-time solvable for complete multipartite  $\{1\}^*$ -switches, similar to STCON-SWITCHING?

For GCON-SWITCHING with matroid-switches, it is conceivable that a dichotomy result analogous to that for STCON-SWITCHING with  $\{1\}^*$ -switches could be established. Specifically, if the set of admissible switches includes even a single non-matroid switch, does the problem become NP-hard? Likewise, it would be interesting to investigate whether STDISCON-SWITCHING admits a similar dichotomy for switches allowing more than one edge per position. Notably, restricting switch-induced graphs to be complete

---

multipartite is not sufficient for tractability, as `STDISCON-SWITCHING` is already NP-hard for  $\{2, 1\}$ -star switches.

Furthermore, although we have established that `GCON-SWITCHING` is NP-hard for  $\{2, 1\}$ -star switches, even when the union graph is planar with a maximum switch degree of three, it remains an open question whether we can show that `ONOFF-GCON-SWITCHING` is  $W[1]$ -hard under similar constraints, as our current reduction does not guarantee a planar union graph. Similarly, it is also worth exploring whether `STCON-SWITCHING` and `STDISCON-SWITCHING` remain NP-hard under these same constraints.

With respect to the FPT results, we have already begun to explore a win-win approach that builds on our dynamic programming algorithm, solving `WEIGHTED-GCON-SWITCHING` with connected switches in FPT time when parameterized by the maximum switch degree and the maximum number of switch positions, while dropping the treewidth parameter at the cost of requiring the union graph to be planar. Future work could aim to validate and fully develop this approach, or show that such an approach is not feasible. However, if successful, it would be interesting to explore whether the approach can be generalized to `WEIGHTED-STCON-SWITCHING` and `WEIGHTED-STDISCON-SWITCHING`. It would also be valuable to investigate alternative parameterizations and restrictions to identify further FPT cases, for example, when restricting to star switches or when parameterizing by the vertex cover number of the union graph.

Beyond the restrictions studied in this thesis, there are many other natural ones to consider. For instance, one could consider switches whose induced graphs are paths, cycles, or trees, or impose new structural constraints on the union graph. Entirely different types of restrictions, not yet explored, may also prove fruitful.

Another promising direction is to study graph families beyond the three considered here. For example, one could seek configurations that yield planar graphs, Eulerian graphs, or disjoint unions of cliques. An adaptation to directed edges also appears natural, as directed versions have been explored in earlier work [19, 8, 9] and could have practical applications, for example, in modeling traffic networks.

Overall, this thesis has laid the groundwork for a unified framework for switching problems, with several open questions and directions remaining for future research. Its flexibility allows many problems to be modeled as switching instances subject to simple structural constraints, and several existing problems easily reduce to such restricted cases. Thus, we hope that this work provides not only a foundation for understanding switching problems in a unified way but also a practical tool for deriving complexity results for other graph-related problems.



# Overview of Generative AI Tools Used

In the course of writing this thesis, I have made use of the following two AI-tools: **ChatGPT** by OpenAI and **Github Copilot**.

Both tools were used throughout the thesis to refine sentence structure, improve clarity, and correct grammar or spelling. The prompts were adjusted to ensure that the adaptations were kept to a minimum and that the original meaning was preserved. None of the adapted passages were inserted verbatim without modification. All AI-suggested edits were reviewed and further modified as needed.

Another use case of AI tools was to generate summaries of certain sections of the thesis where such a summary was needed. However, these summaries were used merely as a starting point and were usually heavily modified to fit the context of the thesis.

In conclusion, no AI tools were used to generate original content, proofs, or research ideas. Their role was strictly supportive, with the main intent of ensuring higher clarity and readability of my own text.



# List of Figures

1.1	An overview of the complexity results for II-SWITCHING, WEIGHTED-II-SWITCHING, and ONOFF-II-SWITCHING for global connectivity, $s$ - $t$ -connectivity, and $s$ - $t$ -disconnectivity under various restrictions on switches (see Table 1.1 for the meaning of parameters). Edges between two switch restrictions represent containment, meaning the lower restriction is a special case of the upper one. Consequently, hardness results propagate upwards, while tractability results propagate downwards. . . . .	8
2.1	The left shows a switch graph $\widehat{G}$ with four switches (including one static edge): $S_1 := (\{v_2v_4, v_4v_5\}, \{v_1v_4\})$ , $S_2 := (\{v_2v_3\}, \{v_2v_6\})$ , $S_3 := (\{v_3v_6\}, \{v_4v_6\}, \{v_1v_2\})$ , $S_{v_5v_6} = (\{v_5v_6\})$ . Each color represents a different switch, while the different dash styles represent the different switch positions. The right shows $\widehat{G}$ configured by $c$ where $c(S_1) = c(S_2) = c(S_{v_5v_6}) = 1$ and $c(S_3) = 3$ . . . . .	12
4.1	Example of a graph $G$ with transition system $T$ (left) and the corresponding switch graph $\widehat{G}_T$ (right), constructed according to Construction 2. Each transition graph with its vertex and its corresponding switch are shown in the same color. Static edges in $\widehat{G}_T$ are shown in black. The yellow-highlighted edges indicate a possible $T$ -compatible $s$ - $t$ -path in $G$ and the corresponding $s$ - $t$ -path in $\widehat{G}_T$ under a valid configuration. . . . .	26
4.2	Example of Construction 3 applied to a $\{1\}$ -switch $S$ . The left side shows the original switch $S$ , along with a partition of $V(S)$ into two parts and an edge clique cover for each part. On the right, the corresponding two star switches centered at a fresh vertex $w$ are depicted. Small numbers indicate the switch position of each edge. . . . .	28
4.3	Example of a switch graph $\widehat{G}$ (left) and the corresponding graph $G$ with transition system $T$ (right), constructed according to Construction 4. Each switch and its corresponding vertex and transition graph are shown in the same color. The transition graphs of original vertices are shown in gray. The yellow-highlighted edges indicate a possible $s$ - $t$ -path in $\widehat{G}$ under a valid configuration and the corresponding $T$ -compatible $s$ - $t$ -path in $G$ . . . . .	30

6.1	Example switch graph $\widehat{G}_\varphi$ for a propositional formula $\varphi$ in negation normal form, constructed as described in Construction 9. Each color corresponds to a variable and its associated switch; solid edges represent edges of the on-position, while dashed edges represent edges of the off-position. . . . .	55
6.2	Example of the transformation from a $\{1, 1\}$ -switch to four $\{2, 1\}$ -star switches according to Construction 10. Each switch is represented by a different color, with solid edges indicating the on-position and dashed edges indicating the off-position. . . . .	59
7.1	Example switch graph $\widehat{G}_\varphi^*$ for a 2-normalized formula $\varphi$ , constructed as described in Construction 11. Each color corresponds to a variable and its associated switch; solid edges represent edges of the on-position, while dashed edges represent edges of the off-position. . . . .	67
7.2	Example switch graph $\widehat{G}_\varphi^\bullet$ for a propositional formula $\varphi \in \Delta_{2,3}^{\text{planar}}$ , constructed as described in Construction 12. Each color corresponds to a variable and its associated vertex and switch; solid edges represent edges of the on-position, while dashed edges represent edges of the off-position. The grey edges represent static edges that connect the variable vertices in a cycle. . . . .	70
8.1	Example switch graph $\widehat{G}_{\overline{\varphi}}$ of the negated formula $\overline{\varphi}$ in NNF, constructed as described in Construction 9, derived from a propositional formula $\varphi$ in NNF. Each color corresponds to a variable and its associated switch; solid edges represent edges of the on-position, while dashed edges represent edges of the off-position. . . . .	74
8.2	Example of the transformation from a $\{1, 1\}$ -switch to two $\{2, 1\}$ -star switches according to Construction 13. Each switch is represented by a different color, with solid edges indicating the on-position and dashed edges indicating the off-position. . . . .	76
10.1	Overview of tractability for unweighted and weighted variants of the switching problem across graph families, depending on the allowed switch structure. .	101

# List of Tables

1.1	Variables and parameters used in the complexity results . . . . .	5
1.2	Complexity results for GCON-SWITCHING, WEIGHTED-GCON-SWITCHING, and ONOFF-GCON-SWITCHING under various restrictions on switches and the union graph (see Table 1.1 for the meaning of variables and parameters).	6
1.3	Complexity results for STCON-SWITCHING, WEIGHTED-STCON-SWITCHING, and ONOFF-STCON-SWITCHING under various restrictions on switches and the union graph (see Table 1.1 for the meaning of variables and parameters).	7
1.4	Complexity results for STCON-SWITCHING, WEIGHTED-STCON-SWITCHING, and ONOFF-STCON-SWITCHING under various restrictions on switches and the union graph (see Table 1.1 for the meaning of parameters). . . . .	8



# List of Algorithms

9.1	Dynamic programming algorithm computing the minimum weight over all connected configurations $\hat{c} \in \mathcal{C}(\hat{G})$ . . . . .	93
-----	---	----



# Bibliography

- [1] Hugo A. Akitaya, Erik D. Demaine, Matias Korman, Irina Kostitsyna, Irene Parada, Willem Sonke, Bettina Speckmann, Ryuhei Uehara, and Jules Wulms. Compacting squares: Input-sensitive in-place reconfiguration of sliding squares. In Artur Czumaj and Qin Xin, editors, *Proceedings of the 18th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2022)*, volume 227 of *LIPICs*, pages 4:1–4:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi: 10.4230/LIPICs.SWAT.2022.4.
- [2] Sebastian Böcker and Jan Baumbach. Cluster editing. In Paola Bonizzoni, Vasco Brattka, and Benedikt Löwe, editors, *Proceedings of the 9th Conference on Computability in Europe (CiE, 2013)*, volume 7921 of *Lecture Notes in Computer Science*, pages 33–44. Springer, 2013. doi: 10.1007/978-3-642-39053-1\_5.
- [3] Carl Brezovec, Gérard Cornuéjols, and Fred W. Glover. Two algorithms for weighted matroid intersection. *Mathematical Programming*, 36(1):39–53, 1986. doi: 10.1007/BF02591988.
- [4] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. ISBN 978-3-319-21274-6. doi: 10.1007/978-3-319-21275-3.
- [5] Jack Edmonds. Submodular functions, matroids, and certain polyhedra. In Michael Jünger, Gerhard Reinelt, and Giovanni Rinaldi, editors, *Combinatorial Optimization - Eureka, You Shrink!, Papers Dedicated to Jack Edmonds, 5th International Workshop, Aussois, France, March 5-9, 2001, Revised Papers*, volume 2570 of *Lecture Notes in Computer Science*, pages 11–26. Springer, 2001. doi: 10.1007/3-540-36478-1\_2.
- [6] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006. ISBN 978-3-540-29952-3. doi: 10.1007/3-540-29953-X.
- [7] Jan Friso Groote and Bas Ploeger. Switching graphs. *International Journal of Foundations of Computer Science*, 20(5):869–886, 2009. doi: 10.1142/S0129054109006930.
- [8] Ulrich Huckenbeck. On paths in networks with valves. In Patrice Enjalbert, Alain Finkel, and Klaus W. Wagner, editors, *Proceedings of the 10th annual Symposium on*

*Theoretical Aspects of Computer Science (STACS, 1993)*, volume 665 of *Lecture Notes in Computer Science*, pages 90–99. Springer, 1993. doi: 10.1007/3-540-56503-5\\_12.

- [9] Ulrich Huckenbeck. On valve adjustments that interrupt all s-t-paths in a digraph. *Journal of Automata, Languages and Combinatorics*, 2(1):19–45, 1997. doi: 10.25596/JALC-1997-019.
- [10] Iyad A. Kanj and Stefan Szeider. Parameterized and subexponential-time complexity of satisfiability problems and applications. *Theoretical Computer Science*, 607:282–295, 2015. doi: 10.1016/J.TCS.2015.08.029.
- [11] Bastian Katz, Ignaz Rutter, and Gerhard J. Woeginger. An algorithmic study of switch graphs. *Acta Informatica*, 49(5):295–312, 2012. doi: 10.1007/S00236-012-0160-4.
- [12] Tuukka Korhonen. A single-exponential time 2-approximation algorithm for treewidth. In *62nd IEEE Annual Symposium on Foundations of Computer Science (FOCS 2021)*, pages 184–192. IEEE, 2021. doi: 10.1109/FOCS52979.2021.00026.
- [13] Bernhard H Korte, Jens Vygen, B Korte, and J Vygen. *Combinatorial Optimization*, volume 1 of *Algorithms and Combinatorics*. Springer, 2011. ISBN 978-3-662-56039-6. doi: 10.1007/978-3-662-56039-6.
- [14] David Lichtenstein. Planar formulae and their uses. *SIAM Journal on Computing*, 11(2):329–343, 1982. doi: 10.1137/0211025.
- [15] Ján Manuch and Daya Ram Gaur. Fitting protein chains to cubic lattice is np-complete. *Journal of Bioinformatics and Computational Biology*, 6(1):93–106, 2008. doi: 10.1142/S0219720008003308.
- [16] Dániel Marx. Can you beat treewidth? *Theory of Computing*, 6(1):85–112, 2010. doi: 10.4086/TOC.2010.V006A005.
- [17] Christoph Meinel. Switching graphs and their complexity. In Antoni Kreczmar and Grazyna Mirkowska, editors, *Proceedings of the 14th Symposium on Mathematical Foundations of Computer Science (MFCS, 1989)*, volume 379 of *Lecture Notes in Computer Science*, pages 350–359. Springer, 1989. doi: 10.1007/3-540-51486-4\\_82.
- [18] Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006. ISBN 9780198566076. doi: 10.1093/ACPROF:OSO/9780198566076.001.0001.
- [19] Klaus Reinhardt. The simple reachability problem in switch graphs. In Mogens Nielsen, Antonín Kucera, Peter Bro Miltersen, Catuscia Palamidessi, Petr Tuma, and Frank D. Valencia, editors, *Proceedings of the 35th Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM, 2009)*, volume 5404 of *Lecture Notes in Computer Science*, pages 461–472. Springer, 2009. doi: 10.1007/978-3-540-95891-8\\_42.

- [20] Stefan Szeider. Finding paths in graphs avoiding forbidden transitions. *Discrete Applied Mathematics*, 126(2-3):261–273, 2003. doi: 10.1016/S0166-218X(02)00251-2.
- [21] Simon Tippenhauer and Wolfgang Muzler. On planar 3-sat and its variants. *Fachbereich Mathematik und Informatik der Freien Universitat Berlin*, 2016.
- [22] Craig A. Tovey. A simplified np-complete satisfiability problem. *Discrete Applied Mathematics*, 8(1):85–89, 1984. doi: 10.1016/0166-218X(84)90081-7.