No-Rainbow Problem and the Surjective Constraint Satisfaction Problem

Dmitriy Zhuk

April 30, 2021

Abstract

The Surjective Constraint Satisfaction Problem (SCSP) is the problem of deciding whether there exists a surjective assignment to a set of variables subject to some specified constraints, where a surjective assignment is an assignment containing all elements of the domain. In this paper we show that the most famous SCSP, called No-Rainbow Problem, is NP-Hard. Additionally, we disprove the conjecture saying that the SCSP over a constraint language Γ and the CSP over the same language with constants have the same computational complexity up to poly-time reductions. Our counter-example also shows that the complexity of the SCSP cannot be described in terms of polymorphisms of the constraint language.

1 Introduction

1.1 Constraint Satisfaction Problem

The *Constraint Satisfaction Problem (CSP)* is the problem of deciding whether there is an assignment to a set of variables subject to some specified constraints. Formally, it can be defined in the following way. Let A be a finite set, Γ be a set of relations on A, called a *constraint language*. Then *the Constraint Satisfaction Problem over the constraint language* Γ, denoted by $\mathrm{CSP}(\Gamma)$, is the following decision problem: given a formula of the form

$$R_1(z_{1,1},\dots,z_{1,n_1}) \wedge \dots \wedge R_s(z_{s,1},\dots,z_{s,n_s}),$$

where $R_1,\dots,R_s \in \Gamma$ and each $z_{i,j} \in \{x_1,\dots,x_n\}$; decide whether the formula is satisfiable. It is well known that many combinatorial problems can be expressed as $\mathrm{CSP}(\Gamma)$ for some constraint language Γ. Moreover, for some sets Γ the corresponding decision problem can be solved in polynomial time; while for others it is NP-complete. It was conjectured in 1998 that $\mathrm{CSP}(\Gamma)$ is either in P, or NP-complete [15]. In 2017 this conjecture was resolved independently by Andrei Bulatov [4, 5] and Dmitriy Zhuk [35, 36]. Moreover, the classification of the complexity for different constraint languages turned out to be very simple and was given in terms of polymorphisms. We say that a k-ary operation f is a *polymorphism* of an m-ary relation R if whenever $(x_1^1,\dots,x_1^m),\dots,(x_k^1,\dots,x_k^m)$ in R, then also $(f(x_1^1,\dots,x_k^1),\dots,f(x_1^m,\dots,x_k^m))$ in R. In this case we also say that f *preserves* R. An operation f is a *polymorphism* of Γ if it is a polymorphism of every relation in Γ. An operation w is called a *weak near-unanimity (WNU) operation* if it satisfies the following identities

$$w(y,x,\dots,x) = w(x,y,x,\dots,x) = \dots = w(x,\dots,x,y).$$

For instance a majority operation, conjunction, disjunction are WNU operations. Then the complexity of $\mathrm{CSP}(\Gamma)$ is described by the following theorem.

Theorem 1 ([4, 5, 35, 36])**.** *Suppose* Γ *is a finite constraint language on a finite set* A*. Then* $\mathrm{CSP}(\Gamma)$ *is solvable in polynomial time if there exists a WNU polymorphism of* Γ*;* $\mathrm{CSP}(\Gamma)$ *is NP-complete otherwise.*

1.2 Surjective Constraint Satisfaction Problem

In this paper we consider a variant of the CSP with an additional global constraint saying that a solution should be surjective. *The Surjective Constraint Satisfaction Problem over a constraint language* Γ *on a domain* A, denoted by $\mathrm{SCSP}(\Gamma)$, is the following decision problem: given a formula of the form

$$R_1(z_{1,1}, \ldots, z_{1,n_1}) \wedge \cdots \wedge R_s(z_{s,1}, \ldots, z_{s,n_s}),$$

where $R_1, \ldots, R_s \in \Gamma$ and each $z_{i,j} \in \{x_1, \ldots, x_n\}$; decide whether there exists a solution such that $\{x_1, \ldots, x_n\} = A$. Here we assume that the set of variables $\{x_1, \ldots, x_n\}$ is fixed and do not require all the variables to appear in some constraint. Unlike the complexity of the CSP, the complexity of $\mathrm{SCSP}(\Gamma)$ remains unknown even for very simple constraint languages Γ.

1.3 Surjective Graph Homomorphism Problem

Probably, the most natural examples of the Surjective CSP are defined as the graph homomorphism problem. Assume that a graph H is fixed, *the Surjective Graph Homomorphism Problem*, denoted by $\mathrm{SurjHom}(H)$, is the problem of deciding for a given graph G whether there exists a surjective homomorphism from G to H. This problem is also known as *the Vertex-Compaction Problem* [33] or *the Surjective* H*-Colouring Problem* [18]. For usual homomorphism this problem is known as *the* H*-colouring problem*. Note that $\mathrm{SurjHom}(H)$ is equivalent to $\mathrm{SCSP}(\{H\})$, where the graph H is viewed as a binary relation.

An interesting fact about the complexity of $\mathrm{SurjHom}(H)$ is that it remained unknown for many years even for very simple graphs H. Recall that the situation was different with the complexity of the CSP and the H-colouring problem: even though the general classification remained open for many years, nobody could show a simple constraint language or a graph with unknown complexity. We will discuss two popular examples of graphs with unknown complexity of the SurjHom in more detail.

First example is the complexity of SurjHom for the reflexive 4-cycle (undirected having a loop at each vertex), which was formulated as an open question in [12] and [16], later it was a principal open question in [21, 20], and the central question discussed in [9] and [13]. This problem is known as *the disconnected cut problem*, since it is equivalent to finding a cutset (a set whose removal results in a disconnected graph) such that the cutset is disconnected itself, and it has attracted a lot of interest from the graph theory community. This problem was known to be tractable for many graph classes [9, 13, 16, 21], but in 2011 it was finally proved that the disconnected cut problem is NP-complete [25].

Second long-standing problem is the complexity of SurjHom for the non-reflexive 6-cycle (undirected without loops), that has been of interest since 1999 [27] when Narayan Vikas proved NP-completeness of a similar problem for the 6-cycle, called *the compaction problem*. The difference between the compaction problem and the vertex-compaction problem is that for compaction we require the homomorphism to be edge-surjective. In 2017 it was finally proved that SurjHom for the 6-cycle is also NP-complete [32]. It is worth mentioning that the compaction and the vertex-compaction problems have the same complexity for all known graphs, and it was conjectured by Winkler, Vikas, and others that these problems are polynomially equivalent. For more information about the relationship between the compaction problem and the vertex-compaction problem (and also the retraction problem) see [33].

Many other results on the complexity of SurjHom(H) can be found in [23, 18, 17, 30, 29, 28, 31, 19, 32, 25]. For instance, the complexity of SurjHom is known for all graphs on at most four vertices [18]. But as far as we know the complexity remains unknown for graphs of size 5 and even for cycles, which makes this problem very intriguing because of the simplicity of the formulation.

1.4 The Complexity of SCSP

Let us discuss what we know about the complexity of the SCSP and the CSP. The complexity of CSP(Γ) is known for any constraint language Γ [4, 5, 35, 36]. The complexity of SCSP(Γ) is widely open. Since we can always add dummy variables we never use and these dummy variables could give surjectivity, CSP(Γ) can be trivially reduced to SCSP(Γ). Also, it is sufficient to consider the reflexive 4-cycle H to get a constraint language $\Gamma = \{H\}$ such that SCSP(Γ) is NP-complete but CSP(Γ) is trivial and solvable in polynomial time. Thus, sometimes SCSP(Γ) is harder than CSP(Γ).

Another important observation is that the Surjective CSP can be reduced to the CSP over the same language with constants (see [2, Section 2]). Adding constants is equivalent to adding unary singleton relations $\{\{a\} \mid a \in A\}$ to the constraint language, because using these relations we can write $x_i = a$. Let us show how this reduction works.

Lemma 2 ([2])**.** *There exists a polynomial-time Turing reduction from* SCSP(Γ) *to* CSP($\Gamma \cup \{\{a\} \mid a \in A\}$)

Proof. Let $A = \{a_1, \ldots, a_n\}$. Suppose we have an instance $\mathcal{I}$ of SCSP(Γ). First, we guess n variables $x_{i_1}, \ldots, x_{i_n}$ such that these variables give all elements of A in a solution, that is $x_{i_j} = a_j$ for every j. Then we consider the following instance of CSP($\Gamma \cup \{\{a\} \mid a \in A\}$)

$$\bigwedge_{j=1}^{n} (x_{i_j} = a_j) \wedge \mathcal{I}.$$

If it has a solution, then $\mathcal{I}$ has a surjective solution.

Since we have only polynomially many choices for the variables, this gives us a polynomial-time Turing reduction to CSP($\Gamma \cup \{\{a\} \mid a \in A\}$). ☐

As a corollary we derive that CSP($\Gamma \cup \{\{a\} \mid a \in A\}$) and SCSP($\Gamma \cup \{\{a\} \mid a \in A\}$) have the same complexity.

Recall that the classification of the complexity of the CSP for different constraint languages was given in terms of polymorphisms (see Theorem 1). It is natural to assume that the complexity of the SCSP can be described in a similar way. In [7] Hubie Chen asked whether Lemma 2 holds in both directions, which would imply the characterization of the complexity of the SCSP in terms of polymorphisms. Thus, he conjectured the following characterization of the complexity of SCSP(Γ).

Conjecture 1. *For any constraint language* Γ *the problems* CSP($\Gamma \cup \{\{a\} \mid a \in A\}$) *and* SCSP($\Gamma$) *are polynomially equivalent.*

An operation f is called *idempotent* if it satisfies $f(x, x, \ldots, x) = x$. Then, using Theorem 1 Conjecture 1 can be formulated in the following form.

Conjecture 2. *For any constraint language* Γ *the problem* SCSP(Γ) *is solvable in polynomial time if there exists an idempotent WNU polymorphism of* Γ*;* SCSP(Γ) *is NP-complete otherwise.*

As it was proved in [10] (see also [11]), Conjectures 1 and 2 hold for a 2-element domain. Also, all known results on the complexity of the SCSP agree with these conjectures (see [2]).

Additionally, in [6] Hubie Chen confirmed that polymorphisms of Γ can be used to describe the complexity of SCSP(Γ). For instance, he proved NP-hardness of SCSP(Γ) for Γ admitting only essentially unary polymorphisms. For more results on the complexity of the SCSP see the survery [2].

1.5 No-Rainbow Problem

In this paper we consider the most famous constraint language with unknown complexity of the SCSP. Let $N = \{(a,b,c) \in \{0,1,2\}^3 \mid \{a,b,c\} \neq \{0,1,2\}\}$. The problem SCSP($\{N\}$) is called *No-Rainbow Problem* because if we interpret $0, 1$, and 2 as colors then the relation N forbids "rainbow". Apparently, No-Rainbow Problem was first formulated under this name in [1]. Nevertheless, this problem can be viewed as the problem of coloring of a co-hypergraph in exactly 3 colors, and such problems were studied in many papers earlier. The notion of coloring of a mixed hypergraph appeared in [34]. Sufficient and necessary conditions for the existence of a k-coloring of a co-hypergraph were studied in [14]. The complexity of the problem of such coloring was studied in [22].

Despite the simplicity of the formulation, the complexity of SCSP($\{N\}$) was an open question for many years and was formulated many times as an important open problem [2, 7]. For example, Hubie Chen presented this problem as one of three concrete open questions at several conferences (see [7]) emphasizing that this problem is an important step toward a full classification of the complexity.

Note that the CSP over N is trivial because any instance has a trivial solution where all variables are equal. But if we add all constant relations then the problem becomes NP-hard, that is CSP($\{N, \{0\}, \{1\}, \{2\}\}$) is NP-complete [3].

As we mentioned earlier SCSP($\{N, \{0\}, \{1\}, \{2\}\}$) and CSP($\{N, \{0\}, \{1\}, \{2\}\}$) have the same complexity. It is interesting to compare the complexity of the CSP and the SCSP if we add some constant relations but probably not all of them.

Constraint Language	CSP	SCSP
$\{N\}$	P (Trivial)	???
$\{N, \{0\}\}$	P (Trivial)	???
$\{N, \{0\}, \{1\}\}$	P (Trivial)	NP-comp. [2]
$\{N, \{0\}, \{1\}, \{2\}\}$	NP-comp.	NP-comp.

For instance, as it was shown in [2] if we add just two constant relations then the CSP is still trivial but the SCSP is already NP-complete. It can be shown that the only problems in the table with unknown complexity, that is SCSP($\{N\}$) and SCSP($\{N, \{0\}\}$), are polynomially equivalent. In fact, to reduce SCSP($\{N, \{0\}\}$) to SCSP($\{N\}$) we just replace all variables x_i that appear in a constraint $x_i = 0$ by a unique fresh variable z and remove all constraints of the form $x_i = 0$.

2 Main Results

In this section we formulate two main results of this paper.

First, we proved that the No-Rainbow Problem is NP-hard, and therefore it is NP-complete.

Theorem 3. SCSP($\{N\}$) *is NP-complete.*

Recently, Hubie Chen published a paper explaining the algebraic framework for proving such hardness results [8]. He shows that this approach can also be used to prove NP-hardness for the reflexive 4-cycle (the disconnected cut problem) and for all NP-hard constraint languages on a 2-element domain.

Theorem 3 agrees with Conjectures 1 and 2. Unfortunately, we found a counter-example to these conjectures, which is the second main result of this paper. Let us define 5-ary and 3-ary relations on the set $\{0, 1, 2\}$ (here columns are tuples):

$$R = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 2 & 1 & 1 & 1 \\ 2 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 2 & 1 & 1 & 1 \\ 0 & 1 & 2 & 0 & 1 & 2 \end{pmatrix}, R' = \begin{pmatrix} 2 & 2 & 1 \\ 2 & 1 & 1 \\ 1 & 2 & 1 \end{pmatrix}.$$

Note that R' is the projection of R onto coordinates 2, 3 and 4. Hence, every polymorphism of R is also a polymorphism of R'.

Theorem 4. *We have*

(1) $\mathrm{CSP}(\{R, \{0\}, \{1\}, \{2\}\})$ *is NP-hard;*

(2) $\mathrm{SCSP}(\{R\})$ *is solvable in polynomial time;*

(3) $\mathrm{SCSP}(\{R'\})$ *is NP-Hard.*

Thus, items (1) and (2) of Theorem 4 disprove Conjectures 1 and 2. Also, comparing (2) and (3) we derive that the complexity of $\mathrm{SCSP}(\Gamma)$ cannot be described in terms of polymorphisms. Hence, we need a brand new idea to describe the complexity of the SCSP for all constraint languages.

Nevertheless, our counter-example is of arity 5 and it is important for the proof. Thus, probably for binary relations Conjectures 1 and 2 still hold. Another way to break our counter-example is to add all projections of relations from the constraint language to the constraint language. For instance, we could achieve this situation by considering only Γ that are defined as the set of all invariants of some algebra, which is a very natural way to define a constraint language for the CSP. Thus, it is interesting to answer the following open questions.

Question 1. *Do Conjectures 1 and 2 hold for* $\Gamma = \{H\}$, *where* H *is a binary relation?*

Question 2. *Do Conjectures 1 and 2 hold for* Γ *consisting of binary relations?*

Question 3. *Do Conjectures 1 and 2 hold for* Γ *that is closed under projections (adding existential quantifiers)?*

For an algebra $\mathbf{A} = (A; f_1, f_2, \dots)$ by $\mathrm{Inv}(\mathbf{A})$ we denote the set of all relations preserved by every operation f_i of an algebra.

Question 4. *Do Conjectures 1 and 2 hold for* Γ *such that* $\Gamma = \mathrm{Inv}(\mathbf{A})$ *for some algebra* $\mathbf{A}$*?*

The paper is organized as follows. In Section 3 and Section 4 we give two different proofs of Theorem 3. Both proofs are based on the reduction from an NP-hard CSP problem, the first reduction is easier in terms of complexity (just three variables for every variable of the original instance) but the idea of the reduction is hidden there. We hope the second reduction is better for understanding but there we create nine variables for every variable of the original instance. In Section 5 we prove Theorem 4.

3 No-Rainbow problem is NP-Hard (the first proof)

Here we present our first proof of the fact that SCSP($\{N\}$) is NP-hard. By $[n]$ we denote the set $\{1, 2, \ldots, n\}$. Let $\Gamma_0 = \{x = y \vee z = 0, x = y \vee z = 1, x = 0, x = 1\}$ be a constraint language on $\{0, 1\}$. The problem CSP(Γ_0) is known to be NP-Hard [26]. Hence, to prove Theorem 3 it is sufficient to prove the following theorem.

Theorem 5. CSP(Γ_0) *can be polynomially reduced to* SCSP($\{N\}$).

Let us define the reduction. Let $\mathcal{I}$ be an instance of CSP(Γ_0). Let us build an instance $\mathcal{J}$ of SCSP($\{N, =\}$) such that $\mathcal{I}$ holds if and only if $\mathcal{J}$ has a surjective solution. Note that the problems SCSP($\{N, =\}$) and SCSP($\{N\}$) are equivalent because we can always propagate out the equalities.

The idea is to introduce constraints (see $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_{10}$ below) such that three concrete variables take all the values from the domain in any surjective solution. Below such variables are x, x', y'. Other variables still have some freedom, for instance $x_1, \ldots, x_n$ can be chosen freely from $\{x, x'\}$. These variables will be used to encode the variables $u_1, \ldots, u_n$ of the instance $\mathcal{I}$. By adding new constraints (see $\mathcal{C}_{11}, \mathcal{C}_{12}, \mathcal{C}_{13}, \mathcal{C}_{14}$ below) we can encode constraints of CSP(Γ_0) on the variables $x_1, \ldots, x_n$. A formal construction is below.

Construction. Let $u_1, \ldots, u_n$ be the variables of $\mathcal{I}$.

Choose variables $x, x', y', x_1, \ldots, x_n, y_1, \ldots, y_n, z_1, \ldots, z_n$. We define 14 sets of constraints:

- $\mathcal{C}_1 = \{N(x, x_i, y_i) \mid i \in [n]\}$,
- $\mathcal{C}_2 = \{N(x', z_i, y_i) \mid i \in [n]\}$,
- $\mathcal{C}_3 = \{N(y', z_i, x_i) \mid i \in [n]\}$,
- $\mathcal{C}_4 = \{N(t_1, t_2, t_3) \mid t_1, t_2, t_3 \in \{x, x', x_1, \ldots, x_n\}\}$,
- $\mathcal{C}_5 = \{N(t_1, t_2, t_3) \mid t_1, t_2, t_3 \in \{x, y', y_1, \ldots, y_n\}\}$,
- $\mathcal{C}_6 = \{N(t_1, t_2, t_3) \mid t_1, t_2, t_3 \in \{x', y', z_1, \ldots, z_n\}\}$,
- $\mathcal{C}_7 = \{N(x_i, y_j, z_i) \mid i, j \in [n]\}$,
- $\mathcal{C}_8 = \{N(x_i, y_j, z_j) \mid i, j \in [n]\}$,
- $\mathcal{C}_9 = \{N(x, x_i, z_i) \mid i \in [n]\}$,
- $\mathcal{C}_{10} = \{N(x, y_i, z_i) \mid i \in [n]\}$,
- $\mathcal{C}_{11} = \{N(x_i, x_j, y_k) \mid (u_i = u_j \vee u_k = 0) \in \mathcal{I}\}$,
- $\mathcal{C}_{12} = \{N(y_i, y_j, x_k) \mid (u_i = u_j \vee u_k = 1) \in \mathcal{I}\}$,
- $\mathcal{C}_{13} = \{x = x_i \mid (u_i = 1) \in \mathcal{I}\}$,
- $\mathcal{C}_{14} = \{x = y_i \mid (u_i = 0) \in \mathcal{I}\}$.

Lemma 6. *$\mathcal{I}$ has a solution if and only if $\mathcal{J}$ has a surjective solution, where $\mathcal{J} = \mathcal{C}_1 \cup \cdots \cup \mathcal{C}_{14}$.*

Proof. Let us show both implications.

$\mathcal{I} \in$ CSP(Γ_0) $\Rightarrow \mathcal{J} \in$ SCSP($\{N, =\}$).

Let $(b_1, \ldots, b_n)$ be a solution of $\mathcal{I}$. Put $x = 1$, $x' = 0$, $y' = 2$, $x_i = b_i$, $y_i = b_i + 1$, $z_i = 2 \cdot b_i$ for every $i \in [n]$. Let us check that all the constraints are satisfied.

- $\mathcal{C}_1$ holds because $|\{1, b_i, b_i+1\}| < 3$ for any $b_i \in \{0,1\}$.
- $\mathcal{C}_2$ holds because $|\{0, 2\cdot b_i, b_i+1\}| < 3$ for any $b_i \in \{0,1\}$.
- $\mathcal{C}_3$ holds because $|\{2, 2\cdot b_i, b_i\}| < 3$ for any $b_i \in \{0,1\}$.
- $\mathcal{C}_4$ holds because $x, x', x_1, \ldots, x_n$ are from the set $\{0,1\}$.
- $\mathcal{C}_5$ holds because $x, y', y_1, \ldots, y_n$ are from the set $\{1,2\}$.
- $\mathcal{C}_6$ holds because $x', y', z_1, \ldots, z_n$ are from the set $\{0,2\}$.
- $\mathcal{C}_7$ holds because $|\{b_i, b_j+1, 2\cdot b_i\}| < 3$ for any $b_i, b_j \in \{0,1\}$.
- $\mathcal{C}_8$ holds because $|\{b_i, b_j+1, 2\cdot b_j\}| < 3$ for any $b_i, b_j \in \{0,1\}$.
- $\mathcal{C}_9$ holds because $|\{1, b_i, 2\cdot b_i\}| < 3$ for any $b_i \in \{0,1\}$.
- $\mathcal{C}_{10}$ holds because $|\{1, b_i+1, 2\cdot b_i\}| < 3$ for any $b_i \in \{0,1\}$.
- $\mathcal{C}_{11}$ holds because each constraint is equivalent to $|\{b_i, b_j, 1+b_k\}| < 3$ and equivalent to $(b_i = b_j) \vee b_k = 0$.
- $\mathcal{C}_{12}$ holds because each constraint is equivalent to $|\{b_i+1, b_j+1, b_k\}| < 3$ and equivalent to $(b_i = b_j) \vee b_k = 1$.
- $\mathcal{C}_{13}$ holds because $x_i = x = 1$ whenever $b_i = 1$.
- $\mathcal{C}_{14}$ holds because $y_i = x = 1$ whenever $b_i = 0$.

$\mathcal{J} \in \mathrm{SCSP}(\{N, =\}) \Rightarrow \mathcal{I} \in \mathrm{CSP}(\Gamma_0)$.

Choose a surjective solution of $\mathcal{J}$. By $\mathcal{C}_4$ we can choose $a, b \in \{0,1,2\}$ such that $x = a$ and $\{x, x', x_1, \ldots, x_n\} \subseteq \{a, b\}$. Since the solution is surjective, there should be an element in the solution equal to $c \in \{0,1,2\} \setminus \{a,b\}$. Consider 5 cases.

Case 1. Assume that $y' = c$ and $x' = b$. Note that by the definition of N for any permutation σ on $\{0,1,2\}$ σ applied to a solution of an instance of $\mathrm{SCSP}(\{N\})$ gives a solution of the instance. Therefore, without loss of generality we may assime that $x = 1, x' = 0, y' = 2$ in our solution. By $\mathcal{C}_4$, $\mathcal{C}_5$, $\mathcal{C}_6$ we know that

$$\{x, x', x_1, \ldots, x_n\} = \{0,1\},$$
$$\{x, y', y_1, \ldots, y_n\} = \{1,2\},$$
$$\{x', y', z_1, \ldots, z_n\} = \{0,2\}.$$

Let us show that $y_i = x_i + 1$. If $x_i = 0$ and $y_i = 2$, then we get a contradiction with $\mathcal{C}_1$.

If $x_i = y_i = 1$, then by $\mathcal{C}_2$ we have $z_i \neq 2$, by $\mathcal{C}_3$ we have $z_i \neq 0$, which implies $z_i = 1$ and contradicts $\mathcal{C}_6$.

Let us show that $(u_1, \ldots, u_n) = (x_1, \ldots, x_n)$ is a solution of $\mathcal{I}$.

$\mathcal{C}_{11}$ guarantees that all the constraints of the form $(u_i = u_j \vee u_k = 0)$ hold, $\mathcal{C}_{12}$ guarantees that all the constraints of the form $(u_i = u_j \vee u_k = 1)$ hold, $\mathcal{C}_{13}$ guarantees that all the constraints of the form $u_i = 1$ hold, $\mathcal{C}_{14}$ guarantees that all the constraints of the form $u_i = 0$ hold. Thus, we proved that it is a solution.

Case 2. Assume that $y' = c$ and $x' = a$. By $\mathcal{C}_5$ we have $\{x, y', y_1, \ldots, y_n\} = \{a, c\}$. By $\mathcal{C}_6$ we have $\{x', y', z_1, \ldots, z_n\} = \{a, c\}$. Since the solution is surjective, there should be i such that $x_i = b$. By $\mathcal{C}_9$ we have $z_i \neq c$, by $\mathcal{C}_3$ we have $z_i \neq a$. Contradiction.

Case 3. Assume that $y' \neq c$ and $y_i = z_i = c$ for some i. By $\mathcal{C}_5$ we have $\{x, y', y_1, \dots, y_n\} = \{a, c\}$, hence $y' = a$. By $\mathcal{C}_6$ we have $\{x', y', z_1, \dots, z_n\} = \{a, c\}$, hence $x' = a$. Since the solution is surjective, there should be j such that $x_j = b$. By $\mathcal{C}_9$, we have $z_j = a$. By $\mathcal{C}_7$ we have $y_\ell = a$ for every ℓ. Contradiction.

Case 4. Assume that $y' \neq c$ and $y_i = c, z_i \neq c$ for some i. By $\mathcal{C}_5$ we have $\{x, y', y_1, \dots, y_n\} = \{a, c\}$, hence $y' = a$. By $\mathcal{C}_1$, we have $x_i = a$. By $\mathcal{C}_8$ we obtain $z_i \in \{a, c\}$ and since $z_i \neq c$ we have $z_i = a$. Then by $\mathcal{C}_8$ we obtain $x_\ell = a$ for every ℓ. If $z_j = b$ for some j, then we get a contradiction with $\mathcal{C}_7$ applied to (x_j, y_i, z_j). Since the solution is surjective, the only remaining option is $x' = b$, which contradicts $\mathcal{C}_2$.

Case 5. Assume that $\{x, y', y_1, \dots, y_n\} \subseteq \{a, b\}$. Since the solution is surjective, there exists i such that $z_i = c$. By $\mathcal{C}_9$ and $\mathcal{C}_{10}$ we have $x_i = y_i = a$. By $\mathcal{C}_7$ we have $y_\ell = a$ for every ℓ, by $\mathcal{C}_8$ we have $x_\ell = a$ for every ℓ. By $\mathcal{C}_2$ we have $x' \neq b$, by $\mathcal{C}_3$ we have $y' \neq b$. Therefore $x' = y' = a$ and by $\mathcal{C}_6$ we obtain $\{x', y', z_1, \dots, z_n\} = \{a, c\}$. Hence, none of the variables can be equal to b. Contradiction. □

4 No-Rainbow problem is NP-Hard (the second proof)

Here we present another proof of the fact that the No-Rainbow problem is NP-Hard. By NAE_3 we denote the ternary relation on $\{0, 1\}$ consisting of all tuples but $(0, 0, 0)$ and $(1, 1, 1)$. The idea is to encode an instance of CSP($\{NAE_3\}$) using the relation N. We assign a binary operation on $A = \{0, 1, 2\}$ to every variable of the instance of CSP($\{NAE_3\}$) and then encode each binary operation with nine variables on $\{0, 1, 2\}$. Our plan is to write conditions that guarantee that every binary operation depends essentially only on one variable. Later we interpret the dependence on the first variable as 0 and the dependence on the second variable as 1.

It is known from [26] that CSP($\{NAE_3\}$) is NP-hard. Hence, to prove Theorem 3 it is sufficient to reduce CSP($\{NAE_3\}$) to SCSP($\{N\}$). We start with a few auxiliary facts. For a relation ρ by Pol(ρ) we denote the set of all polymorphisms of ρ. We say that an operation f *depends essentially only on one variable* if $f(x_1, \dots, x_n) = g(x_i)$ for some i and a unary operation g, that is all variables but i-th are dummy. We will need the following properties of Pol(N).

Lemma 7 (Section 5.2.6 in [24]). *Suppose* $f \in \mathrm{Pol}(N)$ *then*

1. f *depends essentially only on one variable, or*
2. $|\mathrm{Im}(f)| < 3$*, that is,* f *never returns some value* $a \in A$.

We will encode each binary operation f on A via 9 variables on A, which we denote by

$$f(0,0), f(0,1), f(0,2), f(1,0), f(1,1), f(1,2),$$
$$f(2,0), f(2,1), f(2,2).$$

For a tuple α by $\alpha(i)$ we denote the i-th element of this tuple. If we just write the definition of the fact that f is a polymorphism of N we get the following lemma.

Lemma 8. $f \in \mathrm{Pol}(N)$ *if and only if*

$$\bigwedge_{\alpha,\beta \in N} N(f(\alpha(1), \beta(1)), f(\alpha(2), \beta(2)), f(\alpha(3), \beta(3))).$$

Thus, the condition $f \in \mathrm{Pol}(N)$ can be expressed as a conjunction of relations N.

Now we are ready to prove the main theorem of this subsection.

Theorem 9. CSP($\{NAE_3\}$) *can be polynomially reduced to* SCSP($\{N\}$).

Let us show how to encode CSP($\{NAE_3\}$) as SCSP($\{N\}$). Consider an instance $\mathcal{I}$ of CSP($\{NAE_3\}$). Let $x_1, \dots, x_n$ be the variables of $\mathcal{I}$ and T be the set of all triples (i, j, k) such that $NAE_3(x_i, x_j, x_k)$ appears in the instance.

As we mentioned earlier, we assign a binary operation f_i on A to each variable x_i, then we encode each operation with nine variables on A, which we denote by $f_i(0,0), f_i(0,1), f_i(0,2)$, $f_i(1,0)$, $f_i(1,1)$, $f_i(1,2)$, $f_i(2,0)$, $f_i(2,1)$, $f_i(2,2)$.

We want f_i to depend only on the first variable if $x_i = 0$ and only on the second variable if $x_i = 1$. By $\mathcal{I}'$ we denote the following instance:

$$\bigwedge_{i\in[n]} (f_i \in \mathrm{Pol}(N)) \bigwedge_{i\in[n],a\in A} (f_1(a,a) = f_i(a,a))$$

$$\bigwedge_{(i,j,k)\in T} N(f_i(0,1), f_j(1,2), f_k(2,0))$$

$$\bigwedge_{i,j\in[n],a,b,c\in A} N(f_i(a,b), f_i(c,b), f_j(a,c))$$

$$\bigwedge_{i,j\in[n],a,b,c\in A} N(f_i(b,a), f_i(b,c), f_j(a,c))$$

Note that by Lemma 8 the first conjunction can be written as a conjunction of relations N. Hence, this instance can be viewed as an instance of SCSP($\{N\}$) because we use only equalities and the relation N (the equalities can be propagated out).

Lemma 10. $\mathcal{I}$ *has a solution if and only if* $\mathcal{I}'$ *has a surjective solution.*

Proof. $\mathcal{I} \Rightarrow \mathcal{I}'$. Suppose we have a solution $(x_1, \dots, x_n)$ of $\mathcal{I}$. To get a surjective solution of $\mathcal{I}'$ it is sufficient to put $f_i(x, y) = x$ if $x_i = 0$ and $f_i(x, y) = y$ if $x_i = 1$. Since $f_i(0, 0) = 0$, $f_i(1, 1) = 1$, and $f_i(2, 2) = 2$ the solution is surjective. Let us prove that all the above conjunctions hold. The first conjunction holds since projections preserve N. The second conjunction is trivial. The third conjunction holds because $(x_1, \dots, x_n)$ is a solution of $\mathcal{I}$ and therefore the operations f_i, f_j, f_k cannot depend on the same variables. Let us check the fourth conjunction. If $x_i = 0$ then $f_i(a, b), f_i(c, b), f_j(a, c) \in \{a, c\}$, if $x_i = 1$ then $f_i(a, b), f_i(c, b), f_j(a, c) \in \{b, f_j(a, c)\}$. The remaining conjunction can be verified in the same way. Thus, we get a surjective solution of $\mathcal{I}'$.

$\mathcal{I}' \Rightarrow \mathcal{I}$. Suppose we have a surjective solution $(f_1, \dots, f_n)$ of $\mathcal{I}'$. Let $g(x) := f_1(x, x)$. By the second conjunction we have $f_i(x, x) = g(x)$ for every $i \in [n]$.

Assume that $|\mathrm{Im}(g)| = 3$. By the first conjunction and Lemma 7 each f_i depends essentially only on one variable. Assign $x_i = 0$ if f_i depends essentially on the first variable, and $x_i = 1$ if f_i depends essentially on the second variable. The third conjunction guarantees that f_i, f_j, and f_k cannot depend on the same coordinate for each $(i, j, k) \in T$. Indeed, if three operations f_i, f_j, f_k depend only on the first variables then we get

$$N(f_i(0,1), f_j(1,2), f_k(2,0)) = N(g(0), g(1), g(2)),$$

which does not hold because g is a bijection. Thus, for each $(i, j, k) \in T$ we have $NAE_3(x_i, x_j, x_k)$. Hence, we defined a solution of $\mathcal{I}$.

Assume that $|\mathrm{Im}(g)| = 2$. If $|\mathrm{Im}(f_i)| = 3$ for some i then by Lemma 7 f_i depends essentially only on one variable. This means that $\mathrm{Im}(f_i) = \mathrm{Im}(g)$ for every $i \in [n]$, which contradicts the surjectivity of the solution.

Assume that $\operatorname{Im}(g) = \{d\}$. Assume that $|\operatorname{Im}(f_k)| = 3$ for some $k \in [n]$. Then by Lemma 7 f_k depends essentially only on one variable and $\operatorname{Im}(f_k) = \operatorname{Im}(g) = \{d\}$, which gives us a contradiction. Thus, $|\operatorname{Im}(f_k)| < 3$ for every $k \in [n]$. Since the solution is surjective, the remaining two values from $A \setminus \{d\}$ should appear in the images of some functions f_i and f_j. To get a contradiction we use the last two conjunctions. The fifth conjunction is obtained by a permutation of variables in f_i from the fourth conjunction. Therefore, without loss of generality we consider f_i, f_j and $a, b, c \in A$ such that $\{d, f_i(a,b), f_j(a,c)\} = A$. By the fourth conjunction we have $N(f_i(a,b), f_i(c,b), f_j(a,c))$, which together with $\operatorname{Im}(f_i) = \{f_i(a,b), d\}$ implies $f_i(c,b) = f_i(a,b)$. Then by the fifth conjunction we get $N(f_j(a,c), f_j(a,b), f_i(c,b))$ which together with $\operatorname{Im}(f_j) = \{f_j(a,c), d\}$ implies $f_j(a,b) = f_j(a,c)$. Then by the fourth conjunction we have $(f_i(a,b), f_i(b,b), f_j(a,b)) = (f_i(a,b), d, f_j(a,c)) \in N$, which contradicts our assumption. $\square$

5 Counter-example

Here we will prove Theorem 4 from Section 2.

Recall that $R = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 2 & 1 & 1 & 1 \\ 2 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 2 & 1 & 1 & 1 \\ 0 & 1 & 2 & 0 & 1 & 2 \end{pmatrix}$, $R' = \begin{pmatrix} 2 & 2 & 1 \\ 2 & 1 & 1 \\ 1 & 2 & 1 \end{pmatrix}$.

Lemma 11. $\mathrm{CSP}(\{R, \{0\}, \{1\}, \{2\}\})$ *is NP-hard.*

Proof. Note that R' is the projection of R onto the coordinates 2, 3, and 4. We can check that R' is not preserved by any idempotent WNU on $\{1,2\}$. By Theorem 1, $\mathrm{CSP}(\{R', \{1\}, \{2\}\})$ is NP-hard. Therefore, $\mathrm{CSP}(\{R, \{0\}, \{1\}, \{2\}\})$ is NP-hard. $\square$

Lemma 12. $\mathrm{SCSP}(\{R'\})$ *is NP-hard.*

Proof. $\mathrm{SCSP}(\{R'\})$ can be viewed as the SCSP on the two-element set $\{1,2\}$ (we just add a variable that never appears for 0 to be in a solution), which is known to be NP-Hard [10, 11]. $\square$

Lemma 13. $\mathrm{SCSP}(\{R\})$ *is solvable in polynomial time.*

Proof. The idea of the algorithm is very simple. To prevent our instance from having a trivial surjective solution we have to restrict the fifth variable of every appearance of R to a smaller domain. After we did this, we can replace R by a conjunction of easier relations. Thus, we show that our instance has a trivial surjective solution unless all appearances of the relation R could be replaced by easier relations.

Let $\sigma = \{(1,1),(2,2)\}$. We will prove a stronger claim that $\mathrm{SCSP}(\Gamma)$ is solvable in polynomial time for $\Gamma = \{R, \sigma, \{0\}, \{1\}, \{1,2\}\}$. Consider an instance $\mathcal{I}$ of $\mathrm{SCSP}(\Gamma)$.

First, we want to classify every occurrence of a variable in the instance. We say that an occurrence is of *the first type* if the projection of the constraint onto this variable is $\{0\}$, we say that an occurrence is of *the second type* if the projection of the constraint onto this variable is a subset of $\{1,2\}$. In all other cases we say that it is an occurrence of *the third type*. For example in $R(x_1, x_2, x_3, x_4, x_5)$ the variable x_1 is of the first type, the variables x_2, x_3, x_4 are of the second type and x_5 is of the third type. In σ both variables are of the second type. Note that the third type appears only in the relation R.

Second, we want all the occurrences of each variable to be of one type. We do the following:

- If a variable occurs in the first and the second types then we return "No solutions".
- If a variable occurs in the first and third types then it should appear at the last position of R. Then we replace the relation R by σ, $\{0\}$ and $\{1\}$ using the following equation

$$R(x_1, x_2, x_3, x_4, x_5) \wedge (x_5 = 0) = \\ (x_1 = 0) \wedge \sigma(x_2, x_3) \wedge (x_4 = 1) \wedge (x_5 = 0)$$

- If a variable occurs in the second and third types then it should appear at the last position of R. Then we replace the relation R by σ, $\{0\}$ and $\{1,2\}$ using the following equation

$$R(x_1, x_2, x_3, x_4, x_5) \wedge (x_5 \in \{1,2\}) = \\ (x_1 = 0) \wedge \sigma(x_2, x_4) \wedge (x_3 = 1) \wedge (x_5 \in \{1,2\})$$

Finally we get an instance with the same solution set such that all the occurrences of each variable have the same type. Here we may have two cases:

Case 1. The instance $\mathcal{I}$ does not contain R at all. Such an instance is in fact trivial because it contains only equality relation on $\{1,2\}$ and unary relations. Hence, we can check whether it has a surjective solution in polynomial time. To avoid a formal explanation of how to do this, we can reduce such SCSP to CSP. Since CSP($\{\sigma, \{0\}, \{1\}, \{2\}, \{1,2\}\}$) can be solved in polynomial time [3], by Lemma 2 SCSP($\{\sigma, \{0\}, \{1\}, \{1,2\}\}$) is also solvable in polynomial time. We use this algorithm for our instance.

Case 2. The instance $\mathcal{I}$ contains R, which means that the instance has occurrences of variables of all three types (since all types appear in R). To get a surjective solution it is sufficient to send the variables of the first type to 0, the variables of the second type to 1, and the variables of the third type to 2. Since R holds on $(0,1,1,1,2)$ and σ holds on $(1,1)$, this is in fact a solution. □

References

[1] Manuel Bodirsky. *Constraint Satisfaction with Infinite Domains*. PhD thesis, Humboldt-Universität zu Berlin, 2004.

[2] Manuel Bodirsky, Jan Kára, and Barnaby Martin. The complexity of surjective homomorphism problems - a survey. *Discrete Applied Mathematics*, 160(12):1680–1690, 2012.

[3] Andrei A. Bulatov. A dichotomy theorem for constraint satisfaction problems on a 3-element set. *J. ACM*, 53(1):66–120, January 2006.

[4] Andrei A. Bulatov. A dichotomy theorem for nonuniform CSPs. *CoRR*, abs/1703.03021, 2017.

[5] Andrei A Bulatov. A dichotomy theorem for nonuniform CSPs. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 319–330. IEEE, 2017.

[6] Hubie Chen. An algebraic hardness criterion for surjective constraint satisfaction. *Algebra universalis*, 72(4):393–401, 2014.

[7] Hubie Chen. Three concrete complexity questions about constraints, in search of theories. AAA98.Arbeitstagung Allgemeine Algebra - 98-th Workshop on General Algebra. Dresden, Germany, 2019.

[8] Hubie Chen. Algebraic global gadgetry for surjective constraint satisfaction. *arXiv preprint arXiv:2005.11307*, 2020.

[9] Kathryn Cook, Simone Dantas, Elaine M Eschen, Luerbio Faria, Celina MH De Figueiredo, and Sulamita Klein. 2k2 vertex-set partition into nonempty parts. *Discrete mathematics*, 310(6-7):1259–1264, 2010.

[10] Nadia Creignou and J-J Hébrard. On generating all solutions of generalized satisfiability problems. *RAIRO-Theoretical Informatics and Applications*, 31(6):499–511, 1997.

[11] Nadia Creignou, Sanjeev Khanna, and Madhu Sudan. *Complexity classifications of boolean constraint satisfaction problems*, volume 7. SIAM, 2001.

[12] Simone Dantas, Celina MH de Figueiredo, Sylvain Gravier, and Sulamita Klein. Finding h-partitions efficiently. *RAIRO-Theoretical Informatics and Applications*, 39(1):133–144, 2005.

[13] Simone Dantas, FréDéRic Maffray, and Ana Silva. 2k2-partition of some classes of graphs. *Discrete Applied Mathematics*, 160(18):2662–2668, 2012.

[14] Kefeng Diao, Ping Zhao, and Huishan Zhou. About the upper chromatic number of a co-hypergraph. *Discrete Mathematics*, 220(1-3):67–73, 2000.

[15] Tomás Feder and Moshe Y. Vardi. The computational structure of monotone monadic snp and constraint satisfaction: A study through datalog and group theory. *SIAM J. Comput.*, 28(1):57–104, February 1999.

[16] Herbert Fleischner, Egbert Mujuni, Daniël Paulusma, and Stefan Szeider. Covering graphs with few complete bipartite subgraphs. *Theoretical Computer Science*, 410(21-23):2045–2053, 2009.

[17] Jacob Focke, Leslie Ann Goldberg, and Stanislav Zivny. The complexity of counting surjective homomorphisms and compactions. *SIAM J. Discrete Math.*, 33(2):1006–1043, 2019.

[18] Petr A. Golovach, Matthew Johnson, Barnaby Martin, Daniël Paulusma, and Anthony Stewart. Surjective h-colouring: New hardness results. *Computability*, 8(1):27–42, 2019.

[19] Petr A. Golovach, Daniël Paulusma, and Jian Song. Computing vertex-surjective homomorphisms to partially reflexive trees. *Theor. Comput. Sci.*, 457:86–100, 2012.

[20] Takehiro Ito, Marcin Kamiński, Daniël Paulusma, and Dimitrios M Thilikos. On disconnected cuts and separators. *Discrete applied mathematics*, 159(13):1345–1351, 2011.

[21] Takehiro Ito, Marcin Kamiński, Daniël Paulusma, and Dimitrios M Thilikos. Parameterizing cut sets in a graph by the number of their components. *Theoretical computer science*, 412(45):6340–6350, 2011.

[22] Daniel Král, Jan Kratochvíl, Andrzej Proskurowski, and Heinz-Jürgen Voss. Coloring mixed hypertrees. *Discrete Applied Mathematics*, 154(4):660–672, 2006.

[23] Benoît Larose, Barnaby Martin, and Daniël Paulusma. Surjective h-colouring over reflexive digraphs. *TOCT*, 11(1):3:1–3:21, 2019.

[24] D. Lau. *Function algebras on finite sets*. Springer, 2006.

[25] Barnaby Martin and Daniël Paulusma. The computational complexity of disconnected cut and 2_{k2}-partition. *J. Comb. Theory, Ser. B*, 111:17–37, 2015.

[26] 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. ACM.

[27] Narayan Vikas. Computational complexity of compaction to cycles. In *Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*, pages 977–978, 1999.

[28] Narayan Vikas. Computational complexity of compaction to reflexive cycles. *SIAM J. Comput.*, 32(1):253–280, 2002.

[29] Narayan Vikas. Computational complexity of compaction to irreflexive cycles. *J. Comput. Syst. Sci.*, 68(3):473–496, 2004.

[30] Narayan Vikas. A complete and equal computational complexity classification of compaction and retraction to all graphs with at most four vertices and some general results. *J. Comput. Syst. Sci.*, 71(4):406–439, 2005.

[31] Narayan Vikas. Algorithms for partition of some class of graphs under compaction and vertex-compaction. *Algorithmica*, 67(2):180–206, 2013.

[32] Narayan Vikas. Computational complexity of graph partition under vertex-compaction to an irreflexive hexagon. In Kim G. Larsen, Hans L. Bodlaender, and Jean-François Raskin, editors, *42nd International Symposium on Mathematical Foundations of Computer Science, MFCS 2017, August 21-25, 2017 - Aalborg, Denmark*, volume 83 of *LIPIcs*, pages 69:1–69:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

[33] Narayan Vikas. Computational complexity relationship between compaction, vertex-compaction, and retraction. *J. Discrete Algorithms*, 52-53:168–181, 2018.

[34] VI Voloshin. The mixed hypergraphs. *Computer Science Journal of Moldova*, 1(1):1, 1993.

[35] Dmitriy Zhuk. A proof of CSP dichotomy conjecture. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 331–342, 2017.

[36] Dmitriy Zhuk. A proof of the CSP dichotomy conjecture. *Journal of the ACM (JACM)*, 67(5):1–78, 2020.