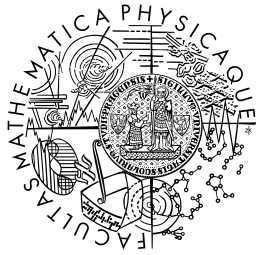


FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS
(updated version)

Martin Dvořák

Minimum 0-Extensions of Graph Metrics

Department of Theoretical Computer Science and Mathematical Logic

Supervisor of the master thesis: RNDr. Jakub Bulín, Ph.D.
Study programme: Computer Science
Study branch: Artificial Intelligence

2021-02-27

I would like to express my gratitude to RNDr. Jakub Bulín, Ph.D. for his patient supervision, to prof. Vladimir Kolmogorov, Ph.D. for his help with the construction of the function g in Section 3.3.2 and his suggestion of a nicer notation for parameter names in my proofs, and to the student affairs departments of both MFF UK and IST Austria for allowing me to finish my master thesis simultaneously with my ongoing postgraduate studies and for supporting me in working from home during the COVID-19 pandemic.

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

Title: Minimum 0-Extensions of Graph Metrics

Author: Martin Dvořák

Department: Department of Theoretical Computer Science and Mathematical Logic

Supervisor: RNDr. Jakub Bulín, Ph.D., Katedra teoretické informatiky a matematické logiky

Abstract: We consider the Minimum 0-Extension Problem for a given fixed undirected graph with positive weights. We study the computational complexity of the threshold decision variant with respect to properties of the fixed graph, in particular modularity and orientability, as defined by Karzanov in [Eur. J. Comb., 19/1 (1998)]. We approach the problem from the viewpoint of the Finite-Valued CSP, which allows us to employ the rich theory that was developed to prove the Dichotomy Conjecture.

On the negative side, we provide an explicit reduction from the Max-Cut Problem to obtain NP-hardness for non-modular graphs. For non-orientable graphs, we express a cost function that satisfies a certain condition which guarantees the existence of an implicit reduction from the Max-Cut Problem. On the positive side, we construct symmetric fractional polymorphisms in order to show that the so-called Basic LP Relaxation can solve two special cases of weighted modular orientable graphs: paths and rectangles.

Keywords: constraint satisfaction, computational complexity, graph metrics, non-modular graphs

Contents

1. Introduction

Constraint Satisfaction Problems (CSPs for short) are an important area of Artificial Intelligence. CSPs are frequently used in planning, scheduling, and game playing. We will study an interesting subclass of binary CSPs, based on graph metrics. The problem is called the "Minimum 0-Extension Problem"[1].

We will generalize the problem from simple graphs to weighted graphs. The generalized problem is equivalent to the "Multifacility Location Problem" which has a natural practical motivation in economics. We will show that small differences in the underlying graph can make huge differences in the computational complexity of the resulting CSP.

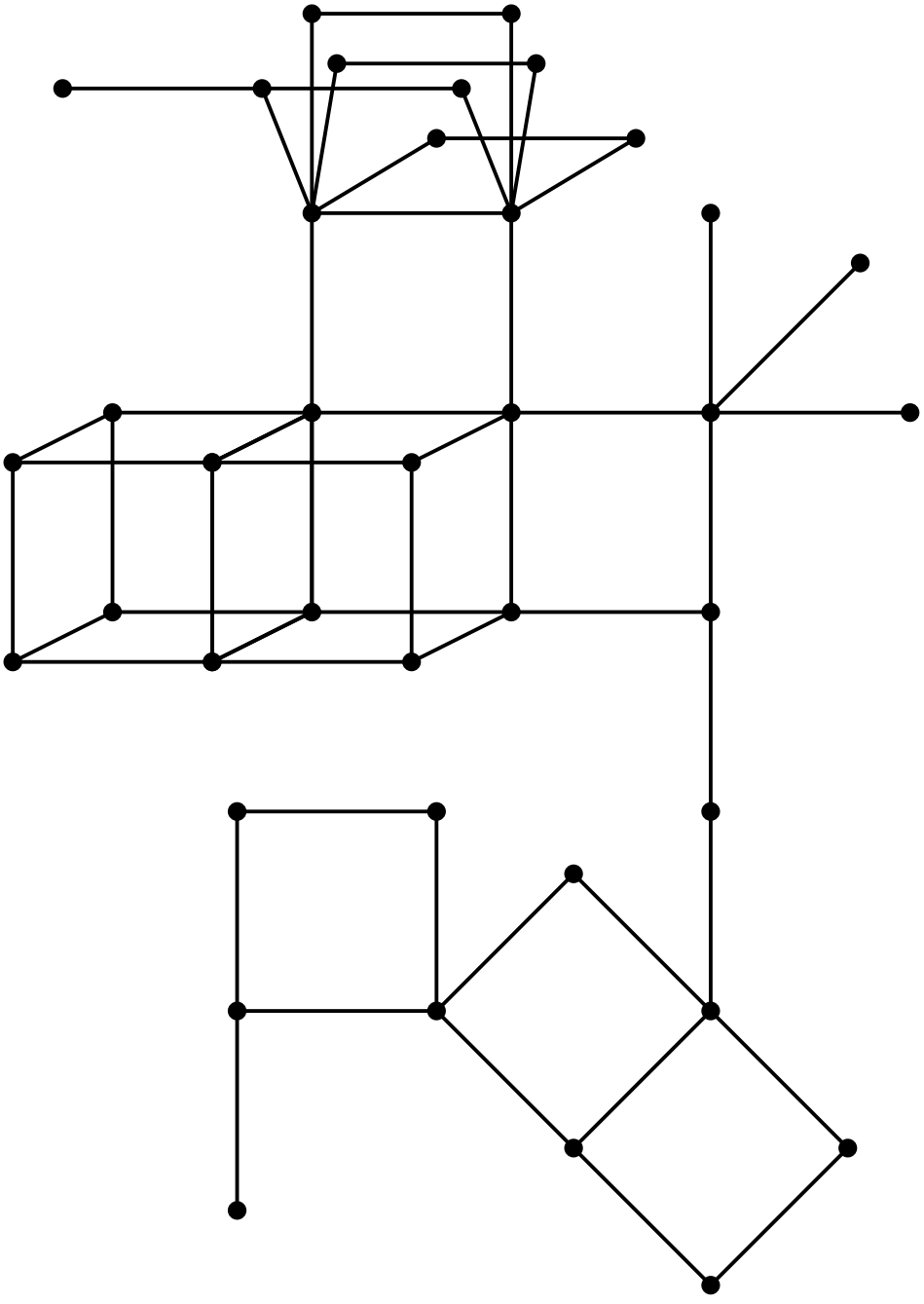


Figure 1: Example of a modular orientable graph

1.1 The Problem

There are several ways how to formulate the Minimum 0-Extension Problem. We will use one formulation here (as the Multifacility Location Problem) but two different formulations (which will be more formal) will follow inside the thesis (Definition 35 and Definition 54). All of them are equivalent (i.e. they posses the same expressive power).

We have a list of facilities, some of which are already built (existing facilities, also referred to as "old facilities"), others are to be located and built later ("new facilities"). We want to minimize the logistic costs. There are two kinds of expenses. First, there are transportation costs between old facilities and new facilities. Second, there are transportation costs between pairs of new facilities. Costs are obtained by the product of the distance and the intensity of traveling (cost weight).

The cost weights are expressed by a binary function c. The distances are expressed by a graph metric d. Let us denote the set of existing facilities by V and the set of new facilities by X.

$$\sum_{x \in X} \sum_{v \in V} \Big(c(x, v) \cdot \mathrm{d}(x, v) \Big) \; + \; \frac{1}{2} \cdot \sum_{x \in X} \sum_{y \in X} \Big(c(x, y) \cdot \mathrm{d}(x, y) \Big)$$

This is the function which we are asked to minimize [2].

The expected form of the solution is a map from X to V, i.e. which old facility should each new facility be built next to. Without this additional condition, the task would be called the Minimum Extension Problem (see Definition 34 for a precise formulation).

⋆ ⋆ ⋆

It is not surprising that the Minimum 0-Extension Problem is (in its full generality) **NP**-complete. We are interested in a finer classification of the computational complexity in order to identify some tractable cases. We will fix the set of the old facilities and their distance matrix, i.e. the weighted graph $G_{\mathrm{w}} = (V, E, \mathrm{w})$.

This now-fixed graph G_{w} will be referred to as the template. The set of new facilities and their cost weights will be always given as the input. We will be interested in the worst-case (over all inputs) time complexity of the Minimum 0-Extension Problem for G_{w} depending on the graph-theoretical properties of the template G_{w}.

We will later see that even small differences in the graph G_{w} can cause dramatic differences in the complexity of the Minimum 0-Extension Problem for G_{w}.

⋆ ⋆ ⋆

The Minimum 0-Extension Problem was previously studied within three different frameworks: (1) graph theory, (2) metric spaces, and (3) constraint satisfaction problems. We will use the third option (CSPs) because it allows a convenient representation of both the instance and its parts.

In particular, we will work with the fixed-language Finite-Valued CSP; see Definition 52 for their detailed description.

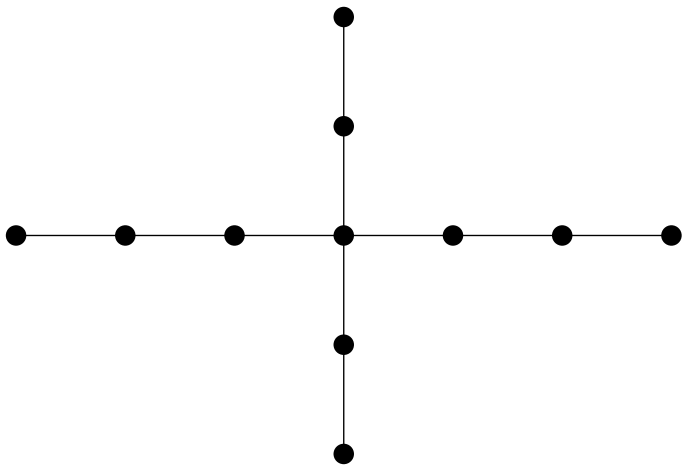
⋆ ⋆ ⋆

It is important to mention that we will use only "good graphs" (connected graphs, with positive weights, without redundant edges; see Definition 32) in place of the templates in the whole thesis. This restriction does not make our results weaker.

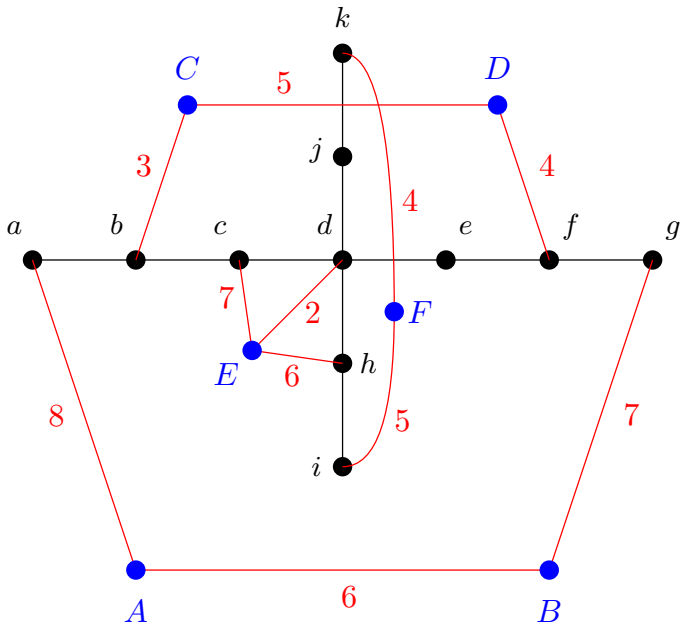
Redundant edges can always be deleted, zero-weight edges can always be contracted, and negative-weight edges do not make any sense in the Multifacility Location Problem. If the template is not a connected graph, we can first easily check for admissibility (i.e. no pair of distinct components from the template gets connected by the instance's edges) and then optimize the cost in each component separately. Therefore, the assumption of "good graphs" does not bring any loss in the expressive power.

1.2 Examples

Let us have a look at two concrete examples. They are both based on the same template — a specific tree on 11 vertices. It is a simple graph, or in other words, all edges have weight 1. Chemists might call it "4,4-diethyl-heptane".



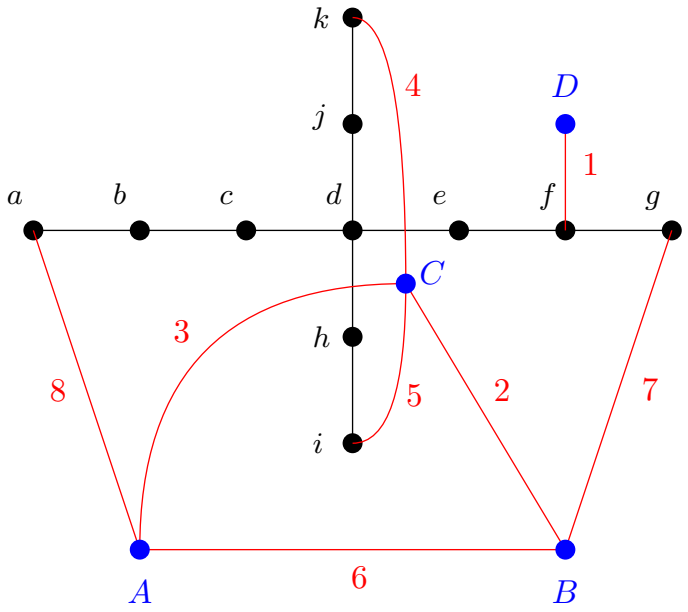
The black graph is this template. Instances are described by blue vertices and red edge weights. The blue vertices must be mapped to the black vertices.



This example is easy to solve by hand. The blue vertex A gets mapped to the black vertex a. The vertex B gets mapped to g. The intuitive reason behind this is that the penalty for the distance between the instance vertex and the template vertex is higher than the penalty for the distance between the instance vertex and the other instance vertex.

The vertices C and D get both mapped to f. Here, it is more expensive to map the instance vertices to different template vertices than to map the instance vertex to a suboptimal template vertex.

The remaining two vertices can be optimized independently of the rest of the instance. The vertex E gets mapped to d. The vertex F gets mapped to i.



Even though the main part is similar to previous example, it is optimal to map all three interconnected blue vertices A, B, and C to the middle vertex d. The remaining blue vertex D gets mapped to f.

1.3 Motivation

Location theory has already been studied for more than 300 year. One of the oldest questions, formulated (probably) first by Pierre de Fermat [3] and solved (probably) first by Evangelista Torricelli, asks for a point X that minimizes the sum of distances $|XA|+|XB|+|XC|$ given (a triangle) points A, B, C as the input [4].

Since 1957, the field of location theory has been significantly expanding to provide answers to many different kinds of questions [5]. Researchers have been studying so-called center problems, asking to minimize the longest distance, to minimize the longest travel time, or to minimize the maximum transportation cost; so-called median problems, asking to minimize the sum of all transportation costs; and so-called plant location problems, asking to minimize the sum of all setup costs plus all transportation costs [5].

The Minimum 0-Extension Problem has its undeniable use in Economics because it is a useful type of the median problem; in particular, it is classified as "discrete p-median problem with mutual communication" [5].

⋆ ⋆ ⋆

Imagine that we own a chemical factory and we want to expand the factory in order to produce greater amount and/or greater diversity of monetizable chemicals. We, of course, want to reduce our expenses and thus increase our profit.

There are many expenses that are beyond our control. However, we can influence the transportation cost within our factory by choosing a convenient location for our new buildings.

We want to minimize the transportation cost between each new building and all old buildings that will transport material from and/or to the new building and, because we are planning location of several new building at once, we can also optimize for low transportation costs between pairs of new buildings.

Depending on other conditions that apply to our scenario, this situation can be best modeled by the Minimum Extension Problem, by the Minimum 0-Extension Problem, or by the Metric Labeling Problem [6].

⋆ ⋆ ⋆

Apart from its usage in Economics, the Minimum 0-Extension Problem has practical applications in Computer Science, especially in Machine Learning. Many tasks in the data classification make use of semantic object clustering.

Consider the task of hypertext categorization [7]. Each piece of text can be characterized by a set of labels. In addition to that, there are connections to other pieces of text. Existence of a hypertext link suggests a probable closeness of the topics. By solving the Minimum 0-Extension Problem, we combine the information about this piece of text itself with the information about its relationship to other pieces of text [7]. Other areas of similar practical applications include image segmentation and biometric analysis [7].

A textbook example of the usage of the Minimum 0-Extension Problem is the task of restoring an image degraded by noise [8]. Kleinberg and Tardos describe the approach as follows [7]:

> *'We are given a large grid of pixels; each pixel has a "true" intensity that we are trying to determine, and an "observed" intensity that is the result of corruption by noise. We would like to find the best way to label each pixel with a (true) intensity value, based on the observed intensities. Our determination of the "best" labeling is based on the trade-off between two competing influences: We would like to give each pixel an intensity close to what we have observed; and — since real images are mainly smooth, with occasional boundary regions of sharp discontinuity — we would like spatially neighboring pixels to receive similar intensity values.'*

1.4 The CSP framework

In this work, we use the formalism of the Constraint Satisfaction Problems. It is a wide framework for discrete optimization problems.

Informally speaking, the CSP deals with various problems in which we want to assign values to some variables such that certain conditions are satisfied. An example can be a distribution of T-shirts of various sizes to a group of people where each person wants to get a T-shirt from some size range that fits them.

In terms of the CSP jargon, we can translate it as follows. People will be modeled by variables. T-shirt sizes correspond to labels. Requirements are expressed by constraints. A map from the set of people to the set of sizes, saying who gets which T-shirt, corresponds to a labeling.

$\star \quad \star \quad \star$

There are many types of the CSP which deal with admissibility and/or optimality in various ways. Most of the previously-studied problems from combinatorial optimization can be modeled using a suitable kind of the CSP.

The Crisp CSP (often called just the CSP) deals with problems like SAT. We are given a set of "constraints" (conditions) and we search for a "labeling" (an assignment of values to variables) such that all constraints are satisfied.

$$(x \vee y) \ \wedge \ (\neg x \vee \neg y) \ \wedge \ (x \vee \neg y \vee \neg z) \ \wedge \ (\neg x \vee y \vee z)$$

This instance has two solutions, namely the labeling $x = 0, y = 1, z = 0$ and the labeling $x = 1, y = 0, z = 1$.

The domain (the set of possible values to be assigned to the variables) can have more than two elements. For instance, the 3-Coloring is an example of the Crisp CSP and its domain contains 3 elements.

The Max CSP deals with problems like MAX-SAT. We are again given a list of constraints but now we want to satisfy the maximum number of them, not necessarily all.

$$(x \vee y),\ (\neg x \vee \neg y),\ (x \vee \neg y \vee \neg z),\ (\neg x \vee y \vee z),\ (x \vee z),\ (y \vee \neg z)$$

If this were an instance of SAT, it would be unsatisfiable. However, we can satisfy 5 out of 6 clauses, for instance by a solution to the previous problem.

Clearly, MAX-SAT is no easier than SAT. If we can solve MAX-SAT, we can easily solve SAT by comparing the maximum number of satisfiable clauses with the total number of clauses. On the other hand, a negative answer to a SAT instance does not tell much about the solution of the corresponding MAX-SAT instance. Under certain assumptions, we can say that MAX-SAT is even strictly harder than SAT. It is known that 2-SAT is in **P** whereäs MAX-2-SAT is **NP**-hard.

The same observation can be stated in general. The Max CSP is at least as hard as the Crisp CSP with the same constraints. Another example, in which the "Max version" is strictly harder, under certain assumptions, is the Max-Cut Problem (**NP**-hard Max CSP) whose corresponding Crisp CSP (determining whether a given graph is bipartite) is solvable in linear time.

The Max CSP could also be weighted, that is, each condition is given its importance (a positive rational weight). The goal is to maximize the sum of weights of all satisfied conditions. This is sometimes called "Weighted CSP". Do not mistake it for "Valued CSP" which is discussed below.

The Finite-Valued CSP generalizes it further. Crisp constraints are replaced by rational-valued cost functions. They are no longer restricted to the two values 0 and 1.

From now on, the standard convention is to minimize the sum. We will work with minimization in this thesis as well. The Minimum 0-Extension Problem is an example of the Finite-Valued CSP. Rational values are used because there does not exists any efficient encoding of general real numbers in a computer. Moreover, real values can be approximated by rational numbers very well.

Crisp conditions and rational-valued cost functions can be combined. The first intermediate level is known as the Min-Cost-Hom. In the Min-Cost-Hom Problem, all rational-valued cost functions are unary but crisp conditions can have any arity. The crisp conditions are sometimes called "hard constraints"; the rational-valued cost functions are sometimes called "soft constraints". An example follows.

minimize: $3x + 5y + 4z$
under the conditions: $(x \vee y) \wedge (\neg x \vee \neg y) \wedge (x \vee \neg y \vee \neg z) \wedge (\neg x \vee y \vee z)$

The optimum labeling is $x = 0, y = 1, z = 0$.

The General-Valued CSP is the most general type of the CSP discussed here. It is based on the notion of general-valued cost functions. They are functions of arbitrary arity that can output infinity for some inputs and various rational values for other inputs. The General-Valued CSP generalizes both the Finite-Valued CSP and the Min-Cost-Hom.

$\star \quad \star \quad \star$

We can restrict what concrete constraints can appear in the instance. Many researchers are interested in the computational complexity of these so-called fixed-language CSPs [9] [10]. For example, 3-SAT is defined by the language $\{ \{0,1\}^3 \setminus (0,0,0),\ \{0,1\}^3 \setminus (0,0,1),\ \{0,1\}^3 \setminus (0,1,1),\ \{0,1\}^3 \setminus (1,1,1) \}$.

In this framework, the Minimum 0-Extension Problem is not one problem. It is a class of problems. Every graph defines its own Minimum 0-Extension Problem where the template is fixed and the input graph is mapped onto it. Every graph has a unique finite-valued constraint language. We study the complexity of the Minimum 0-Extension Problem with respect to the properties of the fixed template that defines this finite-valued constraint language.

1.5 History

Several special cases of the Minimum 0-Extension Problem were studied before. In the following short survey, we will assume that the template is a simple graph unless stated otherwise.

In 1956, Ford and Fulkerson [11] studied the Maximum s,t-flow Problem and the (dual) Minimum s,t-cut Problem. The latter is equivalent to the Minimum 0-Extension Problem where the template is K_2 (see Definition 12). Their algorithm provides a constructive proof that the Minimum 0-Extension Problem for K_2 is in **P** (see Definition 41).

In 1978, Picard [12] studied the Minimum 0-Extension Problem for trees (see Definition 10). The generalization to forests is straightforward. He thereby proved that the Minimum 0-Extension Problem for acyclic graphs is in **P**. See also [13] (a follow-up report written by Kolen in 1979) for an alternative formulation of the same algorithm and his later book [5] for a clearer explanation and more context.

In 1994, Dahlhaus, Papadimitriou, Seymour, Johnson, and Yannakakis [14] studied the Multiterminal Cut Problem (see Definition 56). This problem is equivalent to the Minimum 0-Extension Problem for K_n. They proved that it is **NP**-complete (see Definition 48) for all $n \geq 3$.

In 1996, Chepoi [2] proved that the Minimum 0-Extension Problem for median graphs (see Definition 64) is in **P**. This generalized the positive result of Picard and Kolen to other graph classes, e.g. hypercubes, grids, and cartesian products of trees [15].

In 1998, Karzanov [1] stated the Minimum 0-Extension Problem in its full generality (for simple graphs) and established the following important results. If the template satisfies all three conditions: that it is bipartite (see Definition 11), that it is orientable (for his own definition of orientability that deals only with cycles of length four; see Definition 18), and that there is no isometric cycle (see Definition 8) of length greater than five, then the Minimum 0-Extension Problem for this template is in **P**. He also provided an equivalent condition saying that, if the template is both heredity modular (see Definition 15) and orientable (see Definition 18), then it is in **P**. He also established two hard cases. If the template is non-modular (see Definition 14), then it is **NP**-complete. And also, if the template is

non-orientable (see Definition 19), then it is **NP**-complete as well. Karzanov used a reduction from the Max-Cut Problem (see Definition 59) in his both hardness proofs. His method is based on previous hardness proofs for K_n; in particular, it is based on the "submodularity counterexample" (a counterexample to an incorrect algorithm for the 3-Terminal Cut Problem) from [14]. These results were a significant improvement over the results from 1978 and 1994 (see Observation 20 and Observation 21) but were independent of the Chepoi's result from 1996 (see Remark 65). In remained an open question of whether the Minimum 0-Extension Problem for modular and orientable graphs that are not hereditary modular (see Observation 16) is in **P**.

In 2012, Hirai [16] answered this question positively. He provided an elaborate algorithm based on Linear Programming (see Definition 96) which solved the Minimum 0-Extension Problem for all graphs that are both modular and orientable (see Figure 1). This generalized both the Chepoi's result from 1996 and the positive Karzanov's results from 1998.

When I was introduced to this problem by professor Vladimir Kolmogorov [17] during my internship at IST Austria in 2019, I incorrectly believed this was the state of the art at that moment. I was unaware of a publication by Karzanov from 2004 that had generalized the hard cases to weighted graphs. I started an independent research on this generalization.

⋆ ⋆ ⋆

In parallel with the research on the Minimum 0-Extension Problem, there was an important research on the CSP complexity in general. Computational complexity as such was still a rapidly-developing area. In 1975, Ladner [18] discovered that, assuming $\mathbf{P} \neq \mathbf{NP}$, there must be some **NP**-intermediate problems. This raised a question of whether the fixed-language CSP can be **NP**-intermediate. This question lead to a fruitful decades-long research programme on the CSP dichotomy.

There are many dichotomy theorems with various degrees of generality. In 1978, Schaeffer [19] solved the case of the Boolean CSP (generalizes SAT) by giving a list of polymorphisms such that if the language admits at least one of them, the corresponding CSP is in **P**; otherwise it is **NP**-complete. The Dichotomy Conjecture was formulated — that the fixed-language CSP is always either in **P**, or it is **NP**-complete; or in other words, that it is never **NP**-intermediate.

In 1990, Hell and Nešetřil [20] solved the case of Graph Homomorphism Problems (a different problem than what we study in this thesis; it generalizes the k-Coloring) by stating that, for a fixed codomain (template), the Graph Homomorphism Problem is in **P** if the graph (template) is bipartite; otherwise it is **NP**-complete.

In 2012, Thapper and Živný [21] solved the case of every Finite-Valued CSP by stating that the VCSP of a fixed finite-valued language is in **P** if the language admits a symmetric binary fractional polymorphism (see Definition 95); otherwise it is **NP**-complete.

In 2017, Zhuk [22] solved the case of all Crisp CSP by stating that the fixed-language Crisp CSP is in **P** if the fixed language admits a weak

near-unanimity polymorphism; otherwise it is **NP**-complete. Almost at the same time, Bulatov [23] provided an independent proof of the same theorem (both were announced in Spring 2017). This result significantly generalized the old result of Shaeffer from 1978 and provided a positive answer to the Dichotomy Conjecture by Feder and Vardi [24] from 1993.

In 2017, Kolmogorov, Rolínek, and Krokhin [25] connected the results and proved the dichotomy theorem for every fixed-language General-Valued CSP. In reality, they started this research prior to the announcement of Zhuk's (and Bulatov's) result but their proof is based on the assumption that the Crisp CSP Dichotomy has already been established.

⋆ ⋆ ⋆

The main motivation for the study of the CSP has long been its usefulness in artificial intelligence [26] [27]. As one of the first practical application of the CSP, we would like to mention the scene labeling task [28] [29] studied in the 1970s (if we ignore some "prehistoric problems", such as the 8-queen problem, which had been studied long before the CSP was formulated; even though it can now be viewed as an instance of the CSP, although we cannot require the CSP language to be fixed if we want to model the general N-queen problem by the CSP).

As of 1992, the CSP was used in planning, scheduling, temporal and spatial reasoning, causal reasoning, computer vision, natural language processing, and diagnostic reasoning [30]. The CSP is also used in the design of algorithms for playing various games [31].

The theory of the CSP is also connected to the foundations of the database theory [32] [33]. As a matter of fact, the CSP in the most typical definition (i.e. the Crisp CSP with constraints defined by tables given on the input) is equivalent to asking whether a NATURAL JOIN in the SQL database [34] is non-empty.

Most recently, the CSP has been heavily studied using the tools of universal algebra (such as the theory of clones) [35] [36] [37].

1.6 Methods and organization of this work

The complexity of the hard cases will be established by polynomial reductions from the Max-Cut Problem (see Definition 59). Our first reduction will be a classical explicit reduction. Our second reduction will be more algebraic.

The complexity of the easy cases will be established using the Basic LP Relaxation (see Definition 98).

⋆ ⋆ ⋆

Chapter 2 defines the terms used throughout the thesis and explains the basic relations between them. The majority of unusual terms is defined in Sections 2.4, 2.6, and 2.7 in details. Chapter 3 generalizes the case of non-modular graphs. Chapter 4 generalizes the case of non-orientable graphs. Chapter 5 attempts to generalize the case of modular orientable graphs but ends up solving only two small subclasses of them.

2. Preliminaries

Even though we are standing on the grounds of a formal science, the terminology and the formal notations are not fully consistent among all authors. Therefore, it is our best interest that we define the terms and symbols used in the thesis before we present our discoveries and thereby we hope that this work will be sufficiently self-contained. By defining "everything" we use, we also allow our work to be accessible for a slightly wider audience.

Yes, we expect the reader is familiar with elementary algebra and knows the commonly-used symbols from the first-order logic and set theory. On the other hand, we do not assume that the reader is educated in the areas of graph theory and the Constraint Satisfaction Problems.

Apart from the definitions, this chapter contains many observations that should be obvious immediately after reading the definitions.

We aim to be as formal as possible. However, we occasionally resort to a compromise between the mathematical rigor and readability.

In order to minimize the risk of confusion, we write our formulas in the prenex normal form [38]. We omit brackets around quantifiers and we instead write small glues around them for better readability. A colon is used as a delimiter between the prefix and the matrix.

2.1 Basic terminology of graph theory

This section introduces the most fundamental terms in graph theory and declares the basic formalism exactly as it will be used throughout the thesis.

Definition 1. Let V be a finite set. Let E be a set containing some pairs of elements from V, that is $E \subseteq \binom{V}{2}$. We say that $G = (V, E)$ is a *graph*. The set V is called *vertices*. The set E is called *edges*.

Definition 2. Let $G = (V, E)$ be a graph. Let w be a function $E \to \mathbb{Q}_0^+$. We say that $G_\mathrm{w} = (V, E, \mathrm{w})$ is a *weighted graph*, w is its *weight function* and, for any edge $e \in E$, the value $\mathrm{w}(e)$ is the *weight* of the edge e.

For graphs which are not weighted, we will call them *simple* graphs, we implicitly assume a constant weight function (in case we need to use weights in our calculations), i.e. we define all weights of edges to be equal to one.

We say that G_w is a *graph with positive weights* if all edges have strictly positive weights. Every simple graph can be viewed as a graph with positive weights.

If we say only "graph", as we did in Definition 1, we promise that things do not break apart when we plug in a weighted graph. We will use the term "weighted graph" only when working with the weight function w explicitly.

Definition 3. Let $G = (V, E)$ and $H = (W, F)$ be graphs. We say that H is a *subgraph* of G, denoted $H \leq G$, if H can be obtained from G by deleting edges and deleting vertices. Formally, $W \subseteq V$ and $F \subseteq E$.

In the case of weighted graphs $H_{\mathrm{w}'} \leq G_\mathrm{w}$, the weight function w' of $H_{\mathrm{w}'}$ must correspond to the weight function w of G_w restricted from E to F, i.e. $\forall e \in F : \mathrm{w}'(e) = \mathrm{w}(e)$.

Definition 4. Let $G = (V, E)$ be a graph. We say that $(v_1, v_2, \ldots, v_k)$ is a *path* in G if $\forall i \in \{2, 3, \ldots, k\} : \{v_{i-1}, v_i\} \in E$. We say that $(v_1, v_2, \ldots, v_k)$ is a path from v_1 to v_k of length $k - 1$. The path is *simple* if $i \neq j$ implies $v_i \neq v_j$.

Definition 5. Let $G = (V, E)$ be a graph. We say that the graph G is *connected* if for all pairs of vertices $u, v \in V$, there exists a path from u to v in G.

Definition 6. Let $G_{\mathrm{w}} = (V, E, \mathrm{w})$ be a weighted graph. Let $u, v \in V$ be any two vertices. We say that the *distance* between vertices u and v, denoted by $\mathrm{d}_{G_{\mathrm{w}}}(u, v)$ or just $\mathrm{d}(u, v)$ for brevity, is equal to the minimum sum of edge weights on a path from u to v.

$$\mathrm{d}_{G_{\mathrm{w}}}(u, v) = \min_{\substack{(x_1, x_2, \ldots, x_k) \\ \text{path in } G_{\mathrm{w}}, \\ u = x_1, v = x_k}} \sum_{i=2}^{k} \mathrm{w}(x_{i-1}, x_i)$$

Just a brief comment about the "corner" cases. Note that $\mathrm{d}(u, u) = 0$. Also note that $\mathrm{d}(u, v) = \infty$ if and only if the vertex v cannot be reached from u by any path (which implies that the graph G_{w} is not connected).

Definition 7. Let $G = (V, E)$ be a graph. Let $(v_1, v_2, \ldots, v_k, v_{k+1})$ be a path in G. If $v_1 = v_{k+1}$, then we say that $(v_1, v_2, \ldots, v_k)$ is a *cycle* in G of length k. The cycle is *simple* if $k \geq 3$ and every $1 \leq i < j \leq k$ implies $v_i \neq v_j$.

Definition 8. Let $G = (V, E)$ be a graph. Let $(c_1, c_2, \ldots, c_k)$ be a simple cycle in G. Let us denote the cycle as a graph $C = (\{c_1, c_2, \ldots, c_k\}, \{\{c_1, c_2\}, \{c_2, c_3\}, \ldots, \{c_{k-1}, c_k\}, \{c_k, c_1\}\})$. We say that the cycle $(c_1, c_2, \ldots, c_k)$ is *isometric* in G, if $\forall\, i, j \in \{1, 2, \ldots, k\} : \mathrm{d}_G(c_i, c_j) = \mathrm{d}_C(c_i, c_j)$.

Definition 9. Let $G = (V, E)$ and $H = (W, F)$ be graphs such that $H \leq G$. We say H is an *isometric* subgraph of G if $\forall\, u, v \in W : \mathrm{d}_G(u, v) = \mathrm{d}_H(u, v)$.

The similarity of the terms is not accidental. An isometric cycle in G (see Definition 8) induces an isometric subgraph of G (see Definition 9).

2.2 Special classes of graphs

This section defines several special classes of graphs, out of which most important for us will be non-modular graphs and non-orientable graphs.

Definition 10. Let $G = (V, E)$ be a graph. If G is connected and there is no simple cycle in G, we say that G is a *tree*.

Definition 11. Let $G = (V, E)$ be a graph. Let us decompose the graph into partitions (two subsets of the vertex set): $A \subseteq V,\ B = V \setminus A$, such that $\forall\, a_1, a_2 \in A : \{a_1, a_2\} \notin E$ and similarly $\forall\, b_1, b_2 \in B : \{b_1, b_2\} \notin E$.

In other words, all edges of G lie between partitions, no edge lies inside a partition. If such sets A, B exist, we say that the graph G is *bipartite*.

Definition 12. Let $n \in \mathbb{N}$ and $|V| = n$. We say that $K_n = (V, \binom{V}{2})$ is a *complete graph* on n vertices.

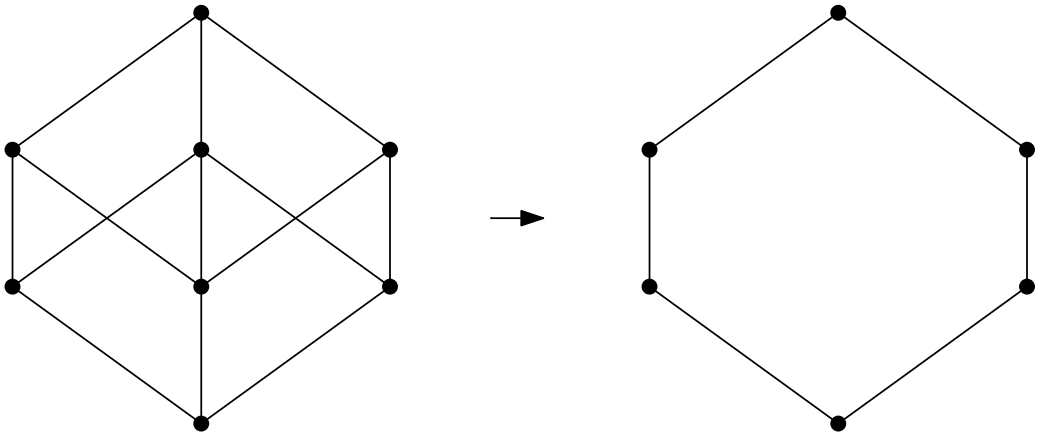
Definition 13. Let $G = (V, E)$ be a graph. We say that G is *modular* [1] if:

$$\forall\, u_1, u_2, u_3 \in V\ \exists v \in V\ \forall\, (1 \leq i < j \leq 3) :\ \mathrm{d}(u_i, v) + \mathrm{d}(v, u_j) = \mathrm{d}(u_i, u_j)$$

Definition 14. Let $G = (V, E)$ be a graph. If G is not modular, we say that G is *non-modular* [1].

Definition 15. Let $G = (V, E)$ be a simple graph. If every isometric subgraph of G is modular, then we say that G is *hereditary modular* [1].

Observation 16. If G is a hereditary modular graph, then G is also modular, because G is a "trivial" isometric subgraph of G. On the other hand, not every modular graph is hereditary modular, see this example [39]:



Definition 17. Let $G = (V, E)$ be a graph. Let $\vec{E} \subseteq V^2$ be a set of ordered tuples of vertices. We say that $\vec{E}$ is an *orientation* of E if both of the following conditions hold.

1. $\forall\, \{u, v\} \in E\ :\ (u, v) \in \vec{E}\ \vee\ (v, u) \in \vec{E}$
2. $\forall\, (u, v) \in \vec{E}\ :\ \{u, v\} \in E\ \wedge\ (v, u) \notin \vec{E}$

In such a case, we additionally define an oriented graph $\vec{G} = (V, \vec{E})$.

Definition 18. Let $G = (V, E)$ be a graph. We say that G is *orientable* [1] if there exists an orientation $\vec{E}$ such that each simple cycle of length 4 in G, denoted by (a, b, c, d), has an orientation in $\vec{G}$ such that both of the following conditions hold.

1. $(a, b) \in \vec{E} \iff (d, c) \in \vec{E}$
2. $(b, c) \in \vec{E} \iff (a, d) \in \vec{E}$

Definition 19. Let $G = (V, E)$ be a graph. If G is not orientable, we say that G is *non-orientable* [1].

Observation 20. Every tree is a bipartite graph, an orientable graph, and a (hereditary) modular graph (see Definitions 10, 11, 15, 18).

Observation 21. For $n \geq 4$, the complete graph K_n is a connected graph, a non-modular graph, and a non-orientable graph (see Definitions 5, 12, 14, 19).

2.3 Semimetrics

In this section, we explain the basic terms connected to metric spaces and their relationship with graph theory.

In addition to that, we define the notion of "good graph" which will be used in all templates for the Minimum 0-Extension Problem from now on.

Definition 22. Let X be a set. We say that a function $m : X \times X \to \mathbb{R}_0^+$ is a *metric* on X if:

1. $\forall\, a, b \in X :\ m(a, b) = 0 \iff a = b$
2. $\forall\, a, b \in X :\ m(a, b) = m(b, a)$
3. $\forall\, a, b, c \in X :\ m(a, b) + m(b, c) \geq m(a, c)$

Definition 23. Let X be a set. We say that a function $m : X \times X \to \mathbb{R}_0^+$ is a *semimetric* on X if:

1. $\forall\, a \in X :\ m(a, a) = 0$
2. $\forall\, a, b \in X :\ m(a, b) = m(b, a)$
3. $\forall\, a, b, c \in X :\ m(a, b) + m(b, c) \geq m(a, c)$

This m is typically called "pseudometric" [40]. However, we follow Karzanov's terminology [1] that calls it a semimetric.

Observation 24. Every metric is a semimetric. The only different between Definition 22 and Definition 23 is in relaxing the first condition. In the case of a semimetric, only one implication is required in the first condition, thus there can be two distinct points in zero distance.

Definition 25. Let $X \subseteq Y$ be sets. Let m be a semimetric on X. Let μ be a semimetric on Y. We say that μ is an *extension* of m if it satisfies [1]:

$$\forall\, a, b \in X : m(a, b) = \mu(a, b)$$

We can rephrase the definition informally as follows. If m measures distances between elements of X and μ measures distances between elements of its superset Y, then they agree on the distances among all pairs in X, so they have the same "unit size" and one cannot find a shortcut between two points in X by going through some points outside of X.

Definition 26. Let $G_{\mathrm{w}} = (V, E, \mathrm{w})$ be a connected weighted graph. We say that the distance function $\mathrm{d}_{G_{\mathrm{w}}}$ is a *graph metric*.

Observation 27. If $G_{\mathrm{w}} = (V, E, \mathrm{w})$ is a connected weighted graph, then its graph metric $\mathrm{d}_{G_{\mathrm{w}}}$ is a semimetric on V. It is easy to verify all three axioms.

Note that we say "graph metric" (since it is the standard term) even though it is only guaranteed to be a semimetric (some edges can have zero weight).

Observation 28. Let G, H be connected simple graphs. If H is an isometric subgraph of G, then the graph metric d_G is an extension of the graph metric d_H.

Definition 29. Let $X \subseteq Y$ be sets. Let m be a semimetric on X. Let μ be a semimetric on Y. We say [1] that μ is a *0-extension* of m, if μ is an extension of m such that:

$$\forall a \in Y \; \exists b \in X : \; \mu(a, b) = 0$$

In simple terms, the definition says that μ is an extension of m which attaches each point in Y to a point in X.

As a result, if Y is a finite set, then the distance matrix for Y is the same as the distance matrix for X where some rows and some columns are repeated more than once.

Definition 30. Let $G_{\mathrm{w}} = (V, E, \mathrm{w})$ be a weighted graph. Let $e \in E$ be an edge of G_{w}. We say that e is a *redundant* edge if the graph metric stays unchanged in the case of removing the edge e.

$$\forall\, u, v \in V \;\; : \;\; \mathrm{d}_{G_{\mathrm{w}}}(u, v) = \mathrm{d}_{(G_{\mathrm{w}} - e)}(u, v)$$

Observation 31. Let $G_{\mathrm{w}} = (V, E, \mathrm{w})$ be a weighted graph and $\{a, b\} = e \in E$ an edge of G_{w}. The condition that e is non-redundant is equivalent to stating that $\mathrm{w}(e) < \mathrm{d}_{(G_{\mathrm{w}} - e)}(a, b)$. All edges in all simple graphs trivially satisfy this condition.

Definition 32. Let $G_{\mathrm{w}} = (V, E, \mathrm{w})$ be a graph with positive weights such that no edge $e \in E$ is redundant and the graph G is connected. We say that G_{w} is a *good* graph.

Observation 33. Every connected simple graph is a good graph.

2.4 Extension problems

This section contains a formal definition of the Minimum Extension Problem and the Minimum 0-Extension Problem, which was outlined in Section 1.1.

Definition 34. Let $G_{\mathrm{w}} = (V, E, \mathrm{w})$ be a good graph. This graph will be fixed including its weight function. The *Minimum Extension Problem* for G_{w} is defined as follows [1]. The input describes a simple graph $K = (X, L)$ such that $G \leq K$, a rational number N, and a cost function $c : L \to \mathbb{Q}_0^+$. The task is to assign weights $\widetilde{\mathrm{w}}$ to edges in L such that $\mathrm{d}_{K_{\widetilde{\mathrm{w}}}}$ becomes an extension of $\mathrm{d}_{G_{\mathrm{w}}}$ and the weighted sum is low, i.e. $\sum_{e \in L} c(e) \cdot \widetilde{\mathrm{w}}(e) \leq N$. We say that (K, c, N) is an instance of the Minimum Extension Problem for G_{w}.

Definition 35. Let $G_{\mathrm{w}} = (V, E, \mathrm{w})$ be a good graph. This graph will be fixed including its weight function. The *Minimum 0-Extension Problem* for G_{w} is defined as follows [1]. The input describes a simple graph $K = (X, L)$ such that $G \leq K$, a rational number N, and a cost function $c : L \to \mathbb{Q}_0^+$. The task is to assign weights $\widetilde{\mathrm{w}}$ to edges in L such that $\mathrm{d}_{K_{\widetilde{\mathrm{w}}}}$ becomes a 0-extension of $\mathrm{d}_{G_{\mathrm{w}}}$ and the weighted sum is low, i.e. $\sum_{e \in L} c(e) \cdot \widetilde{\mathrm{w}}(e) \leq N$. We say that (K, c, N) is an instance of the Minimum 0-Extension Problem for G_{w}.

The only difference between Definition 34 and Definition 35 is that, in the Minimum 0-Extension Problem, we require a 0-extension of the semimetric (see Definition 29), but in the Minimum Extension Problem, we only require an extension of the semimetric (see Definition 25).

2.5 Computational complexity

This section introduces the most fundamental complexity classes within **NP**. We assume that the reader is familiar with Turing Machines, at least on an intuitive level. For a formal definition, use any reasonable textbook on theoretical computer science, for example [41].

We work with Turing Machines because they are used in the standard terminology. Nevertheless, Random Access Machines can work for the purposes of this thesis as well as, if not better than, Turing Machines. In order to justify this claim, we only need the overhead to by polynomial (which is easy to show in both directions).

Definition 36. Let Σ be a finite set, later referred to as an alphabet. Let L be a set of words, $L \subseteq \Sigma^*$. We say that "deciding whether a given word $w \in \Sigma^*$ belongs to L" is a *decision problem*. If $w \in L$, the answer is YES. If $w \notin L$, the answer is NO. The length of the word w, which is synonymous with the *input size*, is denoted by $|w|$.

Note that L can be an infinite set (and it indeed is for all interesting problems; if L or its complement is finite, then its decision problem can be decided in constant time).

Definition 37. Let f and g be functions $\mathbb{N} \to \mathbb{N}$. We define the *Big-$\mathcal{O}$* relation as $f(n) \in \mathcal{O}(g(n))$ if:

$$\exists K \in \mathbb{R}^+ \; \forall n \in \mathbb{N} : \; f(n) \leq K \cdot g(n)$$

For example $(5n^3 + 9n - 58) \in \mathcal{O}(n^3)$ can be proved using $K \coloneqq 14$.

Definition 38. Let f be a function $\mathbb{N} \to \mathbb{N}$. Let M be any Deterministic Turing Machine that accepts exactly the set $A \subseteq \Sigma^*$. If the number of steps needed by M for a computation over an input $w \in \Sigma^*$ always lies in $\mathcal{O}(f(|w|))$, then A belongs to the class DTIME(f).

If A belongs to the class DTIME(id), we informally say that A can be computed in linear time.

Definition 39. Let f be a function $\mathbb{N} \to \mathbb{N}$. Let N be any Nondeterministic Turing Machine that accepts exactly the set $A \subseteq \Sigma^*$. If the number of steps needed by N on every computation branch over an input $w \in \Sigma^*$ always lies in $\mathcal{O}(f(|w|))$, then A belongs to the class NTIME(f).

Observation 40. For each function $f : \mathbb{N} \to \mathbb{N}$, we see that the class DTIME(f) is a subset of the class NTIME(f).

Definition 41. The *deterministic Polynomial* class **P** is defined as follows.

$$\mathbf{P} = \bigcup_{k \in \mathbb{N}} \mathrm{DTIME}(n^k)$$

If $L \in \mathbf{P}$, we say that L can be *computed* in polynomial time.

Definition 42. The *Nondeterministic Polynomial* class **NP** (sometimes incorrectly referred to as "Non-Polynomial class") is defined as follows.

$$\mathbf{NP} = \bigcup_{k \in \mathbb{N}} \mathrm{NTIME}(n^k)$$

If $L \in \mathbf{NP}$, we say that L can be *verified* in polynomial time.

Remark 43. We see $\mathbf{P} \subseteq \mathbf{NP}$, but the other inclusion (whether $\mathbf{NP} \subseteq \mathbf{P}$) is an open question. Nobody currently knows whether $\mathbf{P} = \mathbf{NP}$.

Definition 44. Let A and B be sets defining decision problems. Let f be a function $\Sigma^* \to \Sigma^*$. We say that f is a *reduction* from A to B if:

$$\forall w \in \Sigma^* : \; w \in A \iff f(w) \in B$$

Definition 45. Let f be a reduction from A to B. We say that f is a *polynomial reduction* from A to B if f is computable in polynomial time.

Observation 46. If $f : \Sigma^* \to \Sigma^*$ is a polynomial reduction, then $|f(w)|$ is polynomial in $|w|$ for all $w \in \Sigma^*$.

Definition 47. Let B be a set defining a decision problem. We say that B is **NP**-*hard*, if for each decision problem A in **NP**, there exists a polynomial reduction from A to B.

Definition 48. If B in **NP** happens to be at the same time **NP**-hard, we say that B is **NP**-*complete*. In other words, the class **NP**-complete is an intersection of classes **NP** and **NP**-hard.

Observation 49. Let f be a reduction from A to B. Let g be a reduction from B to C. Then the composite function $(g \circ f)$ is a reduction from A to C. Furthermore, if both f and g are polynomial, then $(g \circ f)$ is a polynomial reduction from A to C (see Observation 46).

As a result, the relation "existence of a polynomial reduction from A to B" is transitive. Since the identity is a polynomial reduction from A to A, the relation is also reflexive. We say that the relation "existence of a polynomial reduction from A to B" is a quasiorder.

Observation 50. If B is **NP**-hard, C is in **NP**, and there exists a polynomial reduction from B to C, then C is **NP**-complete.

Observation 51. If B is **NP**-hard, C is in **P**, and there exists a polynomial reduction from B to C, then $\mathbf{P} = \mathbf{NP}$.

2.6 VCSP over a fixed language

This section contains a formal definition of the Finite-Valued Constraint Satisfaction Problem, which was briefly mentioned in Section 1.4.

Definition 52. Let D be a finite set, called a *domain*. Elements of D are often referred to as *labels*. We say that $\phi : D^n \to \mathbb{Q}$ is a *cost function* of arity n. We say that Γ is a *finite-valued constraint language* over D if Γ is a set of cost functions (Γ can be a mix of cost functions of various arities). We say that I is an *instance* of VCSP(Γ) if $I = (S, D, \Phi, R)$ where $S = \{x_1, x_2, \ldots, x_s\}$ is a finite set of *variables*, R is a rational number called a *threshold value*, and Φ is an *objective function* expressed as

$$\Phi(x_1, x_2, \ldots, x_s) = \sum_{i=1}^{m} \alpha_i \cdot \phi_i(\mathbf{x}_i)$$

where $\alpha_i \in \mathbb{Q}^+$, $\phi_i \in \Gamma$, and $\mathbf{x}_i \in S^{\mathrm{ar}(\phi_i)}$.

In literature [10], the goal often is to find an *optimal labeling* $\mathbf{x}$ which is an assignment $S \to D$ that minimizes Φ's value. For our purposes, however, decision problems are more convenient. For an explanation of the relationship between optimization and decision problems, see for example [5].

We define that the *labeling* $\mathbf{x} : S \to D$ is a *solution* to I if $\mathbf{x}$ yields $\Phi(\mathbf{x}_{x_1}, \mathbf{x}_{x_2}, \ldots, \mathbf{x}_{x_s}) \leq R$. We say that the instance I is *satisfiable* if there exists any solution $\mathbf{x} : S \to D$ to $I = (S, D, \Phi, R)$.

Any variable $x_j \in V$ can appear in multiple cost functions ϕ_i in Φ. Any cost function $\phi \in \Gamma$ can be used in multiple summands ϕ_i of Φ. Any label $a \in D$ can be assigned to multiple variables $x_j \in S$ in the labeling $\mathbf{x}$.

Remark 53. VCSP stands for *Valued Constraint Satisfaction Problem*. For a fixed language Γ over D, we will speak about VCSP(Γ) as of a decision problem (see Definition 36) or just a *problem* for short. Typically, we will just say that I is an instance of VCSP(Γ), without mentioning "the problem".

2.7 VCSP language for our problem

This section provides an equivalent definition of the Minimum 0-Extension Problem by reformulating it in the VCSP terminology. The formalism that is introduced in the following definition will be used in all remaining chapters.

Definition 54. Given a good graph $G_{\mathrm{w}} = (V, E, \mathrm{w})$, we will construct a finite-valued constraint language $\Gamma_{G_{\mathrm{w}}} = \{\mathrm{d}\} \cup \{\delta_v : v \in V\}$ where $\mathrm{d} = \mathrm{d}_{G_{\mathrm{w}}}$ (see Definition 6) and $\delta_v(w) = \mathrm{d}(v, w)$.

Lemma 55. *Definition 54 is equivalent to Definition 35.*

In exact terms, we claim that the Minimum 0-Extension Problem for G_{w} is reducible to VCSP($\Gamma_{G_{\mathrm{w}}}$) in linear time and that VCSP($\Gamma_{G_{\mathrm{w}}}$) is reducible to the Minimum 0-Extension Problem for G_{w} in linear time. However, we will not explore the details of the input encodings and transducers used in the conversion. Let us keep this lemma short and informal.

Proof. Let $G_{\mathrm{w}} = (V, E, \mathrm{w})$ be a good graph. We will use the same graph G_{w} in both Definition 54 and Definition 35.

First, we will show how to convert an instance (K, c, N) of the Minimum 0-Extension Problem for G_{w}, where $G \leq K = (X, L)$, to an instance I of VCSP($\Gamma_{G_{\mathrm{w}}}$).

We set $I = (S, V, \Phi, R)$ where $S = X \setminus V$. This is the variable set, which consists of "new" vertices. Let us look at the edges. For each edge $\{u, v\} \in L$ where $u \notin V$ and $v \in V$, we add $c(\{u, v\}) \cdot \delta_v(u)$ as a summand of Φ. And for each edge $\{u, v\} \in L$ where $u \notin V$ and $v \notin V$, we add $c(\{u, v\}) \cdot \mathrm{d}(u, v)$ as a summand of Φ.

$$\Phi(S) = \sum_{\substack{\{u,v\}\in L:\\ u\notin V \wedge v\in V}} c(\{u, v\}) \cdot \delta_v(u) \; + \sum_{\substack{\{u,v\}\in L:\\ u\notin V \wedge v\notin V}} c(\{u, v\}) \cdot \mathrm{d}(u, v)$$

The optimal labeling $\mathbf{x}$ yields a value $\Phi(\mathbf{x})$ that is equal to the Minimum 0-Extension of G_{w} minus a constant. The constant is equal to $\sum_{e\in E} c(e)\cdot \mathrm{w}(e)$ which can be set to be zero by a restriction of the cost function c without any loss in the expressive power of the Minimum 0-Extension Problem for G_{w}.

Finally, we set the threshold value $R = N - \sum_{e\in E} c(e) \cdot \mathrm{w}(e)$. As a result, the instance I is satisfiable if and only if the instance (K, c, N) is satisfiable.

For the other implication, let $I = (S, V, \Phi, R)$ be an instance of VCSP($\Gamma_{G_{\mathrm{w}}}$). We will construct an instance (K, c, N) of the Minimum 0-Extension Problem for G_{w} as follows.

$$K = (V \cup S, E \cup B) \;\text{ where }\; B = \{\{u, v\} \mid u \in S \;\wedge\; v \in (V \cup S)\}$$

We construct the cost function c in the following way. Let $u \in S$ and $v \in X$.

$$c(\{u, v\}) = \begin{cases} \alpha, & \text{if } v \in V;\ \text{and } \alpha \cdot \delta_v(u) \text{ is a summand of } \Phi \\ \alpha, & \text{if } v \in S;\ \text{and } \alpha \cdot \mathrm{d}(u, v) \text{ is a summand of } \Phi \\ 0, & \text{otherwise} \end{cases}$$

In order to complete our reduction, we set the threshold value $N = R$.

It is easy to check that, as in the previous case, the new instance (K, c, N) is satisfiable if and only if the instance I is satisfiable.

□

2.8 Selected NP-complete problems

In this section, we provide two (meta)examples of **NP**-complete problems. These problems are highly relevant for the methods used in this work.

Definition 56. Let $G = (V, E)$ be a simple graph and $\{t_1, t_2, \ldots, t_k\} \subseteq V$ be a set of distinct vertices called terminals. A set of edges $Z \subseteq E$ such that, for each $1 \leq i < j \leq k$, every path from t_i to t_j contains at least one edge from Z is called a k-terminal cut [42].

The task of searching for a k-terminal cut Z such that $|Z|$ is minimum for the given input $(G, \{t_1, t_2, \ldots t_k\})$ is called a *Multiterminal Cut Problem*.

Fact 57. If we fix k, then the Multiterminal Cut Problem is in **P** for $k \leq 2$; but for $k \geq 3$ it is **NP**-complete [14].

Remark 58. The Multiterminal Cut Problem is often defined on weighted graphs [42] but the basic version (see Definition 56) is sufficient for Fact 57 to hold [14].

Definition 59. The *Max-Cut Problem* is defined as follows. Given a simple graph $H = (W, F)$ and a number $r \in \mathbb{N}$, we search for a set $X \subseteq W$ such that:

$$|\{\{u, v\} \in F \,:\, |\{u, v\} \cap X| = 1\}| \;\geq\; r$$

We say that (H, r) is an *instance* of the Max-Cut Problem. The set X that satisfied the condition above is called a *solution* of the Max-Cut instance (H, r).

Remark 60. Unlike searching for minimum cuts in graphs, for which efficient algorithms are known [43], searching for maximum cuts in graphs requires a long computation. The decision problem (whether such a set X exists, as used in this text) is known to be **NP**-complete.

NP-completeness was first proved for weighted graphs [44] (which is the Max-Cut Problem variant that we are not interested in) and then for simple graphs [45] (which we use here).

Later, we will reduce the Max-Cut Problem to the Minimum 0-Extension Problem in order to prove that the Minimum 0-Extension Problem is **NP**-hard (and thus it is **NP**-complete).

3. NP-completeness for non-modular graphs

In this chapter, we will prove that the Minimum 0-Extension Problem with a non-modular (see Definition 14) good graph as the template is **NP**-complete.

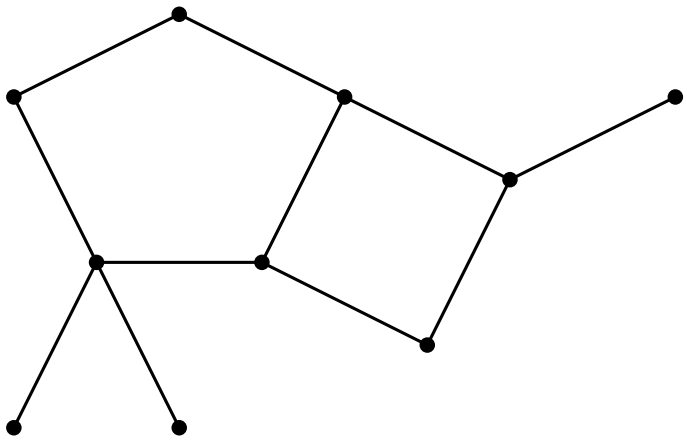


Figure 2: Example of a non-modular orientable graph

3.1 Intervals and medians

Definition 61. Let $G = (V, E)$ be a graph. For two vertices $u, v \in V$, we denote the *interval* between u and v as the set of all vertices of G which lie on any shortest path from u to v, formally:

$$\mathrm{I}(u, v) = \{t \in V : \mathrm{d}(u, t) + \mathrm{d}(t, v) = \mathrm{d}(u, v)\}$$

Observation 62. For any $u, v \in V$, their interval contains them, formally $\mathrm{I}(u, v) \supseteq \{u, v\}$. Moreover, if $G_{\mathrm{w}} = (V, E, \mathrm{w})$ is a good graph and $\{u, v\} \in E$, then $\mathrm{I}(u, v) = \{u, v\}$. The notion of interval between vertices will be central to our work.

Definition 63. Let $G = (V, E)$ be a graph. For any set of three vertices $\{u, v, w\} \subseteq V$, we denote by $\mathrm{Med}(u, v, w)$ the *median set* of triplet $\{u, v, w\}$, that is the set of all vertices of G which lie on all three shortest paths between vertex pairs u—v, v—w, w—u, formally:

$$\mathrm{Med}(u, v, w) = \mathrm{I}(u, v) \cap \mathrm{I}(v, w) \cap \mathrm{I}(w, u)$$

If their intersection is empty, that is $\mathrm{Med}(u, v, w) = \emptyset$, then we say that $\{u, v, w\}$ is a *median-less triplet* in G.

Definition 64. Let $G = (V, E)$ be a graph. We say that G is a *median graph* if every median set is a singleton, that is:

$$\forall\, u_1, u_2, u_3 \in V : \; |\,\mathrm{Med}(u, v, w)| = 1$$

Fact 65. Every median graph is modular and orientable, yet the other implication is not generally true [16]. It should also be said that there exist some median graphs that are not hereditary modular [39].

3.2 Properties of non-modular graphs

Lemma 66. *Let $G = (V, E)$ be a non-modular graph. There must be at least one median-less triplet in G.*

Proof. If we negate the modularity condition

$$\forall u_1, u_2, u_3 \in V\ \exists v \in V\ \forall\, (1 \leq i < j \leq 3) : \mathrm{d}(u_i, v) + \mathrm{d}(v, u_j) = \mathrm{d}(u_i, u_j)$$

we obtain

$$\exists u_1, u_2, u_3 \in V\ \forall v \in V\ \exists\, (1 \leq i < j \leq 3) : \mathrm{d}(u_i, v) + \mathrm{d}(v, u_j) \neq \mathrm{d}(u_i, u_j)$$

which says that no v lies in all three intervals $\mathrm{I}(u_1, u_2)$, $\mathrm{I}(u_2, u_3)$, $\mathrm{I}(u_3, u_1)$. Therefore $\{u_1, u_2, u_3\}$ is a median-less triplet in G. □

Remark 67. The other implication (a modular graph has no median-less triplet) holds as well; however, we will not need it.

Lemma 68. *Let $G_{\mathrm{w}} = (V, E, \mathrm{w})$ be a non-modular good graph. Let $\{u, v, w\}$ be a median-less triplet in G. Then "the triangle inequality is strict":*

$$\mathrm{d}(u, v) + \mathrm{d}(v, w) > \mathrm{d}(w, u)$$

Proof. Since d is a graph metric, we know that $\mathrm{d}(u, v) + \mathrm{d}(v, w) \geq \mathrm{d}(w, u)$. This follows from the properties 22.2 and 22.3 and from Definition 27.

Furthermore, if $\mathrm{d}(u, v) + \mathrm{d}(v, w) = \mathrm{d}(w, u)$, then $v \in \mathrm{Med}(u, v, w)$, which is in contradiction with our triplet $\{u, v, w\}$ being median-less. □

Definition 69. Let $G = (V, E)$ be a non-modular graph. Let $\{u, v, w\}$ be a median-less triplet in G. We denote the sum of their three pairwise distances $\mathrm{d}(u, v) + \mathrm{d}(v, w) + \mathrm{d}(w, u)$ as the *circumference* of the median-less triplet $\{u, v, w\}$.

Definition 70. Let $G = (V, E)$ be a non-modular graph. Let $\{u, v, w\}$ be a median-less triplet in G. We say that $\{u, v, w\}$ is a *minimal* median-less triplet in G if the circumference of $\{u, v, w\}$ is less or equal to the circumference of every median-less triplet in G.

Lemma 71. *Let $G_{\mathrm{w}} = (V, E, \mathrm{w})$ be a non-modular good graph. Let $\{u, v, w\}$ be a minimal median-less triplet in G_{w}. Then any set of three shortest paths between u—v, v—w, w—u form together a simple cycle in G_{w}.*

Proof. None of the three paths alone can contain duplicate vertices, because they must be shortest. In order to observe a simple cycle in G_{w}, we must assure that every two paths meet only at their ends.

For contradiction, let there be a vertex t distinct from v inside both of the shortest paths u—v and v—w. Then $\mathrm{I}(u, t) \subset \mathrm{I}(u, v)$ and $\mathrm{I}(t, w) \subset \mathrm{I}(v, w)$. It follows that the median set of the triplet $\{u, t, w\}$ is empty.

$$\begin{aligned}\mathrm{Med}(u, t, w) &= \mathrm{I}(u, t) \cap \mathrm{I}(t, w) \cap \mathrm{I}(w, u) \subseteq \\ &\subseteq \mathrm{I}(u, v) \cap \mathrm{I}(v, w) \cap \mathrm{I}(w, u) = \mathrm{Med}(u, v, w) = \emptyset\end{aligned}$$

This inclusion says that $\mathrm{Med}(u, t, w) = \emptyset$. As a result, the triplet $\{u, t, w\}$ is median-less, but its circumference $\mathrm{d}(u, t) + \mathrm{d}(t, w) + \mathrm{d}(w, u)$ is less than the circumference of the median-less triplet $\{u, v, w\}$, thus our median-less triplet $\{u, v, w\}$ cannot be minimal, a contradiction. □

3.3 Construction of the reduction

3.3.1 A high-level description

We are given a non-modular good graph $G_{\mathrm{w}} = (V, E, \mathrm{w})$ where $\{a, b, c\} \subseteq V$ is a minimal median-less triplet in G_{w}. Without loss of generality, we assume that $\mathrm{d}(a, b) \leq \mathrm{d}(b, c) \leq \mathrm{d}(a, c)$. The graph G_{w} will be fixed for the whole construction, independent of the input instance.

For any instance of the Max-Cut Problem (H, r), where $H = (W, F)$ is a simple graph, we create an instance I of $\mathrm{VCSP}(\Gamma_{G_{\mathrm{w}}})$ of size polynomial in $|W|$ such that the instance I will have a solution if and only if the Max-Cut instance (H, r) has a solution.

We will construct the instance as follows. In the first step, given a pair of vertices $p, q \in \{a, b, c\}$, where $p \neq q$, we will force a variable $w \in S$ to get a label from $\mathrm{I}(p, q)$ using a function $f_{p,q}$. In the second step, we will force the variable w to get a label from $\{p, q\}$ using a function $g_{p,q}$. In the final step, we will construct an objective function m such that its minimum value will be a decreasing function of the size of the maximum cut in H.

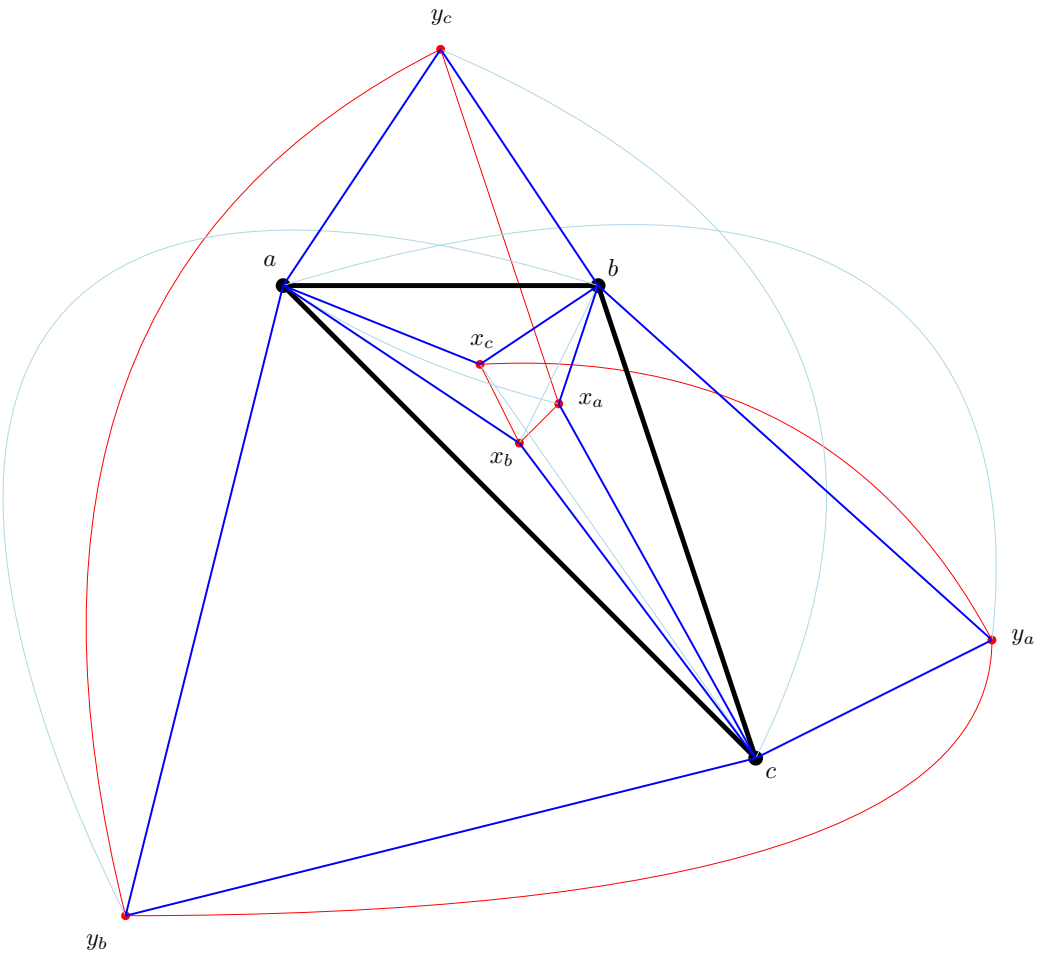


Figure 3: A gadget for an edge $\{x, y\}$. The strong black edges represent the median-less triplet $\{a, b, c\}$. The strong blue edges are added by the function f. The faint blue edges are added by the function g. The red edges are added by the function h'.

3.3.2 The construction

For brevity, we introduce notation $\overline{pq} \coloneqq l$ such that $l \in \{a, b, c\}, l \neq p, l \neq q$. This "bar" notation was suggested by professor Kolmogorov [17] for better readability. When written explicitly for $p, q \in \{a, b, c\}, p \neq q$, it says:

$$\overline{ab} = c \quad , \quad \overline{bc} = a \quad , \quad \overline{ca} = b$$

We start with an easy-to-imagine function f.

$$f_{p,q}(x) = \delta_p(x) + \delta_q(x)$$

The construction of the following function, which has the desired properties, was suggested to us by professor Vladimir Kolmogorov [17]. Let M be a sufficiently large constant that depends only on G_{w}.

$$g_{p,q}(x) = M \cdot f_{p,q}(x) + \frac{\mathrm{d}(p,q)}{\mathrm{d}(p,\overline{pq}) - \mathrm{d}(q,\overline{pq})} \cdot \delta_{\overline{pq}}(x) + \delta_p(x)$$

We assumed that p is incident with the longer path to the remaining vertex, i.e. $\mathrm{d}(p,\overline{pq}) > \mathrm{d}(q,\overline{pq})$. If the inequality happens to be the opposite, we must swap p and q. In the case of an equality between them, i.e. $\mathrm{d}(p,\overline{pq}) = \mathrm{d}(q,\overline{pq})$, we simply define function g as follows.

$$g_{p,q}(x) = M \cdot f_{p,q}(x) + \delta_{\overline{pq}}(x)$$

We will use the same triplet of median-less vertices in order to add six calls of the function g for each edge $e \in F$ of the Max-Cut graph.

$$h(y_a, y_b, y_c, z_a, z_b, z_c) = g_{b,c}(y_a) + g_{b,c}(z_a) + g_{a,c}(y_b) + g_{a,c}(z_b) + g_{a,b}(y_c) + g_{a,b}(z_c)$$

Now we create essential connections between variables.

$$\begin{aligned} h'(y_a, y_b, y_c, z_a, z_b, z_c) &= M \cdot h(y_a, y_b, y_c, z_a, z_b, z_c) + \\ &+ \mathrm{d}(y_a, y_b) + \mathrm{d}(y_b, y_c) + \mathrm{d}(y_c, z_a) + \mathrm{d}(z_a, z_b) + \mathrm{d}(z_b, z_c) + \mathrm{d}(z_c, y_a) \end{aligned}$$

Then we must introduce a penalty for avoiding the most distant vertex.

$$\begin{aligned} \hat{h}(y_a, y_b, y_c, z_a, z_b, z_c) &= h'(y_a, y_b, y_c, z_a, z_b, z_c) + \\ &+ \frac{\mathrm{d}(b,c) - \mathrm{d}(a,b)}{\mathrm{d}(b,c)} \cdot (\delta_c(y_a) + \delta_c(z_a)) + \frac{\mathrm{d}(b,c) - \mathrm{d}(a,b)}{\mathrm{d}(a,c)} \cdot (\delta_c(y_b) + \delta_c(z_b)) \end{aligned}$$

In the end, we get the objective function by summing $\hat{h}$ over all edges of the Max-Cut graph H. Our instance is built as follows.

$$S = \{w_a, w_b, w_c : w \in W\}$$

The set S contains three variables for each vertex of the graph H. The objective function m contains a summand $\hat{h}$ (applied to six variables) for each edge of the graph H.

$$m(\mathbf{x}) = \sum_{\{u,v\} \in F} \hat{h}(\mathbf{x}_{u_a}, \mathbf{x}_{u_b}, \mathbf{x}_{u_c}, \mathbf{x}_{v_a}, \mathbf{x}_{v_b}, \mathbf{x}_{v_c})$$

The goal is to minimize m. More precisely, we search for a labeling $\mathbf{x} : S \to V$ such that $m(\mathbf{x}) \leq K$ for some K based on H. The target value for K will result from the analysis (Corollary 79 in particular).

3.4 Analysis of the reduction

3.4.1 Analysis of the functions

Lemma 72. *Let f be the function $V \to \mathbb{Q}$ as defined in Section 3.3.2.*

$$\arg\min(f_{p,q}) = \mathrm{I}(p,q)$$

Proof. Since d is a graph metric, the triangle inequality tells us that

$$f_{p,q}(p) = \mathrm{d}(p,q) = f_{p,q}(q)$$

is the (typically not strict) minimum of f. The definition of interval

$$\mathrm{I}(p,q) = \{t \in V : \mathrm{d}(p,t) + \mathrm{d}(t,q) = \mathrm{d}(p,q)\}$$

is equivalent to

$$\{v \in V : \delta_p(v) + \delta_q(v) = \mathrm{d}(p,q)\}$$

and this set is equal to $\arg\min(f_{p,q})$ because $\min(f_{p,q}) = \mathrm{d}(p,q)$. □

Lemma 73. *Let g be the function $V \to \mathbb{Q}$ as defined in Section 3.3.2.*

$$\arg\min(g_{p,q}) = \{p,q\}$$

Proof. If $\mathrm{d}(p,\overline{pq}) = \mathrm{d}(q,\overline{pq})$, the situation is simple. We see that:

$$g_{p,q}(p) = M \cdot \mathrm{d}(p,q) + \mathrm{d}(p,\overline{pq}) = M \cdot \mathrm{d}(p,q) + \mathrm{d}(q,\overline{pq}) = g_{p,q}(q)$$

Claim 1. $\mathrm{d}(x,\overline{pq}) > \mathrm{d}(p,\overline{pq})$

Let us first have a look at what Claim 1 implies.

For vertices other than p and q, we see that either $x \notin \mathrm{I}(p,q)$ makes the first summand very high or we get:

$$\forall x \in \mathrm{I}(p,q) - \{p,q\} : g_{p,q}(x) = M \cdot \mathrm{d}(p,q) + \mathrm{d}(x,\overline{pq}) > \\ > M \cdot \mathrm{d}(p,q) + \mathrm{d}(p,\overline{pq}) = g_{p,q}(p)$$

As a result, the case $\mathrm{d}(p,\overline{pq}) = \mathrm{d}(q,\overline{pq})$ is solved.

Proof of Claim 1. For contradiction, let us suppose that $\exists x \in \mathrm{I}(p,q) - \{p,q\}$ such that $\mathrm{d}(x,\overline{pq}) \leq \mathrm{d}(p,\overline{pq})$. Without loss of generality, we can assume that $\mathrm{d}(\overline{pq},p) \leq \mathrm{d}(\overline{pq},q)$. Because $\{a,b,c\}$ is minimal among all median-less triplets, the smaller triplet $\{p,\overline{pq},x\}$ must have a median u. Since $u \in \mathrm{I}(p,x)$ and $x \in \mathrm{I}(p,q)$, also $u \in \mathrm{I}(p,q)$.

As a result, the triplet $\{\overline{pq},q,u\}$ must have a median as well, we will denote the median by v. By definition, this median v belongs to $\mathrm{I}(\overline{pq},q)$. Furthermore, since $v \in \mathrm{I}(q,u)$ and $u \in \mathrm{I}(p,q)$, then also $v \in \mathrm{I}(p,q)$. At the same time, since $v \in \mathrm{I}(\overline{pq},u)$ and $u \in \mathrm{I}(\overline{pq},q)$, then also $v \in \mathrm{I}(\overline{pq},q)$. Putting it all together, $v \in \mathrm{Med}(p,q,\overline{pq})$, which is a contradiction with $\{a,b,c\}$ being median-less. ■

Let us move to the (more complicated) case of $\mathrm{d}(p, \overline{pq}) > \mathrm{d}(q, \overline{pq})$. It would help us if we knew that any three shortest paths $p — q$, $q — \overline{pq}$, $\overline{pq} — p$ form together an isometric cycle in G. However, the following weaker property will be sufficient.

Claim 2.

$$\forall x \in \mathrm{I}(p, q) : \{p, q\} \cap \mathrm{I}(\overline{pq}, x) \neq \emptyset$$

Proof of Claim 2. Let us suppose for contradiction the negation of the claim.

$$\exists x \in \mathrm{I}(p, q) : p, q \notin \mathrm{I}(\overline{pq}, x)$$

In other words, $\mathrm{d}(\overline{pq}, x) < \mathrm{d}(\overline{pq}, i) + \mathrm{d}(i, x)$ for both $i = p$ and $i = q$. However, this makes both $\{\overline{pq}, q, x\}$ and $\{\overline{pq}, p, x\}$ be median-less. That is obviously in contradiction with $\{p, q, \overline{pq}\}$ being a minimal median-less triplet. ■

In order to check which labels fall into $\arg\min(g_{p,q})$, we will consider five possible cases of $g_{p,q}(x)$ according to the location of x with respect to p, q.
I) $x = p$
II) $x = q$
III) $x \notin \mathrm{I}(p, q)$
IV) $x \in \mathrm{I}(p, q) \setminus \{p, q\}$, $p \in \mathrm{I}(\overline{pq}, x)$
V) $x \in \mathrm{I}(p, q) \setminus \{p, q\}$, $q \in \mathrm{I}(\overline{pq}, x)$

The previous property implies that our decomposition into the five cases is comprehensive. For short, let us denote:

$$\alpha \coloneqq \frac{\mathrm{d}(p, q)}{\mathrm{d}(p, \overline{pq}) - \mathrm{d}(q, \overline{pq})}$$

We immediately see that $\alpha > 1$, because $\{\overline{pq}, q, p\}$ is median-less. Using this notation, we will write:

$$g_{p,q}(x) = M \cdot f_{p,q}(x) + \alpha \cdot \delta_{\overline{pq}}(x) + \delta_p(x)$$

I)

$$g_{p,q}(x) = g_{p,q}(p) = M \cdot f_{p,q}(p) + \alpha \cdot \mathrm{d}(\overline{pq}, p) + \mathrm{d}(p, p) =$$
$$= M \cdot \min(f_{p,q}) + \alpha \cdot \mathrm{d}(\overline{pq}, p)$$

II)

$$g_{p,q}(x) = g_{p,q}(q) = M \cdot f_{p,q}(q) + \alpha \cdot \mathrm{d}(\overline{pq}, q) + \mathrm{d}(p, q) =$$
$$= M \cdot \min(f_{p,q}) + \alpha \cdot \mathrm{d}(\overline{pq}, q) + \mathrm{d}(p, q)$$

Expanding the definition of α we get:

$$\alpha \cdot \mathrm{d}(\overline{pq}, q) + \mathrm{d}(p, q) =$$

$$= \frac{\mathrm{d}(p, q) \cdot \mathrm{d}(\overline{pq}, q) + (\mathrm{d}(p, \overline{pq}) - \mathrm{d}(q, \overline{pq})) \cdot \mathrm{d}(p, q)}{\mathrm{d}(p, \overline{pq}) - \mathrm{d}(q, \overline{pq})} =$$

$$= \frac{\mathrm{d}(p, \overline{pq}) \cdot \mathrm{d}(p, q)}{\mathrm{d}(p, \overline{pq}) - \mathrm{d}(q, \overline{pq})} = \alpha \cdot \mathrm{d}(p, \overline{pq})$$

We got $g_{p,q}(q) = g_{p,q}(p)$. It remains to show that all other values of $g_{p,q}$ are higher.

III)

$$g_{p,q}(x) = M \cdot f_{p,q}(x) + \alpha \cdot \mathrm{d}(\overline{pq}, x) + \mathrm{d}(p, x) > M \cdot (\min(f_{p,q}) + \varepsilon) > g_{p,q}(p)$$

We put $\varepsilon = \mathrm{d}(p, x) + \mathrm{d}(q, x) - \mathrm{d}(p, q) > 0$ and M be sufficiently high.

IV)
We will first use (in the first equality) the definition of g, then we will use (in the second equality) Lemma 72, then we will use (in the third equality) that $p \in \mathrm{I}(\overline{pq}, x)$, and finally we will use (in the inequality) that $x \neq p$.

$$\begin{aligned} g_{p,q}(x) &= f_{p,q}(x) + \alpha \cdot \mathrm{d}(\overline{pq}, x) + \mathrm{d}(p, x) = \\ &= \min(f_{p,q}) + \alpha \cdot \mathrm{d}(\overline{pq}, x) + \mathrm{d}(p, x) = \\ &= \min(f_{p,q}) + \alpha \cdot (\mathrm{d}(\overline{pq}, p) + \mathrm{d}(p, x)) + \mathrm{d}(p, x) > \\ &> \min(f_{p,q}) + \alpha \cdot \mathrm{d}(\overline{pq}, p) = g_{p,q}(p) \end{aligned}$$

V)
We will first use (in the first equality) the definition of g, then we will use (in the second equality) Lemma 72, then we will use (in the third equality) that $q \in \mathrm{I}(\overline{pq}, x)$, then we will use (in the following inequality) that $\alpha > 1$, and finally (in the penultimate equality) we will use that $x \in \mathrm{I}(p, q)$.

$$\begin{aligned} g_{p,q}(x) &= f_{p,q}(x) + \alpha \cdot \mathrm{d}(\overline{pq}, x) + \mathrm{d}(p, x) = \\ &= \min(f_{p,q}) + \alpha \cdot \mathrm{d}(\overline{pq}, x) + \mathrm{d}(p, x) = \\ &= \min(f_{p,q}) + \alpha \cdot (\mathrm{d}(\overline{pq}, q) + \mathrm{d}(q, x)) + \mathrm{d}(p, x) > \\ &> \min(f_{p,q}) + \alpha \cdot \mathrm{d}(\overline{pq}, q) + \mathrm{d}(q, x) + \mathrm{d}(p, x) = \\ &= \min(f_{p,q}) + \alpha \cdot \mathrm{d}(\overline{pq}, q) + \mathrm{d}(q, p) = g_{p,q}(q) \end{aligned}$$

□

Corollary 74. *If $(y_a, y_b, y_c, z_a, z_b, z_c)$ minimizes h, then:*

$$y_a, z_a \in \{b, c\} \quad , \quad y_b, z_b \in \{a, c\} \quad , \quad y_c, z_c \in \{a, b\}$$

Proof. The function h is a sum of six calls to the function g. Each call gets a different (unique) variable. Thus, we can minimize each part independently of the other parts.

$$\begin{aligned} \min(h(y_a, y_b, y_c, z_a, z_b, z_c)) = &\min(g_{b,c}(y_a)) + \min(g_{b,c}(z_a)) + \min(g_{a,c}(y_b)) + \\ &+ \min(g_{a,c}(z_b)) + \min(g_{a,b}(y_c)) + \min(g_{a,b}(z_c)) \end{aligned}$$

It remains to apply Lemma 73 to each summand. □

Lemma 75. *There is a number $t \in \mathbb{R}^+$ and a tuple $(y_a, y_b, y_c, z_a, z_b, z_c) \in V^6$ where $y_a \neq z_a$, which minimizes $\hat{h}$ in such a way that:*

- $\hat{h}(y_a, y_b, y_c, z_a, z_b, z_c) = \hat{h}(z_a, z_b, z_c, y_a, y_b, y_c) = T$
- $\hat{h}(r_a, r_b, r_c, s_a, s_b, s_c) \geq T + t$
  for all the other tuples $(r_a, r_b, r_c, s_a, s_b, s_c) \in V^6$

Proof. The function $\hat{h}(y_a, y_b, y_c, z_a, z_b, z_c)$ contains $h(y_a, y_b, y_c, z_a, z_b, z_c)$ as a summand with a very high coefficient. This forces us to narrow down the possible assignments to exactly those 64 assignments which satisfy the conditions given by Corollary 74.

We claim that the minimum value of $\hat{h}(y_a, y_b, y_c, z_a, z_b, z_c)$ is attained by the following assignment:

$$y_a = y_b = c \quad , \quad y_c = z_a = b \quad , \quad z_b = z_c = a$$

We also claim that the only other assignment that gives the same value is the symmetric one:

$$z_a = z_b = c \quad , \quad z_c = y_a = b \quad , \quad y_b = y_c = a$$

By plugging them into the definition of $\hat{h}$ we get:

$$\begin{aligned}
T = \hat{h}(c, c, b, b, a, a) &= h'(c, c, b, b, a, a) + \\
&+ \frac{\mathrm{d}(b, c) - \mathrm{d}(a, b)}{\mathrm{d}(b, c)} \cdot (\delta_c(c) + \delta_c(b)) + \frac{\mathrm{d}(b, c) - \mathrm{d}(a, b)}{\mathrm{d}(a, c)} \cdot (\delta_c(c) + \delta_c(a)) = \\
&= M \cdot h(c, c, b, b, a, a) + \mathrm{d}(c, b) + \mathrm{d}(b, a) + \mathrm{d}(a, c) + \\
&+ \frac{\mathrm{d}(b, c) - \mathrm{d}(a, b)}{\mathrm{d}(b, c)} \cdot \delta_c(b) + \frac{\mathrm{d}(b, c) - \mathrm{d}(a, b)}{\mathrm{d}(a, c)} \cdot \delta_c(a) = \\
&= M \cdot \min(h) + \mathrm{d}(c, b) + \mathrm{d}(b, a) + \mathrm{d}(a, c) + 2 \cdot (\mathrm{d}(b, c) - \mathrm{d}(a, b)) = \\
&= M \cdot \min(h) + 3 \cdot \mathrm{d}(b, c) + \mathrm{d}(a, c) - \mathrm{d}(a, b)
\end{aligned}$$

In the symmetric case, the calculation goes exactly the same way.

$$T = \hat{h}(b, a, a, c, c, b) = M \cdot \min(h) + 3 \cdot \mathrm{d}(b, c) + \mathrm{d}(a, c) - \mathrm{d}(a, b)$$

In order to prove the uniqueness, we will classify assignments according to number of changes along the cycle $(y_a, y_b, y_c, z_a, z_b, z_c)$ based on the binary functions inside the definition of h'. By the number of changes we mean:

$$1_{[y_a \neq y_b]} + 1_{[y_b \neq y_c]} + 1_{[y_c \neq z_a]} + 1_{[z_a \neq z_b]} + 1_{[z_b \neq z_c]} + 1_{[z_c \neq y_a]}$$

We will argue that 3 changes are optimal. Note that it is equal to the number of changes in the assignments above.

Whenever a pair of consecutive variables on the cycle $(y_a, y_b, y_c, z_a, z_b, z_c)$ has different labels, at least $\mathrm{d}(b, c)$ is paid extra there, as we will show now. Let us focus on the following value:

$$\Delta = \hat{h}(y_a, y_b, y_c, z_a, z_b, z_c) - h(y_a, y_b, y_c, z_a, z_b, z_c)$$

We will calculate a lower bound for Δ by enumerating all (three) possible cases, ignoring whether p follows after q or q follows after p.

- A change between b and c yields $\Delta \geq \mathrm{d}(b, c)$.
- A change between a and c yields $\Delta \geq \mathrm{d}(a, c) \geq \mathrm{d}(b, c)$.
- A change between a and b is more complicated. Note that both values cannot be attained in variables y_c and z_c, because we speak about two consecutive variables in the cycle, where y_c is in the antipodal position to z_c.
  - If $y_a = b$ or $z_a = b$, then we get:

$$\Delta \geq \mathrm{d}(a, b) + \frac{\mathrm{d}(b, c) - \mathrm{d}(a, b)}{\mathrm{d}(b, c)} \cdot \mathrm{d}(c, b) = \mathrm{d}(b, c)$$

  - If $y_b = a$ or $z_b = a$, then we get:

$$\Delta \geq \mathrm{d}(a, b) + \frac{\mathrm{d}(b, c) - \mathrm{d}(a, b)}{\mathrm{d}(a, c)} \cdot \mathrm{d}(c, a) = \mathrm{d}(b, c)$$

What we got by bounding $\Delta \geq \mathrm{d}(b, c)$ is a lower bound for the price of a change. Knowing this, we can establish the optimality of exactly 3 changes. If we have more than 3 changes on our cycle, then:

$$\hat{h}(y_a, y_b, y_c, z_a, z_b, z_c) \geq M \cdot \min(h) + 4 \cdot \mathrm{d}(b, c) = T + t$$

Now we must check the difference between this and previous values of $\hat{h}$.

$$t = \mathrm{d}(a, b) + \mathrm{d}(b, c) - \mathrm{d}(a, c)$$

Since $\{a, b, c\}$ is median-less, Lemma 68 gives us that $t > 0$.

On the other hand, we can assure that less than 3 changes of a label on the cycle are incompatible with minimizing $\hat{h}$ as well, because no consecutive triplet on the cycle can share one label, and at the same time be compatible with Corollary 74.

As for labelings with exactly 3 changes, there are only two possibilities where the changes can occur, because there are only two perfect matchings of C_6. Both of them are shown above. □

Lemma 76. *There is a number $t \in \mathbb{R}^+$ and a tuple $(y_a, y_b, y_c, z_a, z_b, z_c) \in V^6$ where $y_a \neq z_a$, which minimizes $\hat{h}$ in such a way that:*

- $\hat{h}(y_a, y_b, y_c, z_a, z_b, z_c) = \hat{h}(z_a, z_b, z_c, y_a, y_b, y_c) = T$
- $\hat{h}(y_a, y_b, y_c, y_a, y_b, y_c) = \hat{h}(z_a, z_b, z_c, z_a, z_b, z_c) = T + t$
- $\hat{h}(r_a, r_b, r_c, s_a, s_b, s_c) \geq T + t$
  for all the other tuples $(r_a, r_b, r_c, s_a, s_b, s_c) \in V^6$

These properties are similar to those used in the proof of **NP**-completeness of the Minimum 3-Terminal Cut Problem [14].

Proof. We will use the same values as in Lemma 75.

$$y_a = y_b = c \quad , \quad y_c = z_a = b \quad , \quad z_b = z_c = a$$

The first calculation in the second item $\hat{h}(y_a, y_b, y_c, y_a, y_b, y_c)$ is as follows.

$$\begin{aligned}\hat{h}(c,c,b,c,c,b) &= h'(c,c,b,c,c,b) + \\ &+ \frac{\mathrm{d}(b,c) - \mathrm{d}(a,b)}{\mathrm{d}(b,c)} \cdot (\delta_c(c) + \delta_c(c)) + \frac{\mathrm{d}(b,c) - \mathrm{d}(a,b)}{\mathrm{d}(a,c)} \cdot (\delta_c(c) + \delta_c(c)) = \\ &= h'(c,c,b,c,c,b) = M \cdot \min(h) + 4 \cdot \mathrm{d}(b,c) = T + t\end{aligned}$$

The second calculation in the second item $\hat{h}(z_a, z_b, z_c, z_a, z_b, z_c)$ is as follows.

$$\begin{aligned}\hat{h}(b,a,a,b,a,a) &= h'(b,a,a,b,a,a) + \\ &+ \frac{\mathrm{d}(b,c) - \mathrm{d}(a,b)}{\mathrm{d}(b,c)} \cdot (\delta_c(b) + \delta_c(b)) + \frac{\mathrm{d}(b,c) - \mathrm{d}(a,b)}{\mathrm{d}(a,c)} \cdot (\delta_c(a) + \delta_c(a)) = \\ &= h'(b,a,a,b,a,a) + 2 \cdot (\mathrm{d}(b,c) - \mathrm{d}(a,b)) + 2 \cdot (\mathrm{d}(b,c) - \mathrm{d}(a,b)) = \\ &= M \cdot \min(h) + 4 \cdot \mathrm{d}(a,b) + 4 \cdot (\mathrm{d}(b,c) - \mathrm{d}(a,b)) = \\ &= M \cdot \min(h) + 4 \cdot \mathrm{d}(b,c) = T + t\end{aligned}$$

The rest of the proof follows directly from Lemma 75. □

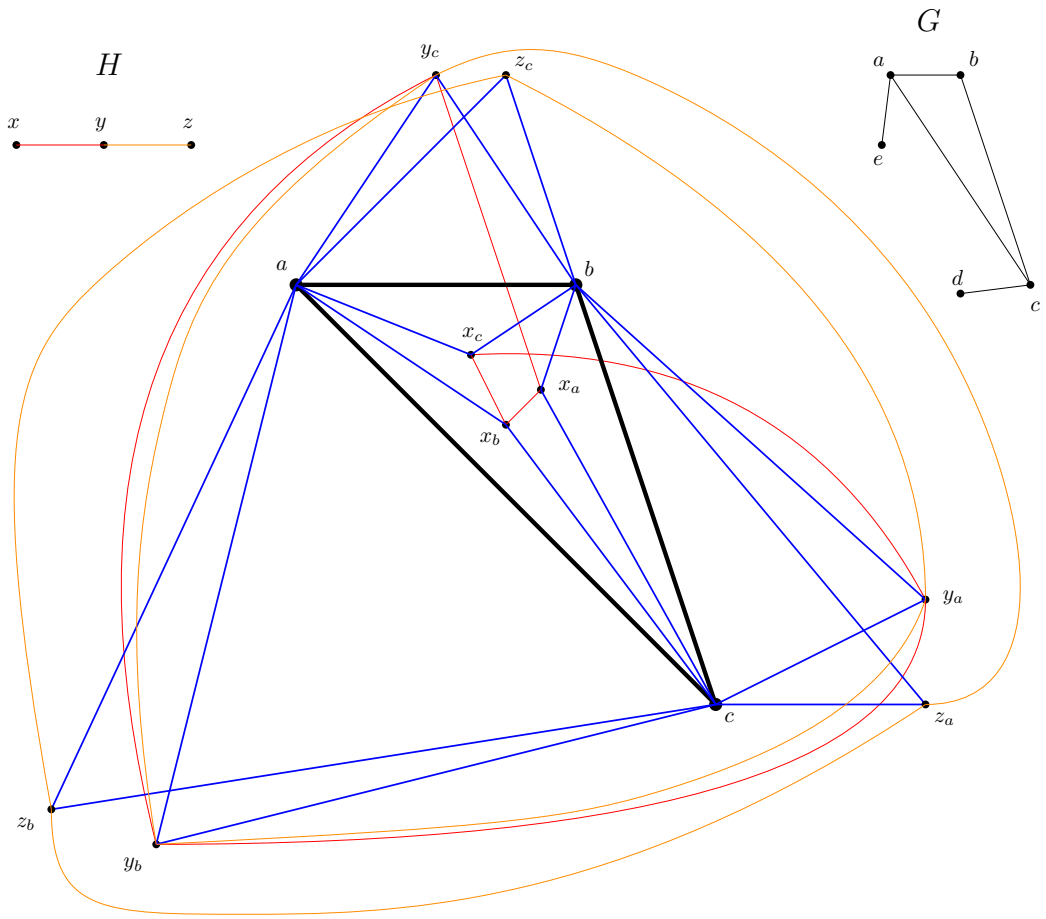


Figure 4: A small example of the reduction. The Max-Cut Problem for the graph $H = (\{x, y, z\}, \{\{x, y\}, \{y, z\}\})$ is reduced to the Minimum 0-Extension Problem for the graph $G = (\{a, b, c, d, e\}, \{\{a, b\}, \{a, c\}, \{a, e\}, \{b, c\}, \{c, d\}\})$. The faint blue edges (from the previous figure) are omitted here.

3.4.2 Analysis of the instances

Lemma 77. *There is a tuple* $(y_a, y_b, y_c, z_a, z_b, z_c) \in V^6$ *where* $y_a \neq z_a$ *such that, if a labeling* $\mathbf{x}$ *minimizes* $m(\mathbf{x})$*, then:*

$$\forall w \in W : \Big((\exists v \in W : \{v, w\} \in F) \implies \\ \implies \big((\mathbf{x}_{w_a}, \mathbf{x}_{w_b}, \mathbf{x}_{w_c}) = (y_a, y_b, y_c) \ \vee \ (\mathbf{x}_{w_a}, \mathbf{x}_{w_b}, \mathbf{x}_{w_c}) = (z_a, z_b, z_c) \big) \Big)$$

Proof. The tuple $(y_a, y_b, y_c, z_a, z_b, z_c)$ is provided by Lemma 76. Recall the objective function:

$$m(\mathbf{x}) = \sum_{\{u,v\} \in F} \hat{h}(\mathbf{x}_{u_a}, \mathbf{x}_{u_b}, \mathbf{x}_{u_c}, \mathbf{x}_{v_a}, \mathbf{x}_{v_b}, \mathbf{x}_{v_c})$$

For contradiction, let $m(\mathbf{x})$ be minimum and $w \in W$ be a vertex that does not satisfy the condition. Then we take any neighbor $v \in W : \{v, w\} \in F$.

We declare a set $U \subseteq W$ of "ugly vertices", i.e. vertices whose corresponding variables are labeled by neither (y_a, y_b, y_c) nor (z_a, z_b, z_c).

$$U = \{u \in W : (\mathbf{x}_{u_a}, \mathbf{x}_{u_b}, \mathbf{x}_{u_c}) \notin \{(y_a, y_b, y_c), (z_a, z_b, z_c)\}\}$$

If $(\mathbf{x}_{v_a}, \mathbf{x}_{v_b}, \mathbf{x}_{v_c}) = (y_a, y_b, y_c)$, then we create a new labeling $\mathbf{x}^*$ as follows.

$$\begin{array}{rrcl} & (\mathbf{x}^*_{w_a}, \mathbf{x}^*_{w_b}, \mathbf{x}^*_{w_c}) & := & (z_a, z_b, z_c) \\ \forall u \in U \setminus \{w\} : & (\mathbf{x}^*_{u_a}, \mathbf{x}^*_{u_b}, \mathbf{x}^*_{u_c}) & := & (y_a, y_b, y_c) \\ \forall x \in W \setminus U : & (\mathbf{x}^*_{x_a}, \mathbf{x}^*_{x_b}, \mathbf{x}^*_{x_c}) & := & (\mathbf{x}_{x_a}, \mathbf{x}_{x_b}, \mathbf{x}_{x_c}) \end{array}$$

If $(\mathbf{x}_{v_a}, \mathbf{x}_{v_b}, \mathbf{x}_{v_c}) \neq (y_a, y_b, y_c)$, then we create a new labeling $\mathbf{x}^*$ as follows.

$$\begin{array}{rrcl} & (\mathbf{x}^*_{w_a}, \mathbf{x}^*_{w_b}, \mathbf{x}^*_{w_c}) & := & (y_a, y_b, y_c) \\ \forall u \in U \setminus \{w\} : & (\mathbf{x}^*_{u_a}, \mathbf{x}^*_{u_b}, \mathbf{x}^*_{u_c}) & := & (z_a, z_b, z_c) \\ \forall x \in W \setminus U : & (\mathbf{x}^*_{x_a}, \mathbf{x}^*_{x_b}, \mathbf{x}^*_{x_c}) & := & (\mathbf{x}_{x_a}, \mathbf{x}_{x_b}, \mathbf{x}_{x_c}) \end{array}$$

Thanks to Lemma 76, all summands of $m(\mathbf{x}^*)$ are less than or equal to $T + t$. At the same time, whenever any summand of $m(\mathbf{x})$ was equal to T, which could happen only when the corresponding edge had both ends in $(W \setminus U)$, the corresponding summand in $m(\mathbf{x}^*)$ is equal to T as well, because vertices from $(W \setminus U)$ get identical labels in $\mathbf{x}^*$ as in $\mathbf{x}$. Finally, the value of $\hat{h}$ for the edge $\{v, w\}$ in $m(\mathbf{x}^*)$ is less than the corresponding value in $m(\mathbf{x})$ by at least t.

In either case regarding $(\mathbf{x}_{v_a}, \mathbf{x}_{v_b}, \mathbf{x}_{v_c})$, after summing everything together, we obtain $m(\mathbf{x}^*) \leq m(\mathbf{x}) - t < m(\mathbf{x})$. This is a contradiction with $m(\mathbf{x})$ being minimum. □

Theorem 78. *If a labeling* $\mathbf{x}$ *minimizes* $m(\mathbf{x})$*, then*

$$|\{\{u, v\} \in F : \mathbf{x}_{u_a} \neq \mathbf{x}_{v_a}\}|$$

is equal to the size of maximum cut in H*.*

Proof. Let us consider a potentially-minimum labeling $\mathbf{x}$ (that is, $\mathbf{x}$ satisfies the necessary condition stated in Lemma 77). With the help of Lemma 76, we can observe:

$$m(\mathbf{x}) = |F| \cdot T + |\{\{u,v\} \in F : \mathbf{x}_{u_a} = \mathbf{x}_{v_a}\}| \cdot t$$

We choose an arbitrary vertex $s \in W$ and, for the given labeling $\mathbf{x}$, we define a set $X(\mathbf{x}) = \{w \in W : \mathbf{x}_{w_a} = \mathbf{x}_{s_a}\}$. The condition for minimality of $m(\mathbf{x})$ implies that $\forall\{u,v\} \in F, u \notin X, v \notin X : \mathbf{x}_{u_a} = \mathbf{x}_{v_a}$. It immediately follows that:

$$\{\{u,v\} \in F : |\{u,v\} \cap X| = 1\} \;=\; \{\{u,v\} \in F : \mathbf{x}_{u_a} \neq \mathbf{x}_{v_a}\}$$

It is trivial that $|\{\{u,v\} \in F : \mathbf{x}_{u_a} = \mathbf{x}_{v_a}\}|$ is minimum over all labelings $\mathbf{x}$ if and only if $|\{\{u,v\} \in F : \mathbf{x}_{u_a} \neq \mathbf{x}_{v_a}\}|$ is maximum. As a result, if $m(\mathbf{x})$ is minimum, then X determines the maximum cut of H. □

3.5 Hardness result

Corollary 79. *If $G_{\mathrm{w}} = (V, E, \mathrm{w})$ is a non-modular good graph, then the VCSP($\Gamma_{G_{\mathrm{w}}}$) is* **NP***-hard.*

Proof. Let $H = (W, F)$ be a simple graph and $r \in \mathbb{N}$. For an instance (H, r) of the Max-Cut Problem, we create the following instance of VCSP($\Gamma_{G_{\mathrm{w}}}$).

$$S = \{w_a, w_b, w_c : w \in W\}$$

$$m(\mathbf{x}) = \sum_{\{u,v\} \in F} \hat{h}(\mathbf{x}_{u_a}, \mathbf{x}_{u_b}, \mathbf{x}_{u_c}, \mathbf{x}_{v_a}, \mathbf{x}_{v_b}, \mathbf{x}_{v_c})$$

Goal: say YES iff there is any labeling $\mathbf{x} : S \to V$ such that

$$m(\mathbf{x}) \leq |F| \cdot T + (|F| - r) \cdot t$$

where T, t and $\hat{h}$ is given by Lemma 75. If $\min(m(\mathbf{x})) \leq |F| \cdot T + (|F| - r) \cdot t$, then the size of the maximum cut of H is greater or equal to r, as follows from our Theorem 78. If $\min(m(\mathbf{x})) > |F| \cdot T + (|F| - r) \cdot t$, then the size of the maximum cut of H is less than r, for the same reason. Therefore, our reduction is correct. And, as we already mentioned (see Fact 60), the Max-Cut Problem is **NP**-hard.

By checking all steps of the construction (Section 3.3.2), we can easily see that the reduction is polynomial, because all steps of the construction are linear in $|W|$ and $|F|$. As a result, VCSP($\Gamma_{G_{\mathrm{w}}}$) is **NP**-hard. □

Corollary 80. *The Minimum 0-Extension Problem for any non-modular good graph is* **NP***-complete.*

Proof. Every solution to any (V)CSP has a linear size and it can be verified in polynomial time. The existence of a polynomial deterministic verification algorithm is obvious. As a result, the problem is in **NP**. In conjunction with Corollary 79 and Lemma 55, we can conclude that the Minimum 0-Extension Problem for a non-modular good graph is **NP**-complete. □

4. NP-completeness for modular non-orientable graphs

Let us move to the case of non-orientable (see Definition 19) good graphs. Unlike the previous (less general) Karzanov's proof [1], we will assume modularity in the proof for non-orientable graphs, since the case of non-modular graphs has already been solved.

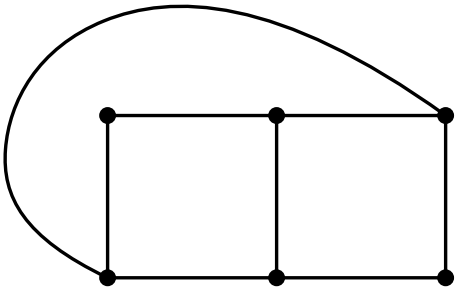


Figure 5: Example of a modular non-orientable graph ($K^{-}_{3,3}$)

4.1 Properties of modular non-orientable graphs

The following Lemma will show that every quadrilateral must be a rectangle. This is the step where modularity will be used.

Lemma 81. *Let $G_{\mathrm{w}} = (V, E, \mathrm{w})$ be a modular good graph. Let (a, b, c, d) be a simple cycle in G_{w} of length 4. Then $\mathrm{w}(a, b) = \mathrm{w}(c, d)$ and $\mathrm{w}(b, c) = \mathrm{w}(d, a)$.*

Proof. Note that the absence of redundant edges forces that $\{a, c\} \notin E$ and $\{b, d\} \notin E$. In other words, the cycle (a, b, c, d) is induced. It has to be the case that

$$\mathrm{w}(a, b) + \mathrm{w}(b, c) = \mathrm{w}(c, d) + \mathrm{w}(d, a)$$

because otherwise $\{a, b, c\}$ or $\{c, d, a\}$ would be a median-less triplet. In a similar manner, we observe that:

$$\mathrm{w}(b, c) + \mathrm{w}(c, d) = \mathrm{w}(d, a) + \mathrm{w}(a, b)$$

Summing both equations together gives:

$$\mathrm{w}(a, b) + 2 \cdot \mathrm{w}(b, c) + \mathrm{w}(c, d) = \mathrm{w}(c, d) + 2 \cdot \mathrm{w}(d, a) + \mathrm{w}(a, b)$$

It immediately follows that $\mathrm{w}(b, c) = \mathrm{w}(d, a)$ and then $\mathrm{w}(a, b) = \mathrm{w}(c, d)$. □

Definition 82. Let $G = (V, E)$ be a graph. Let $\{v_0, ..., v_{k-1}, v_k, ..., v_{2k-1}\} \subseteq V$ be a set of (not necessarily distinct) vertices such that $(v_{i-1}, v_i, v_{k+i}, v_{k+i-1})$ is a simple cycle in G for each $0 < i < k$, and additionally, $(v_{k-1}, v_k, v_0, v_{2k-1})$ is a simple cycle in G as well. In such a case, we say that $(v_0, \ldots, v_{2k-1})$ is an *orientation-reversing dual cycle* in G.

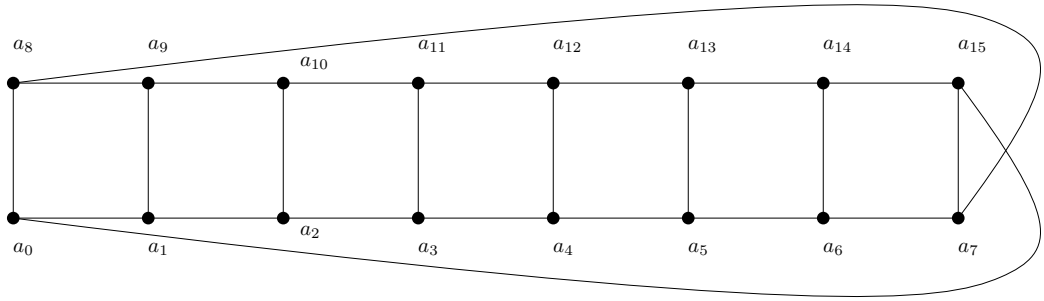
Unless vertices in $(v_0, \ldots, v_{2k-1})$ are repeated, the corresponding subgraph can be visualized as a möbius strip. However, vertices may be repeated. In fact, the same "faces" of G can appear as the quadruple $(v_{i-1}, v_i, v_{k+i}, v_{k+i-1})$ at different positions i in "different rotations" [1].

For instance, such a configuration inevitably appears when considering an orientation-reversing dual cycle in the (non-orientable) graph $K_{3,3}^{-}$.

Fact 83. A graph G is non-orientable if and only if [1] there exists an orientation-reversing dual cycle in G.

4.2 Construction of the model

We are given a fixed non-orientable modular good graph $G_{\mathrm{w}} = (V, E, \mathrm{w})$. We take one arbitrary orientation-reversing dual cycle $(a_0, \ldots, a_{2k-1})$ in G_{w} and we fix it to be "the" orientation-reversing dual cycle. In the end, we define a new symbol $a_{2k} \coloneqq a_0$ in order to make indexing easier.



We will construct new functions f, g, h, and m; different from those used in the construction for non-modular graphs. We start with an easy-to-imagine function f.

$$f_v^u(x, y) = \delta_u(x) + \delta_v(x) + \delta_u(y) + \delta_v(y)$$

In a major part of our construction, we work with $2k$ variables $(z_0, \ldots, z_{2k-1})$, which we denote $\bar{\mathbf{z}}$ for brevity. We furthermore define $z_{2k} \coloneqq z_0$ for our convenience. A composite function g follows.

$$g(\bar{\mathbf{z}}) = f_{a_k}^{a_0}(z_0, z_k) + f_{a_{k+1}}^{a_1}(z_1, z_{k+1}) + \cdots + f_{a_{2k-1}}^{a_{k-1}}(z_{k-1}, z_{2k-1})$$

Let M be a sufficiently large constant that depends only on G. Using the above-defined function g and a sufficiently-high value M, a function h follows.

$$h(\bar{\mathbf{z}}) = M \cdot g(\bar{\mathbf{z}}) + \mathrm{d}(z_0, z_1) + \mathrm{d}(z_1, z_2) + \cdots + \mathrm{d}(z_{2k-1}, z_{2k})$$

In the end, we get the objective function by summing h over all edges of the Max-Cut graph $H = (W, F)$. Our VCSP($\Gamma_{G_{\mathrm{w}}}$) instance (with variable set S, labeling $\mathbf{x}$, and objective function m) is built as follows.

$$S = \{w_0, w_1, \ldots, w_{k-1} : w \in W\}$$

$$m(\mathbf{x}) = \sum_{\{u,v\} \in F} h(\mathbf{x}_{u_0}, \mathbf{x}_{u_1}, \ldots, \mathbf{x}_{u_{k-1}}, \mathbf{x}_{v_0}, \mathbf{x}_{v_1}, \ldots, \mathbf{x}_{v_{k-1}})$$

4.3 Analysis of the model

Lemma 84. *Let $\{u, v\} \in E$. Let f be the function $V^2 \to \mathbb{Q}$ as defined in Section 4.2.*

$$\arg\min(f_v^u) = \{(u,u), (u,v), (v,u), (v,v)\}$$

Proof. Let us define a unary function $\hat{f}_v^u(x) = \delta_u(x) + \delta_v(x)$, which is a part of original f, because $f_v^u(x,y) = \hat{f}_v^u(x) + \hat{f}_v^u(y)$. Since d is a metric, the triangle inequality tells us that

$$\hat{f}_v^u(u) = \mathrm{d}(u,v) = \hat{f}_v^u(v)$$

is the minimum of $\hat{f}$. The non-redundancy of the edge $\{u, v\}$ implies that:

$$\forall w \in (V \setminus \{u,v\}) : \hat{f}_v^u(w) > \mathrm{d}(u,v)$$

Therefore, $\arg\min(\hat{f}_v^u) = \{u, v\}$, and thus $\arg\min(f_v^u) = \{u, v\} \times \{u, v\}$. □

Lemma 85. *Let h be the function $V^{2k} \to \mathbb{Q}$ as defined in Section 4.2.*

$$\arg\min(h) = \{(a_0, a_1, \ldots, a_{2k-1}), (a_k, a_{k+1}, \ldots, a_{2k-1}, a_0, a_1, \ldots, a_{k-1})\}$$

Proof. The definition of the orientation-reversing dual cycle, together with Lemma 81, implies that:

$$\mathrm{w}(a_0, a_k) = \mathrm{w}(a_1, a_{k+1}) = \cdots = \mathrm{w}(a_{k-1}, a_{2k-1})$$

Lemma 81 also implies that:

$$\forall i \in \{0, 1, \ldots k-1\} : \mathrm{w}(a_i, a_{i+1}) = \mathrm{w}(a_{i+k}, a_{i+k+1})$$

We denote as the circumference of the cycle by:

$$q \coloneqq \mathrm{w}(a_0, a_1) + \cdots + \mathrm{w}(a_{2k-1}, a_{2k}) = 2 \cdot (\mathrm{w}(a_0, a_1) + \cdots + \mathrm{w}(a_{k-1}, a_k))$$

Now we consider the value which we want to prove to be the minimum:

$$h(a_0, a_1, \ldots, a_{2k-1}) = 2kM \cdot \mathrm{w}(a_0, a_k) + q$$

If we rotate the assignment by k positions, we get the same value:

$$h(a_k, a_{k+1}, \ldots, a_{2k-1}, a_0, a_1, \ldots, a_{k-1}) = 2kM \cdot \mathrm{w}(a_0, a_k) + q$$

What do we know about general value $h(\bar{\mathbf{z}})$? Since only one summand in h is multiplied by M, it is obvious that in order to minimize $h(\bar{\mathbf{z}})$, we have to minimize $g(\bar{\mathbf{z}})$. Lemma 84, when applied to each summand in g, implies that:

$$\forall i \in \{0, 1, \ldots k-1\} : z_i, z_{i+k} \in \{a_i, a_{i+k}\}$$

In the rest of this proof, we will consider only potentially-minimum vectors $\bar{\mathbf{z}}$, i.e. restricted to $z_i, z_{i+k} \in \{a_i, a_{i+k}\}$. Let us count the number of "switches between long cycles".

$$J(\bar{\mathbf{z}}) \coloneqq \{i \in \{0, 1, \dots 2k-1\} \mid (z_i, z_{i+1}) \notin \{(a_i, a_{i+1}), (a_{i+k}, a_{i+k+1})\}\}$$

Let us denote $s(\bar{\mathbf{z}}) \coloneqq |J(\bar{\mathbf{z}})|$. Without loss of generality, we choose i such that $z_i = a_i \ , \ z_{i+1} = a_{i+k+1}$. Modularity of G implies that $\{a_i, a_{i+k}, a_{i+k+1}\}$ must have a median. We see that $\mathrm{I}(a_i, a_{i+k}) = \{a_i, a_{i+k}\}$ (otherwise the edge $\{a_i, a_{i+k}\} \in E$ would be redundant) and that $\mathrm{I}(a_{i+k}, a_{i+k+1}) = \{a_{i+k}, a_{i+k+1}\}$ (otherwise the edge $\{a_{i+k}, a_{i+k+1}\} \in E$ would be redundant). We obtained $\mathrm{I}(a_i, a_{i+k}) \cap \mathrm{I}(a_{i+k}, a_{i+k+1}) = \{a_{i+k}\}$, so the vertex a_{i+k} must be the median of $\{a_i, a_{i+k}, a_{i+k+1}\}$, thus:

$$\mathrm{d}(a_i, a_{i+k+1}) = \mathrm{w}(a_i, a_{i+k}) + \mathrm{w}(a_{i+k}, a_{i+k+1})$$

Note that $\mathrm{w}(a_{i+k}, a_{i+k+1})$ is already included in q, but $\mathrm{w}(a_i, a_{i+k})$ will be paid for each switch between long cycles, where $\mathrm{w}(a_i, a_{i+k})$ has always the same value as $\mathrm{w}(a_0, a_k)$. As a result:

$$\begin{aligned} h(\bar{\mathbf{z}}) = {} & 2kM \cdot \mathrm{w}(a_0, a_k) + \\ & + \Big(q - \sum_{i \in J(\bar{\mathbf{z}})} \mathrm{w}(a_{i+k}, a_{i+k+1})\Big) + \sum_{i \in J(\bar{\mathbf{z}})} \Big(\mathrm{w}(a_i, a_{i+k}) + \mathrm{w}(a_{i+k}, a_{i+k+1})\Big) \end{aligned}$$

This can be simplified to:

$$h(\bar{\mathbf{z}}) = 2kM \cdot \mathrm{w}(a_0, a_k) + q \ + \sum_{i \in J(\bar{\mathbf{z}})} \mathrm{w}(a_i, a_{i+k})$$

Putting everything together gives:

$$h(\bar{\mathbf{z}}) = h(a_0, a_1, \dots, a_{2k-1}) + s(\bar{\mathbf{z}}) \cdot \mathrm{w}(a_0, a_k)$$

It is easy to check that $s(\bar{\mathbf{z}}) = 0$ implies that:

$$\bar{\mathbf{z}} \in \{(a_0, a_1, \dots, a_{2k-1}), (a_k, a_{k+1}, \dots, a_{2k-1}, a_0, a_1, \dots, a_{k-1})\}$$

Since G_{w} is a good graph, $\mathrm{w}(a_0, a_k) > 0$, thus the $\arg\min(h)$ contains only the two vectors $(a_0, a_1, \dots, a_{2k-1})$ and $(a_k, a_{k+1}, \dots, a_{2k-1}, a_0, a_1, \dots, a_{k-1})$. □

Lemma 86. *There is a number $T \in \mathbb{R}$ and a number $t \in \mathbb{R}^+$ such that the vector $(a_0, a_1, \dots, a_{2k-1}) \in V^{2k}$, where $a_0 \neq a_k$, minimizes h in such a way that:*

- $h(a_0, a_1, \dots, a_{2k-1}) = h(a_k, a_{k+1}, \dots, a_{2k-1}, a_0, a_1, \dots, a_{k-1}) = T$
- $h(a_0,\ a_1,\ \dots, a_{k-2},\ a_{k-1},\ \ a_0,\ a_1,\ \dots\ , a_{k-2},\ a_{k-1}) = T + t$
- $h(a_k, a_{k+1}, \dots, a_{2k-2}, a_{2k-1},\ a_k, a_{k+1}, \dots, a_{2k-2}, a_{2k-1}) = T + t$
- $h(\bar{\mathbf{z}}) \geq T + t$ *for all other vectors* $\bar{\mathbf{z}} \in V^{2k}$

Proof. Since $z_0 = z_{2k}$, the value of $s(\bar{\mathbf{z}})$ is always pair for $z_i, z_{i+k} \in \{a_i, a_{i+k}\}$. From Lemma 84, we know that in order to have $h(\bar{\mathbf{z}}) = T + o(M)$, we are bound to $z_i, z_{i+k} \in \{a_i, a_{i+k}\}$. It follows that $h(a_0, a_1, \ldots, a_{2k-1}) + 2 \cdot \mathrm{w}(a_0, a_k)$ is the second lowest value of the function h.

As a result, we can define $T \coloneqq 2kM \cdot \mathrm{w}(a_0, a_k) + q$, and $t \coloneqq 2 \cdot \mathrm{w}(a_0, a_k)$. It remains to check that the desired value $T + t$ is met by the two vectors (see item 2 and item 3 of the Lemma).

$$J(a_0,\ a_1,\ \ldots, a_{k-2},\ a_{k-1},\ \ a_0,\ a_1,\ \ \ldots\ , a_{k-2},\ a_{k-1}\ ) = \{k-1, 2k-1\}$$

$$J(a_k, a_{k+1}, \ldots, a_{2k-2}, a_{2k-1},\ a_k, a_{k+1}, \ldots, a_{2k-2}, a_{2k-1}) = \{k-1, 2k-1\}$$

It follows that:

$$h(a_0,\ a_1,\ \ldots, a_{k-2},\ a_{k-1},\ \ a_0,\ a_1,\ \ \ldots\ , a_{k-2},\ a_{k-1}\ ) = T + t$$

$$h(a_k, a_{k+1}, \ldots, a_{2k-2}, a_{2k-1},\ a_k, a_{k+1}, \ldots, a_{2k-2}, a_{2k-1}) = T + t$$

They are not the only two vectors $\bar{\mathbf{z}}$ which give $h(\bar{\mathbf{z}}) = T + t$, but the uniqueness is not needed here. Only the minimum value T is constrained to be met by unique two vectors. These two vectors (see item 1) are provided by Lemma 85 with a proof of their uniqueness. □

4.4 Hardness results

Definition 87. Let Γ be a finite-valued constraint language. We define $\langle\Gamma\rangle$ to be the set of all functions $f(x_1, \ldots, x_n)$ such that there exists an instance I of VCSP(Γ) with an objective function $\Phi(x_1, \ldots, x_n, x_{n+1}, \ldots, x_s)$ with the following property.

$$f(x_1, \ldots, x_n) = \min_{(x_{n+1}, \ldots, x_s) \in D^{s-n}} \Phi(x_1, \ldots, x_n, x_{n+1}, \ldots, x_s)$$

The set $\langle\Gamma\rangle$ is sometimes called an *expressive power* [46] of the finite-valued constraint language Γ.

Observation 88. Let Γ be a fixed finite-valued constraint language. If I is an instance of VCSP(Γ), we can introduce new variables for any subset of arguments of any function calls used in I, with only a constant overhead in the size of the instance (where the multiplicative constant depends on Γ, which is fixed), and reorder the arguments in any way, without changing how hard it is to solve the instance.

We observe that VCSP(Γ) has the same complexity as VCSP($\langle\Gamma\rangle$). The language $\langle\Gamma\rangle$ can therefore be a useful tool for obtaining lower bounds for the complexity of the problem VCSP(Γ).

Theorem 89. *If $G_{\mathrm{w}} = (V, E, \mathrm{w})$ is a modular non-orientable good graph, then the VCSP($\Gamma_{G_{\mathrm{w}}}$) is* **NP***-complete.*

Proof. It is easy to see that the problem is in **NP**.

As for **NP**-hardness, we will use the condition (MC) from [47]. We declare a new function $\bar{h}$ based on the function h.

$$\bar{h}(x, y) \coloneqq \min_{\bar{\mathbf{z}} \in V^{2k}} h(x, z_1, \ldots, z_{k-2}, z_{k-1},\ y, z_{k+1}, \ldots, z_{2k-2}, z_{2k-1})$$

We can see that $\bar{h} \in \langle \Gamma_{G_{\mathrm{w}}} \rangle$. Lemma 86 implies:

$$\bar{h}(a_0, a_0) = \bar{h}(a_k, a_k) > \bar{h}(a_0, a_k) = \bar{h}(a_k, a_0)$$

We have observed that $f^{a_0}_{a_k}$ is a function such that $\arg\min(f^{a_0}_{a_k}) = \{a_0, a_k\}$. The existence of the binary function $\bar{h}$, together with the existence of the unary function $f^{a_0}_{a_k}$, provides the condition (MC) from [47]. A theorem from [47] (based on a weaker proposition from [48] regarding General-Valued CSP) implies that VCSP($\langle \Gamma_{G_{\mathrm{w}}} \rangle$) is **NP**-hard, thus VCSP($\Gamma_{G_{\mathrm{w}}}$) is **NP**-hard as well. As a result, the problem VCSP($\Gamma_{G_{\mathrm{w}}}$) is **NP**-complete.

Alternatively, we could have instead used that Lemma 85 implies that $\arg\min(\bar{h}) = \{(a_0, a_k), (a_k, a_0)\}$. This provides a slightly different condition (MC) from [46].

To avoid confusion, we better note that the original condition (MC) from [47] is denoted by (MC') in [46].

Either way, the condition (MC) or (MC') provides a reduction from the Max-Cut Problem which we know to be **NP**-hard. We could have constructed the reduction ourselves, as we did in the proof for non-modular graphs, but an explicit construction would be slightly more complicated here, so we decided to use the modern results from [47] or [46] instead.

It could be worth mentioning that another theorem from [46], based on the works of [49], also implies that $\Gamma_{G_{\mathrm{w}}}$ cannot admit any symmetric fractional polymorphism (see Definition 95). □

Corollary 90. *The Minimum 0-Extension Problem for any non-orientable good graph is* **NP***-complete.*

Proof. By definition, a non-orientable good graph is either a non-orientable non-modular good graph or a non-orientable modular good graph. We have established the **NP**-completeness for both cases (see Corollary 80 for the former and Theorem 89 for the latter result). As the final step, we apply Lemma 55. □

5. Easy cases

In this chapter, we attempt to generalize some of the positive results about the Minimum 0-Extension Problem from simple graphs to weighted graphs. That is, we attempt to identify a certain subset of modular orientable good graphs such that, for every good graph G_{w} from this subset, the Minimum 0-Extension Problem for G_{w} can be computed in polynomial time.

5.1 Algebraic tools

In this section, we introduce tools related to the following auxiliary task. We are given two (or more) solutions to a particular subproblem. Can we combine them together to create a new solution with some useful properties?

Definition 91. Let ω be a function of the type $(D^m \to D) \to \mathbb{Q}_0^+$ such that $\sum_{g\colon D^m \to D} \omega(g) = 1$. We say that ω is a an m-ary *fractional operation* on D. We define $\mathrm{supp}(\omega) = \{g : D^m \to D \mid \omega(g) > 0\}$.

On the informal level, we could say that "applying ω" corresponds to "applying g with probability $\omega(g)$ for every g in $\mathrm{supp}(\omega)$" where each applying is elementwise, using the same g at all positions. However, the values $\omega(g)$ are in fact weights, not probabilities.

On the formal level, we need only to define what it means to apply a cost function f to the fractional operation ω of an m-tuple of inputs $(x_1, \ldots, x_m)$. We have $f : D^n \to \mathbb{Q}$, $\omega : (D^m \to D) \to \mathbb{Q}_0^+$, and $x_i \in D^n$ for all $1 \le i \le m$.

$$f\Big(\omega(x_1, \ldots, x_m)\Big) = \\ = \sum_{g \in \mathrm{supp}(\omega)} \omega(g) \cdot f\Big(g(x_1[1], ..., x_m[1]),\ \ldots\ldots, g(x_1[n], ..., x_m[n])\Big)$$

Definition 92. Let $f : D^n \to \mathbb{Q}$ be a cost function (of arity n). Let ω be an m-ary fractional operation on D. We say that the cost function f *admits* the fractional operation ω if:

$$\forall x_1 \in D^n \ldots \forall x_m \in D^n : \quad f\Big(\omega(x_1, \ldots, x_m)\Big) \le \frac{1}{m} \cdot \sum_{i=1}^{m} f(x_i)$$

Definition 93. Let Γ be a finite-valued constraint language over D. Let ω be a fractional operation on D. If all cost functions from Γ admit ω, then we say that ω is a *fractional polymorphism* for Γ.

Definition 94. An m-ary operation g is *symmetric* if it preserves its output under all permutations of its inputs.

$$\forall (x_1, \ldots, x_m) \in D^m \ \ \forall \pi \in S_m \ : \ g(x_1, \ldots, x_m) = g(x_{\pi(1)}, \ldots, x_{\pi(m)})$$

For $m = 2$, the notion of a symmetric operation is equivalent to the notion of a commutative operation, i.e. $\forall x, y \in D : g(x, y) = g(y, x)$.

Definition 95. Let ω be a fractional polymorphism for Γ. If all operations in $\mathrm{supp}(\omega)$ are symmetric, then we say that ω is a *symmetric* fractional polymorphism for Γ.

5.2 Linear optimization tools

In this section, we present powerful tools that stem from the study of linear functions and linear inequalities.

Definition 96. Let $A \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{c} \in \mathbb{R}^n$. A *linear program* is a task to search for a vector $\mathbf{x} \in \mathbb{R}^n$ such that $A\mathbf{x} \leq \mathbf{b}$ holds and $\mathbf{c}^T\mathbf{x}$ is minimum.

Fact 97. A linear program can be solved in a polynomial time, e.g. by using the ellipsoid method [50], thus the decision version is in **P**.

In practice, however, the simplex method is used instead. Even though the simplex method has weaker theoretical guarantees — its worst-case time complexity is exponential [51], it runs surprisingly fast in typical examples; and it was (much later) proved that the average-case time complexity of the simplex method is polynomial [52].

Definition 98. Let Γ be a finite-valued constraint language over a domain D. Let $I = (S, D, \Phi, R)$ be an instance of VCSP(Γ). *Basic LP Relaxation* (where "LP" stands for "Linear Program") is defined as follows [10] (but the first appearance of a similar technique found in [53]).

Variables will be written in the form of functions. Informally speaking, λ_ϕ corresponds to the joint distribution of arguments of the cost function ϕ, whereäs β_s corresponds to the marginal distribution of the variable s.

$$\forall \phi \in \Phi \ \forall \mathbf{x}_\phi \in D^{\mathrm{ar}(\phi)} : \ \lambda_\phi(\mathbf{x}_\phi) \geq 0$$

$$\forall s \in S \ \forall a \in D : \ \beta_s(a) \geq 0$$

We will denote the variable used as the j-th argument of the cost function ϕ in the instance I using a locally-defined notation $S[[I, \phi, j]]$. Linear program:

$$\mathrm{BLP}(I) \ = \ \min \sum_{\phi \in \Phi} \left(\sum_{\mathbf{x}_\phi \in D^{\mathrm{ar}(\phi)}} \lambda_\phi(\mathbf{x}_\phi) \cdot \phi(\mathbf{x}_\phi) \right)$$

$$\forall \phi \in \Phi \ \forall j \in \{1, \ldots, \mathrm{ar}(\phi)\} \ \forall a \in D : \sum_{\substack{\mathbf{x}_\phi \in D^{\mathrm{ar}(\phi)}: \\ \mathbf{x}_\phi[j] = a}} \lambda_\phi(\mathbf{x}_\phi) = \beta_{S[[I,\phi,j]]}(a)$$

$$\forall s \in S : \ \sum_{a \in D} \beta_s(a) = 1$$

In order to turn it into a decision variant, we add the condition $\mathrm{BLP}(I) \leq R$. Formally written, we have to replace the first line by the explicit inequality condition:

$$\sum_{\phi \in \Phi} \left(\sum_{\mathbf{x} \in D^{\mathrm{ar}(\phi)}} \lambda_\phi(\mathbf{x}) \cdot \phi(\mathbf{x}) \right) \ \leq \ R$$

The value $\mathrm{BLP}(I)$ provides a lower bound for the solution of the instance I. In particular $\mathrm{BLP}(I) > R$ implies that I is not satisfiable. If there exists an integer solution that attains the minimum value of $\mathrm{BLP}(I)$, then $\mathrm{BLP}(I) \leq R$ implies that I is satisfiable and the labeling $\mathbf{x}$ such that $\forall s \in S : \beta_s(\mathbf{x}_s) = 1$ is the corresponding solution of the instance I.

If every instance I of VCSP(Γ) has an integer solution that attains the minimum value of $\mathrm{BLP}(I)$, then we say that the Basic LP Relaxation *solves* VCSP(Γ).

5.3 Results

In this section, we will prove that, for two special types of the template G_{w}, the Minimum 0-Extension Problem for G_{w} is in **P**.

Lemma 55 is implicitly used in both theorems. These positive results are presented as the last part of this work for a reason. We will utilize the newly-introduced tools (fractional polymorphisms and linear programming) in both proofs so we will not have to construct the polynomial algorithms explicitly.

Fact 99. Existence of a (binary) symmetric fractional polymorphism for Γ implies [49] that the Basic LP Relaxation solves VCSP(Γ), thus VCSP(Γ) $\in$ **P**.

Theorem 100. *Let $G_{\mathrm{w}} = (V, E, \mathrm{w})$ be a good graph. If $|E| = |V| - 1$ and there exists a simple path of length $|E|$ in G, then the VCSP($\Gamma_{G_{\mathrm{w}}}$) is in* **P**.

Informally speaking, the above-mentioned condition says "If G is a path".

Proof. We construct a symmetric binary fractional polymorphism for $\Gamma_{G_{\mathrm{w}}}$. We start by operations l for "left" and r for "right". We name the vertices in such a way that $V = (v_1, v_2, \ldots, v_n)$ form a (simple) path. If $a \leq b$, we let $l(v_a, v_b) = v_a$ and $r(v_a, v_b) = v_b$, else we let $l(v_a, v_b) = v_b$ and $r(v_a, v_b) = v_a$. Then, we define a symmetric binary fractional operation ω by setting weights $\omega(l) = 1/2 = \omega(r)$.

It is trivial to check that all unary functions from $\Gamma_{G_{\mathrm{w}}}$ admit ω. Let us move to the binary function $\mathrm{d}(u, v)$ that measures the distance between labels assigned to two adjacent vertices u and v.

We are given two solutions $\mathbf{x}, \mathbf{y} : S \to V$. We are interested in comparing $\mathrm{d}(\mathbf{x}_u, \mathbf{x}_v) + \mathrm{d}(\mathbf{y}_u, \mathbf{y}_v)$ against $\mathrm{d}(l(\mathbf{x}_u, \mathbf{y}_u), l(\mathbf{x}_v, \mathbf{y}_v)) + \mathrm{d}(r(\mathbf{x}_u, \mathbf{y}_u), r(\mathbf{x}_v, \mathbf{y}_v))$.

We define a relation $\preccurlyeq$ which means "is to the left of (or identical to)", formally:

$$\mathbf{a} \preccurlyeq \mathbf{b} \iff l(\mathbf{a}, \mathbf{b}) = \mathbf{a} \iff r(\mathbf{a}, \mathbf{b}) = \mathbf{b}$$

If $\mathbf{a} = v_a$ and $\mathbf{b} = v_b$, we can also state:

$$\mathbf{a} \preccurlyeq \mathbf{b} \iff a \leq b$$

Without loss of generality, we can assume $\mathbf{x}_u \preccurlyeq \mathbf{y}_u$. Generality is not lost because we can swap $\mathbf{x}$ with $\mathbf{y}$ because all cost functions and operations we work with are commutative.

There are two main cases we need to care about. Either v's are in the same order as u's, that is $\mathbf{x}_v \preccurlyeq \mathbf{y}_v$, or v's are in the opposite order to u's, that is $\mathbf{x}_v \succ \mathbf{y}_v$. The first case implies

$$\mathrm{d}(l(\mathbf{x}_u, \mathbf{y}_u), l(\mathbf{x}_v, \mathbf{y}_v)) + \mathrm{d}(r(\mathbf{x}_u, \mathbf{y}_u), r(\mathbf{x}_v, \mathbf{y}_v)) = \mathrm{d}(\mathbf{x}_u, \mathbf{x}_v) + \mathrm{d}(\mathbf{y}_u, \mathbf{y}_v)$$

by the identity between all 4 arguments at respective positions. The second case leads to comparing $\mathrm{d}(\mathbf{x}_u, \mathbf{x}_v) + \mathrm{d}(\mathbf{y}_u, \mathbf{y}_v)$ against $\mathrm{d}(\mathbf{x}_u, \mathbf{y}_v) + \mathrm{d}(\mathbf{y}_u, \mathbf{x}_v)$. For the sake of brevity, we declare symbols P and Q as:

$$P \coloneqq \mathrm{d}(\mathbf{x}_u, \mathbf{x}_v) + \mathrm{d}(\mathbf{y}_u, \mathbf{y}_v)$$

$$Q \coloneqq \mathrm{d}(\mathbf{x}_u, \mathbf{y}_v) + \mathrm{d}(\mathbf{y}_u, \mathbf{x}_v)$$

We want $Q \leq P$.

This inequality would mean that ω made the cost "cheaper". A proof by exhaustion follows. The third case will be accompanied by an explanatory illustration.

There are 24 permutations of the symbols $\{\text{“}\mathbf{x}_u\text{”}, \text{“}\mathbf{y}_u\text{”}, \text{“}\mathbf{x}_v\text{”}, \text{“}\mathbf{y}_v\text{”}\}$ in total. Only 6 of them place “$\mathbf{x}_u$” before “$\mathbf{y}_u$” and at the same time “$\mathbf{y}_v$” before “$\mathbf{x}_v$”.

- If $\mathbf{x}_u \preccurlyeq \mathbf{y}_u \preccurlyeq \mathbf{y}_v \preccurlyeq \mathbf{x}_v$ then:

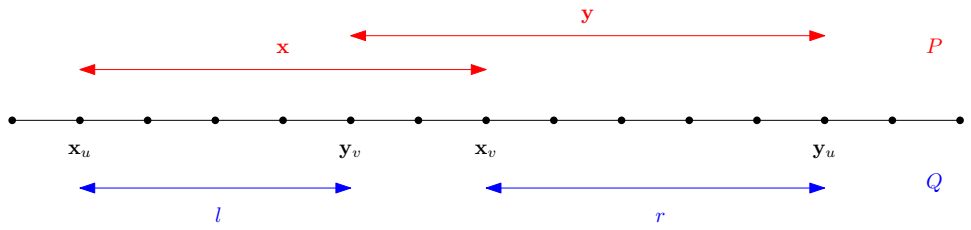
  $P = \mathrm{d}(\mathbf{x}_u, \mathbf{x}_v) + \mathrm{d}(\mathbf{y}_u, \mathbf{y}_v) \qquad Q = \mathrm{d}(\mathbf{x}_u, \mathbf{x}_v) + \mathrm{d}(\mathbf{y}_u, \mathbf{y}_v) \qquad P = Q$

- If $\mathbf{x}_u \preccurlyeq \mathbf{y}_v \preccurlyeq \mathbf{y}_u \preccurlyeq \mathbf{x}_v$ then:

  $P = \mathrm{d}(\mathbf{x}_u, \mathbf{x}_v) + \mathrm{d}(\mathbf{y}_v, \mathbf{y}_u) \qquad Q = \mathrm{d}(\mathbf{x}_u, \mathbf{x}_v) - \mathrm{d}(\mathbf{y}_v, \mathbf{y}_u) \qquad P \geq Q$

- If $\mathbf{x}_u \preccurlyeq \mathbf{y}_v \preccurlyeq \mathbf{x}_v \preccurlyeq \mathbf{y}_u$ then:

  $P = \mathrm{d}(\mathbf{x}_u, \mathbf{y}_u) + \mathrm{d}(\mathbf{y}_v, \mathbf{x}_v) \qquad Q = \mathrm{d}(\mathbf{x}_u, \mathbf{y}_u) - \mathrm{d}(\mathbf{y}_v, \mathbf{x}_v) \qquad P \geq Q$



- If $\mathbf{y}_v \preccurlyeq \mathbf{x}_v \preccurlyeq \mathbf{x}_u \preccurlyeq \mathbf{y}_u$ then:

  $P = \mathrm{d}(\mathbf{y}_v, \mathbf{y}_u) + \mathrm{d}(\mathbf{x}_v, \mathbf{x}_u) \qquad Q = \mathrm{d}(\mathbf{y}_v, \mathbf{y}_u) + \mathrm{d}(\mathbf{x}_v, \mathbf{x}_u) \qquad P = Q$

- If $\mathbf{y}_v \preccurlyeq \mathbf{x}_u \preccurlyeq \mathbf{x}_v \preccurlyeq \mathbf{y}_u$ then:

  $P = \mathrm{d}(\mathbf{y}_v, \mathbf{y}_u) + \mathrm{d}(\mathbf{x}_u, \mathbf{x}_v) \qquad Q = \mathrm{d}(\mathbf{y}_v, \mathbf{y}_u) - \mathrm{d}(\mathbf{x}_u, \mathbf{x}_v) \qquad P \geq Q$

- If $\mathbf{y}_v \preccurlyeq \mathbf{x}_u \preccurlyeq \mathbf{y}_u \preccurlyeq \mathbf{x}_v$ then:

  $P = \mathrm{d}(\mathbf{y}_v, \mathbf{x}_v) + \mathrm{d}(\mathbf{x}_u, \mathbf{y}_u) \qquad Q = \mathrm{d}(\mathbf{y}_v, \mathbf{x}_v) - \mathrm{d}(\mathbf{x}_u, \mathbf{y}_u) \qquad P \geq Q$

In all cases, we got $Q \leq P$. We recall that $l(\mathbf{x}_u, \mathbf{y}_u) = \mathbf{x}_u$, $r(\mathbf{x}_u, \mathbf{y}_u) = \mathbf{y}_u$, $l(\mathbf{x}_v, \mathbf{y}_v) = \mathbf{y}_v$, $r(\mathbf{x}_v, \mathbf{y}_v) = \mathbf{x}_v$; and that we put $Q = \mathrm{d}(\mathbf{x}_u, \mathbf{y}_v) + \mathrm{d}(\mathbf{y}_u, \mathbf{x}_v)$, $P = \mathrm{d}(\mathbf{x}_u, \mathbf{x}_v) + \mathrm{d}(\mathbf{y}_u, \mathbf{y}_v)$; thus we have:

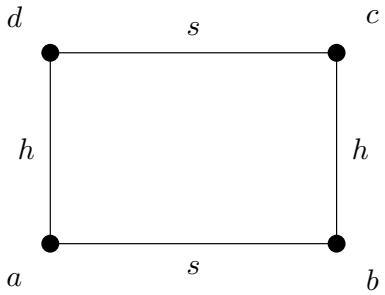
$$\mathrm{d}(l(\mathbf{x}_u, \mathbf{y}_u), l(\mathbf{x}_v, \mathbf{y}_v)) + \mathrm{d}(r(\mathbf{x}_u, \mathbf{y}_u), r(\mathbf{x}_v, \mathbf{y}_v)) \;\leq\; \mathrm{d}(\mathbf{x}_u, \mathbf{x}_v) + \mathrm{d}(\mathbf{y}_u, \mathbf{y}_v)$$

We divide the inequality by two and obtain:

$$\mathrm{d}\Big(\omega((\mathbf{x}_u, \mathbf{x}_v), (\mathbf{y}_u, \mathbf{y}_v))\Big) \;\leq\; \frac{1}{2} \cdot \Big(\mathrm{d}(\mathbf{x}_u, \mathbf{x}_v) + \mathrm{d}(\mathbf{y}_u, \mathbf{y}_v)\Big)$$

We conclude that d admits ω. We already know that all $\delta_{v_1}, \ldots, \delta_{v_n}$ admit ω as well. We have checked every cost function from $\Gamma_{G_{\mathrm{w}}}$. It is easy to check that both operations l and r are symmetric, thus ω is a symmetric fractional polymorphism. As a result, Fact 99 implies that $\mathrm{VCSP}(\Gamma_{G_{\mathrm{w}}}) \in \mathbf{P}$. □

Theorem 101. *Let h and s be positive rational numbers. Let $G_{\mathrm{w}} = (V, E, \mathrm{w})$ be a (good) graph exactly in this form (where a, b, c, d are vertices; and h, s are weights of the edges):*



Then the VCSP($\Gamma_{G_{\mathrm{w}}}$) is in **P**.

Proof. We provide the following symmetric binary fractional polymorphism for $\Gamma_{G_{\mathrm{w}}}$. We start with defining two operations (p and q) by explicit tables.

p	a	b	c	d
a	a	a	d	d
b	a	b	c	d
c	d	c	c	d
d	d	d	d	d

q	a	b	c	d
a	a	b	b	a
b	b	b	b	b
c	b	b	c	c
d	a	b	c	d

We easily check that both the operations p and q are symmetric. We define a binary fractional operation ω by setting the weights to $\omega(p) = 1/2 = \omega(q)$.

We will first check whether all unary cost functions admit ω. This will be easy since almost all pairs $x, y \in V$ give $\{p(x, y), q(x, y)\} = \{x, y\}$ and thus preserve the result of $\delta_v(x) + \delta_v(y)$ unchanged for every $v \in V$. The only different pair of values is $\{a, c\}$, which gets mapped to $\{b, d\}$. Fortunately, we have:

$$\begin{array}{rcccccl}
\delta_a(a) + \delta_a(c) & = & 0 + (s + h) & = & s + h & = & \delta_a(b) + \delta_a(d) \\
\delta_b(a) + \delta_b(c) & = & s + h & = & 0 + (s + h) & = & \delta_b(b) + \delta_b(d) \\
\delta_c(a) + \delta_c(c) & = & (s + h) + 0 & = & h + s & = & \delta_c(b) + \delta_c(d) \\
\delta_d(a) + \delta_d(c) & = & h + s & = & (s + h) + 0 & = & \delta_d(b) + \delta_d(d)
\end{array}$$

We have checked that all four unary cost functions $\{\delta_a, \delta_b, \delta_c, \delta_d\}$ admit ω. As a matter of fact, they always do so with equality. It remains to check the binary cost function (distance d) for admitting ω.

We are interested in the value $\mathrm{d}(u, v)$. We are given two labelings: $\mathbf{x}$, $\mathbf{y}$. This gives us four relevant variables: $\mathbf{x}_u$, $\mathbf{x}_v$, $\mathbf{y}_u$, $\mathbf{y}_v$. Each variable can be assigned any value from $V = \{a, b, c, d\}$. This makes $4^4 = 256$ possibilities in total.

The form of the admissibility condition (whether d admits ω) is the following system of inequalities.

$$\begin{array}{l}
\forall\, (\mathbf{x}_u, \mathbf{x}_v, \mathbf{y}_u, \mathbf{y}_v) \in \{a, b, c, d\}^4 : \\
\quad \mathrm{d}(\mathbf{x}_u, \mathbf{x}_v) + \mathrm{d}(\mathbf{y}_u, \mathbf{y}_v) \;\geq\; \mathrm{d}(p(\mathbf{x}_u, \mathbf{y}_u), p(\mathbf{x}_v, \mathbf{y}_v)) + \mathrm{d}(q(\mathbf{x}_u, \mathbf{y}_u), q(\mathbf{x}_v, \mathbf{y}_v))
\end{array}$$

In order to simplify our task of checking these 256 inequalities, we will add a few symmetry-breaking conditions. We first recall that $\mathrm{d}(u,v) = \mathrm{d}(v,u)$. This allows us to swap $(\mathbf{x}_u, \mathbf{y}_u)$ with $(\mathbf{x}_v, \mathbf{y}_v)$. At the same time, the symmetry of ω (i.e. the property that both p and q are commutative operations) allows us to swap $(\mathbf{x}_u, \mathbf{x}_v)$ with $(\mathbf{y}_u, \mathbf{y}_v)$. As a consequence, we are free to require:

- The label $\mathbf{x}_u$ is (alphabetically) smallest.
- If $\mathbf{x}_u = \mathbf{x}_v$, then $\mathbf{y}_u$ is (alphabetically) before $\mathbf{y}_v$.
- If $\mathbf{x}_u = \mathbf{y}_u$, then $\mathbf{x}_v$ is (alphabetically) before $\mathbf{y}_v$.
- If $\mathbf{x}_u = \mathbf{y}_v$, then $\mathbf{x}_v$ is (alphabetically) before $\mathbf{y}_u$.

We will represent the admissibility condition schematically as follows. The cost function d is applied on rows. The fractional operation ω is applied on columns.

$$\begin{array}{cccc} \mathbf{x}_u & \mathbf{x}_v & \to & \mathrm{d}_{\mathbf{x}} \\ \mathbf{y}_u & \mathbf{y}_v & \to & \mathrm{d}_{\mathbf{y}} \\ \hline p(\mathbf{x}_u, \mathbf{y}_u) & p(\mathbf{x}_v, \mathbf{y}_v) & \to & \mathrm{d}_p \\ q(\mathbf{x}_u, \mathbf{y}_u) & q(\mathbf{x}_v, \mathbf{y}_v) & \to & \mathrm{d}_q \end{array} \qquad \Longrightarrow \qquad \mathrm{d}_{\mathbf{x}} + \mathrm{d}_{\mathbf{y}} \;\geq\; \mathrm{d}_p + \mathrm{d}_q$$

We finish our proof by exhaustion. In order to save horizontal space, we will denote the sum $h + s$ by R, so we will have $\mathrm{d}(a,c) = R = \mathrm{d}(b,d)$.

$\underline{\mathbf{x}_u = d}$:

$$\begin{array}{cccc} d & d & \to & 0 \\ d & d & \to & 0 \\ \hline d & d & \to & 0 \\ d & d & \to & 0 \end{array} \checkmark$$

$\underline{\mathbf{x}_u = c}$:

$$\begin{array}{cccc} c & c & \to & 0 \\ c & c & \to & 0 \\ \hline c & c & \to & 0 \\ c & c & \to & 0 \end{array} \checkmark \qquad \begin{array}{cccc} c & c & \to & 0 \\ c & d & \to & s \\ \hline c & d & \to & s \\ c & c & \to & 0 \end{array} \checkmark \qquad \begin{array}{cccc} c & c & \to & 0 \\ d & d & \to & 0 \\ \hline d & d & \to & 0 \\ c & c & \to & 0 \end{array} \checkmark \qquad \begin{array}{cccc} c & d & \to & s \\ c & d & \to & s \\ \hline c & d & \to & s \\ c & d & \to & s \end{array} \checkmark$$

$$\begin{array}{cccc} c & d & \to & s \\ d & c & \to & s \\ \hline d & d & \to & 0 \\ c & c & \to & 0 \end{array} \checkmark \qquad \begin{array}{cccc} c & d & \to & s \\ d & d & \to & 0 \\ \hline d & d & \to & 0 \\ c & d & \to & s \end{array} \checkmark$$

$\underline{\mathbf{x}_u = b}$:

$$\begin{array}{cccc} b & b & \to & 0 \\ b & b & \to & 0 \\ \hline b & b & \to & 0 \\ b & b & \to & 0 \end{array} \checkmark \qquad \begin{array}{cccc} b & b & \to & 0 \\ b & c & \to & h \\ \hline b & c & \to & h \\ b & b & \to & 0 \end{array} \checkmark \qquad \begin{array}{cccc} b & b & \to & 0 \\ b & d & \to & R \\ \hline b & d & \to & R \\ b & b & \to & 0 \end{array} \checkmark \qquad \begin{array}{cccc} b & b & \to & 0 \\ c & c & \to & 0 \\ \hline c & c & \to & 0 \\ b & b & \to & 0 \end{array} \checkmark$$

$$\begin{array}{llll} b & b & \to & 0 \\ c & d & \to & s \\ \hline c & d & \to & s \\ b & b & \to & 0 \end{array}\checkmark \qquad \begin{array}{llll} b & b & \to & 0 \\ d & d & \to & 0 \\ \hline d & d & \to & 0 \\ b & b & \to & 0 \end{array}\checkmark \qquad \begin{array}{llll} b & c & \to & h \\ b & c & \to & h \\ \hline b & c & \to & h \\ b & c & \to & h \end{array}\checkmark \qquad \begin{array}{llll} b & c & \to & h \\ b & d & \to & R \\ \hline b & d & \to & R \\ b & c & \to & h \end{array}\checkmark$$

$$\begin{array}{llll} b & d & \to & R \\ b & d & \to & R \\ \hline b & d & \to & R \\ b & d & \to & R \end{array}\checkmark \qquad \begin{array}{llll} b & c & \to & h \\ c & b & \to & h \\ \hline c & c & \to & 0 \\ b & b & \to & 0 \end{array}\checkmark \qquad \begin{array}{llll} b & c & \to & h \\ d & b & \to & R \\ \hline d & c & \to & s \\ b & b & \to & 0 \end{array}\checkmark \qquad \begin{array}{llll} b & d & \to & R \\ d & b & \to & R \\ \hline d & d & \to & 0 \\ b & b & \to & 0 \end{array}\checkmark$$

$$\begin{array}{llll} b & c & \to & h \\ c & c & \to & 0 \\ \hline c & c & \to & 0 \\ b & c & \to & h \end{array}\checkmark \qquad \begin{array}{llll} b & c & \to & h \\ c & d & \to & s \\ \hline c & d & \to & s \\ b & c & \to & h \end{array}\checkmark \qquad \begin{array}{llll} b & d & \to & R \\ c & c & \to & 0 \\ \hline c & d & \to & s \\ b & c & \to & h \end{array}\checkmark \qquad \begin{array}{llll} b & c & \to & h \\ d & c & \to & s \\ \hline d & c & \to & s \\ b & c & \to & h \end{array}\checkmark$$

$$\begin{array}{llll} b & c & \to & h \\ d & d & \to & 0 \\ \hline d & d & \to & 0 \\ b & c & \to & h \end{array}\checkmark \qquad \begin{array}{llll} b & d & \to & R \\ c & d & \to & s \\ \hline c & d & \to & s \\ b & d & \to & R \end{array}\checkmark \qquad \begin{array}{llll} b & d & \to & R \\ d & c & \to & s \\ \hline d & d & \to & 0 \\ b & c & \to & h \end{array}\checkmark \qquad \begin{array}{llll} b & d & \to & R \\ d & d & \to & 0 \\ \hline d & d & \to & 0 \\ b & d & \to & R \end{array}\checkmark$$

$\underline{\mathbf{x}_u = a}$; with at least 3 a's :

$$\begin{array}{llll} a & a & \to & 0 \\ a & a & \to & 0 \\ \hline a & a & \to & 0 \\ a & a & \to & 0 \end{array}\checkmark \qquad \begin{array}{llll} a & a & \to & 0 \\ a & b & \to & s \\ \hline a & a & \to & 0 \\ a & b & \to & s \end{array}\checkmark \qquad \begin{array}{llll} a & a & \to & 0 \\ a & c & \to & R \\ \hline a & d & \to & h \\ a & b & \to & s \end{array}\checkmark \qquad \begin{array}{llll} a & a & \to & 0 \\ a & d & \to & h \\ \hline a & d & \to & h \\ a & a & \to & 0 \end{array}\checkmark$$

$\underline{\mathbf{x}_u = a}$; with exactly 2 a's :

$$\begin{array}{llll} a & a & \to & 0 \\ b & b & \to & 0 \\ \hline a & a & \to & 0 \\ b & b & \to & 0 \end{array}\checkmark \qquad \begin{array}{llll} a & a & \to & 0 \\ b & c & \to & h \\ \hline a & d & \to & h \\ b & b & \to & 0 \end{array}\checkmark \qquad \begin{array}{llll} a & a & \to & 0 \\ b & d & \to & R \\ \hline a & d & \to & h \\ b & a & \to & s \end{array}\checkmark \qquad \begin{array}{llll} a & a & \to & 0 \\ c & c & \to & 0 \\ \hline d & d & \to & 0 \\ b & b & \to & 0 \end{array}\checkmark$$

$$\begin{array}{llll} a & a & \to & 0 \\ c & d & \to & s \\ \hline d & d & \to & 0 \\ b & a & \to & s \end{array}\checkmark \qquad \begin{array}{llll} a & a & \to & 0 \\ d & d & \to & 0 \\ \hline d & d & \to & 0 \\ a & a & \to & 0 \end{array}\checkmark \qquad \begin{array}{llll} a & b & \to & s \\ a & b & \to & s \\ \hline a & b & \to & s \\ a & b & \to & s \end{array}\checkmark \qquad \begin{array}{llll} a & b & \to & s \\ a & c & \to & R \\ \hline a & c & \to & R \\ a & b & \to & s \end{array}\checkmark$$

$$\begin{array}{llll} a & b & \to & s \\ a & d & \to & h \\ \hline a & d & \to & h \\ a & b & \to & s \end{array}\checkmark \qquad \begin{array}{llll} a & c & \to & R \\ a & c & \to & R \\ \hline a & c & \to & R \\ a & c & \to & R \end{array}\checkmark \qquad \begin{array}{llll} a & c & \to & R \\ a & d & \to & h \\ \hline a & d & \to & h \\ a & c & \to & R \end{array}\checkmark \qquad \begin{array}{llll} a & d & \to & h \\ a & d & \to & h \\ \hline a & d & \to & h \\ a & d & \to & h \end{array}\checkmark$$

$$\begin{array}{llll} a & b & \to & s \\ b & a & \to & s \\ \hline a & a & \to & 0 \\ b & b & \to & 0 \end{array}\checkmark \qquad \begin{array}{llll} a & b & \to & s \\ c & a & \to & R \\ \hline d & a & \to & h \\ b & b & \to & 0 \end{array}\checkmark \qquad \begin{array}{llll} a & b & \to & s \\ d & a & \to & h \\ \hline d & a & \to & h \\ a & b & \to & s \end{array}\checkmark \qquad \begin{array}{llll} a & c & \to & R \\ c & a & \to & R \\ \hline d & d & \to & 0 \\ b & b & \to & 0 \end{array}\checkmark$$

$$\begin{array}{llll} a & c & \to & R \\ d & a & \to & h \\ \hline d & d & \to & 0 \\ a & b & \to & s \end{array} \checkmark \qquad \begin{array}{llll} a & d & \to & h \\ d & a & \to & h \\ \hline d & d & \to & 0 \\ a & a & \to & 0 \end{array} \checkmark$$

$\underline{\mathbf{x}_u = a}$; with no other a :

$$\begin{array}{llll} a & b & \to & s \\ b & b & \to & 0 \\ \hline a & b & \to & s \\ b & b & \to & 0 \end{array} \checkmark \qquad \begin{array}{llll} a & b & \to & s \\ b & c & \to & h \\ \hline a & c & \to & R \\ b & b & \to & 0 \end{array} \checkmark \qquad \begin{array}{llll} a & b & \to & s \\ c & b & \to & h \\ \hline d & b & \to & R \\ b & b & \to & 0 \end{array} \checkmark \qquad \begin{array}{llll} a & c & \to & R \\ b & b & \to & 0 \\ \hline a & c & \to & R \\ b & b & \to & 0 \end{array} \checkmark$$

$$\begin{array}{llll} a & b & \to & s \\ d & b & \to & R \\ \hline d & b & \to & R \\ a & b & \to & s \end{array} \checkmark \qquad \begin{array}{llll} a & d & \to & h \\ b & b & \to & 0 \\ \hline a & d & \to & h \\ b & b & \to & 0 \end{array} \checkmark \qquad \begin{array}{llll} a & b & \to & s \\ b & d & \to & R \\ \hline a & d & \to & h \\ b & b & \to & 0 \end{array} \checkmark$$

$$\begin{array}{llll} a & b & \to & s \\ c & c & \to & 0 \\ \hline d & c & \to & s \\ b & b & \to & 0 \end{array} \checkmark \qquad \begin{array}{llll} a & c & \to & R \\ b & c & \to & h \\ \hline a & c & \to & R \\ b & c & \to & h \end{array} \checkmark \qquad \begin{array}{llll} a & c & \to & R \\ c & b & \to & h \\ \hline d & c & \to & s \\ b & b & \to & 0 \end{array} \checkmark \qquad \begin{array}{llll} a & b & \to & s \\ c & d & \to & s \\ \hline d & d & \to & 0 \\ b & b & \to & 0 \end{array} \checkmark$$

$$\begin{array}{llll} a & b & \to & s \\ d & c & \to & h \\ \hline d & c & \to & h \\ a & b & \to & s \end{array} \checkmark \qquad \begin{array}{llll} a & c & \to & R \\ b & d & \to & R \\ \hline a & d & \to & h \\ b & c & \to & h \end{array} \checkmark \qquad \begin{array}{llll} a & d & \to & h \\ b & c & \to & h \\ \hline a & d & \to & h \\ b & c & \to & h \end{array} \checkmark \qquad \begin{array}{llll} a & c & \to & R \\ d & b & \to & R \\ \hline d & c & \to & h \\ a & b & \to & s \end{array} \checkmark$$

$$\begin{array}{llll} a & d & \to & h \\ c & b & \to & h \\ \hline d & d & \to & 0 \\ b & b & \to & 0 \end{array} \checkmark \qquad \begin{array}{llll} a & b & \to & s \\ d & d & \to & 0 \\ \hline d & d & \to & 0 \\ a & b & \to & s \end{array} \checkmark \qquad \begin{array}{llll} a & d & \to & h \\ b & d & \to & R \\ \hline a & d & \to & h \\ b & d & \to & R \end{array} \checkmark \qquad \begin{array}{llll} a & d & \to & h \\ d & b & \to & R \\ \hline d & d & \to & 0 \\ a & b & \to & s \end{array} \checkmark$$

$$\begin{array}{llll} a & c & \to & R \\ c & c & \to & 0 \\ \hline d & c & \to & s \\ b & c & \to & h \end{array} \checkmark \qquad \begin{array}{llll} a & c & \to & R \\ c & d & \to & s \\ \hline d & d & \to & 0 \\ b & c & \to & h \end{array} \checkmark \qquad \begin{array}{llll} a & c & \to & R \\ d & c & \to & s \\ \hline d & c & \to & s \\ a & c & \to & R \end{array} \checkmark \qquad \begin{array}{llll} a & d & \to & h \\ c & c & \to & 0 \\ \hline d & d & \to & 0 \\ b & c & \to & h \end{array} \checkmark$$

$$\begin{array}{llll} a & c & \to & R \\ d & d & \to & 0 \\ \hline d & d & \to & 0 \\ a & c & \to & R \end{array} \checkmark \qquad \begin{array}{llll} a & d & \to & h \\ c & d & \to & s \\ \hline d & d & \to & 0 \\ b & d & \to & R \end{array} \checkmark \qquad \begin{array}{llll} a & d & \to & h \\ d & c & \to & s \\ \hline d & d & \to & 0 \\ a & c & \to & R \end{array} \checkmark \qquad \begin{array}{llll} a & d & \to & h \\ d & d & \to & 0 \\ \hline d & d & \to & 0 \\ a & d & \to & h \end{array} \checkmark$$

We have checked the required inequality in 1 case where $\mathbf{x}_u = d$, in 6 cases where $\mathbf{x}_u = c$, in 20 cases where $\mathbf{x}_u = b$, and in 49 cases where $\mathbf{x}_u = a$. All of them hold. That makes 76 cases in total.

Let us analyze whether they are all there is. We must calculate how many inequalities were thrown away by the symmetry-breaking conditions.

If the quadruple $(\mathbf{x}_u, \mathbf{x}_v, \mathbf{y}_u, \mathbf{y}_v)$ has labels symmetric with respect to exactly one of {the vertical axis, or the horizontal axis, or both of the diagonal axes}, then the symmetry-breaking conditions made us check two inequalities at the same time. If the quadruple $(\mathbf{x}_u, \mathbf{x}_v, \mathbf{y}_u, \mathbf{y}_v)$ has none of the above-mentioned symmetries, then the symmetry-breaking conditions made us check four inequalities at the same time.

Let us count how many instances of each symmetry we had. It is an easy combinatorial task in which we are choosing labels for the quadruple $(\mathbf{x}_u, \mathbf{x}_v, \mathbf{y}_u, \mathbf{y}_v)$ from four possible values $\{a, b, c, d\}$.

- Fully symmetric, that is $\mathbf{x}_u = \mathbf{x}_v = \mathbf{y}_u = \mathbf{y}_v$:
  possible 4 choices
- Vertically symmetric, that is $\mathbf{x}_u = \mathbf{x}_v$ and $\mathbf{y}_u = \mathbf{y}_v$, but not fully symm.:
  possible 12 choices
- Horizontally symm., that is $\mathbf{x}_u = \mathbf{y}_u$ and $\mathbf{x}_v = \mathbf{y}_v$, but not fully symm.:
  possible 12 choices
- Diagonally symm., that is $\mathbf{x}_u = \mathbf{y}_v$ and $\mathbf{x}_v = \mathbf{y}_u$, but not fully symm.:
  possible 12 choices

Counting up to symmetries:

$$\frac{4}{1} + \frac{12}{2} + \frac{12}{2} + \frac{12}{2} + \frac{256 - 4 - 12 - 12 - 12}{4} =$$

$$= 4 + 6 + 6 + 6 + 54 = 76$$

This is equal to the number of inequalities that we have checked. We have not forgotten anything.

We have verified that the binary cost function d admits the fractional operation ω. The unary cost functions and the symmetricity property were discussed at the beginning of this proof.

As a result, we have learnt that ω is a symmetric fractional polymorphism. Using Fact 99, we conclude that the VCSP($\Gamma_{G_{\mathrm{w}}}$) is in **P**. □

Remark 102. We could have used the Burnside's Lemma [54] [55] for the final check in the proof of Theorem 101. The calculation would go as follows.

Our group of symmetries is isomorphic to the Klein group [56] (it has two generators: the swap between $\mathbf{x}$, $\mathbf{y}$ and the swap between u, v). The numbers of fixed points are 16, 16, 16, and 256, respectively. Their arithmetic average is equal to 76. This is equal to the number of inequalities we checked in the proof.

Remark 103. The assumptions that $\mathrm{w}(a, b) = \mathrm{w}(c, d)$ and $\mathrm{w}(a, d) = \mathrm{w}(b, c)$ in Theorem 101 seem to be necessary. Unless **P** = **NP** (see Observation 51), any of $\mathrm{w}(a, b) \neq \mathrm{w}(c, d)$ or $\mathrm{w}(a, d) \neq \mathrm{w}(b, c)$ would make the good graph G_{w} be non-modular (see Lemma 81) which would in turn make the VCSP($\Gamma_{G_{\mathrm{w}}}$) be **NP**-complete (see Corollary 80).

6. Conclusion

We first summarized the basics of graph theory, of complexity theory, and of the CSP. We put emphasis on the fixed-language Finite-Valued CSP, which we used as the model in all our proofs. Constructions of reductions followed. We showed that the Minimum 0-Extension Problem is **NP**-complete for all good non-modular graphs and for all good non-orientable graphs as well. We thereby generalized two Karzanov's [1] negative results from simple graphs to weighted graphs. In the last chapter, we introduced modern algebraic tools used for solving the VCSP efficiently. We showed two positive results using these tools — most notably, that the Minimum 0-Extension Problem for a "weighted pathgraph" is in **P**.

This thesis was assigned two goals. The main goal of this thesis was to generalize the hardness proof regarding the non-modular graphs to the weighted non-modular graphs. This goal was fully reached. The secondary goal was to determine which special classes of the weighted graphs allowed for a fast algorithm. This goal was fulfilled only to a small extend. Apart from that, we reached an additional goal by generalizing also the hardness proof regarding the non-orientable graphs to the weighted non-orientable graphs.

⋆ ⋆ ⋆

About two weeks after finishing my main proof (the one about non-modular graphs), I accidentally learnt that this result had already been discovered and published [57] in 2004 by Karzanov.

I conclude that I independently discovered an alternative proof of the same theorem. My proof uses a different formalism (the fixed-language Finite-Valued CSP, as opposed to metric spaces). The main idea of my proof is the same as the Karzanov's because we both built our reduction from the Max-Cut Problem based on the "submodularity counterexample" regarding the 3-Terminal Cut Problem [14]; and also using ideas from the previous (less general) proof of **NP**-completeness for simple graphs [1]. However, most of my techniques and details are different from those used by Karzanov in [57].

Bibliography

[1] A. V. Karzanov. Minimum 0-Extensions of Graph Metrics. *European Journal of Combinatorics*, 19(1):71–101, 1998.

[2] V. Chepoi. A multifacility location problem on median spaces. *Discrete Applied Mathematics*, 64(1):1–29, 1996.

[3] Jakob Krarup and Cornelis Roos. On the Fermat point of a triangle. *Nieuw Archief voor Wiskunde*, 5/18:280–286, 2017.

[4] Thomas Simpson. The doctrine and application of fluxions. *Before the Textbook*, 1737.

[5] Antoon Kolen. *Tree network and planar rectilinear location theory*. CWI Tracts. CWI, 1986.

[6] Howard Karloff, Subhash Khot, Aranyak Mehta, and Yuval Rabani. On Earthmover Distance, Metric Labeling, and 0-Extension. *SIAM J. Comput.*, 39:371–387, 2009.

[7] J. Kleinberg and E. Tardos. Approximation algorithms for classification problems with pairwise relationships: metric labeling and Markov random fields. In *40th Annual Symposium on Foundations of Computer Science (Cat. No.99CB37039)*, pages 14–23, New York City, NY, USA, 1999. IEEE Comput. Soc.

[8] S. Geman and D. Geman. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6), 1984.

[9] Libor Barto, Andrei Krokhin, and Ross Willard. Polymorphisms, and How to Use Them. In Andrei Krokhin and Stanislav Zivny, editors, *The Constraint Satisfaction Problem: Complexity and Approximability*, volume 7 of *Dagstuhl Follow-Ups*, pages 1–44. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2017.

[10] Andrei Krokhin and Stanislav Živný. The Complexity of Valued CSPs. In *The Constraint Satisfaction Problem: Complexity and Approximability*, volume 7 of *Dagstuhl Follow-Ups*, pages 233–266. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2017.

[11] L. R. Ford and D. R. Fulkerson. Maximal Flow Through a Network. *Canadian Journal of Mathematics*, 8:399–404, 1956. Publisher: Cambridge University Press.

[12] Jean-Claude Picard and H. Donald Ratliff. A Cut Approach to the Rectilinear Distance Facility Location Problem. *Operations Research*, 26(3):422–433, 1978. Publisher: INFORMS.

[13] Antoon Kolen. Equivalence between the Direct Search Approach and the Cut Approach to the Rectilinear Distance Location Problem. *Operations Research*, 29(3):616–620, 1981. Publisher: INFORMS.

[14] E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. The Complexity of Multiterminal Cuts. *SIAM Journal on Computing*, 23(4):864–894, 1994.

[15] R. Ram Kumar and B. Kannan. Median Sets and Median Number of a Graph. *ISRN Discrete Mathematics*, 2012.

[16] H. Hirai. Discrete Convexity and Polynomial Solvability in Minimum 0-Extension Problems. *Mathematical Programming*, Series A(155):1–55, 2016.

[17] V. Kolmogorov. Personal communication, 2019.

[18] Richard E. Ladner. On the Structure of Polynomial Time Reducibility. *Journal of the ACM*, 22(1):155–171, 1975.

[19] Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the tenth annual ACM symposium on Theory of computing*, STOC '78, pages 216–226, New York, NY, USA, 1978. Association for Computing Machinery.

[20] Pavol Hell and Jaroslav Nešetřil. On the complexity of H-coloring. *Journal of Combinatorial Theory, Series B*, 48(1):92–110, 1990.

[21] Johan Thapper and Stanislav Živný. The complexity of finite-valued CSPs. In *Proceedings of the forty-fifth annual ACM symposium on Theory of Computing*, STOC '13, pages 695–704, New York, NY, USA, 2013. Association for Computing Machinery.

[22] Dmitriy Zhuk. A Proof of the CSP Dichotomy Conjecture. *Journal of the ACM*, 67(5):30:1–30:78, 2020.

[23] Andrei Bulatov. A Dichotomy Theorem for Nonuniform CSPs. In *IEEE 58th Annual Symposium on Foundations of Computer Science*, pages 319–330, 2017.

[24] Tomás Feder and Moshe Y. Vardi. Monotone monadic SNP and constraint satisfaction. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of Computing*, STOC '93, pages 612–622, New York, NY, USA, 1993. Association for Computing Machinery.

[25] Vladimir Kolmogorov, Andrei Krokhin, and Michal Rolinek. The Complexity of General-Valued CSPs. *arXiv:1502.07327 [cs]*, 2017.

[26] Johan De Kleer. A comparison of ATMS and CSP techniques. In *Proceedings of the 11th international joint conference on Artificial intelligence - Volume 1*, IJCAI'89, pages 290–296, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

[27] Roman Barták, Miguel Salido, and Francesca Rossi. New trends in constraint satisfaction, planning, and scheduling: A survey. *Knowledge Eng. Review*, 25:249–279, 2010.

[28] Ugo Montanari. Networks of constraints: Fundamental properties and applications to picture processing. *Information Sciences*, 7:95–132, 1974.

[29] David Waltz. Understanding Line Drawings of Scenes with Shadows. In *The Psychology of Computer Vision*. McGraw-Hill, 1975.

[30] Eugene C. Freuder and Alan K. Mackworth. Introduction to the special volume on constraint-based reasoning. *Artificial Intelligence*, 58(1):1–2, 1992.

[31] Vishal Soni, Satinder Singh, and Michael P. Wellman. Constraint satisfaction algorithms for graphical games. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, AAMAS '07, pages 1–8, New York, NY, USA, 2007. Association for Computing Machinery.

[32] S. Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, Reading, Mass, 1995.

[33] Ashok K. Chandra and David Harel. Horn clause queries and generalizations. *The Journal of Logic Programming*, 2(1):1–15, 1985.

[34] ISO. SQL-92. Standard ISO/IEC 9075:1992, International Electrotechnical Commision, Geneva, Switzerland, 1992.

[35] David A. Cohen, Martin C. Cooper, Paidi Creed, Peter G. Jeavons, and Stanislav Zivny. An Algebraic Theory of Complexity for Discrete Optimization. *SIAM Journal on Computing*, 42(5):1915–1939, 2013.

[36] Libor Barto. The collapse of the bounded width hierarchy. *Journal of Logic and Computation*, 26(3):923–943, 2016.

[37] Libor Barto and Michael Pinsker. Topology Is Irrelevant (In a Dichotomy Conjecture for Infinite Domain Constraint Satisfaction Problems). *SIAM Journal on Computing*, 49(2):365–393, 2020.

[38] Peter G. Hinman. *Fundamentals of Mathematical Logic*. CRC Press, 2018.

[39] H.-J. Bandelt. Hereditary modular graphs. *Combinatorica*, 8(2):149–157, 1988.

[40] N. R. Howes. *Modern Analysis and Topology*. Universitext. Springer-Verlag, New York, 1st edition, 1995.

[41] John E. Hopcroft and Jeffrey D. Ullman. *Formal Languages and Their Relation to Automata*. Addison-Wesley, Reading, Mass., 1st edition edition, 1969.

[42] William Cunningham. The optimal multiterminal cut problem. *Discrete Mathematics and Theoretical Compututer Science*, 5:105–120, 1991.

[43] G. B. Dantzig and D. R. Fulkerson. On the max-flow min-cut theorem of networks. *Linear Inequalities, Annals of Mathematics Studies*, 38(1):215–221, 1956.

[44] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computation*, pages 85–103. Springer, Boston, MA, 1972.

[45] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, 1976.

[46] Johan Thapper and Stanislav Živný. The Complexity of Finite-Valued CSPs. *Journal of the ACM*, 63(4):1–33, 2016.

[47] Anna Huber, Andrei Krokhin, and Robert Powell. Skew Bisubmodularity and Valued CSPs. *SIAM Journal on Computing*, 43, 2014.

[48] David Cohen, Martin Cooper, Peter Jeavons, and Andrei Krokhin. The complexity of soft constraint satisfaction. *Artificial Intelligence*, 170:983–1016, 2006.

[49] Vladimir Kolmogorov. The Power of Linear Programming for Finite-Valued CSPs: A Constructive Characterization, 2012.

[50] L.G. Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53–72, 1980.

[51] Victor Klee and George J. Minty. How good is the simplex algorithm? *Inequalities III, Proceedings Third Symposium*, pages 159–175, 1972.

[52] Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *Journal of the ACM*, 51(3):385–463, 2004.

[53] M. I. Shlezinger. Syntactic analysis of two-dimensional visual signals in the presence of noise. *Kibernetika*, 12:612–628, 1977.

[54] Ferdinand Georg Frobenius. *Über die Congruenz nach einem aus zwei endlichen Gruppen gebildeten Doppelmodul*. ETH-Bibliothek Zürich, 1887.

[55] William Burnside. *Theory of Groups of Finite Order*. Cambridge University Press, 1897.

[56] Felix Klein. *Vorlesungen über das Ikosaeder: und die Auflösung der Gleichungen vom fünften Grade*. Teubner, Leipzig, 1884.

[57] NOT USED AS A RESOURCE FOR OBTAINING MY RESULTS: Karzanov. Hard cases of the multifacility location problem. *Discrete Applied Mathematics*, 143:368–373, 2004.