On the Constant-Factor Approximability of Minimum Cost Constraint Satisfaction Problems

Ian DeHaan
University of Michigan, Ann Arbor, MI, USA

Neng Huang
University of Michigan, Ann Arbor, MI, USA

Euiwoong Lee
University of Michigan, Ann Arbor, MI, USA

Abstract

We study *minimum cost constraint satisfaction problems* (MinCostCSP) through the algebraic lens. We show that for any constraint language Γ which has the *dual discriminator* operation as a polymorphism, there exists a $|D|$-approximation algorithm for MinCostCSP(Γ) where D is the domain. Complementing our algorithmic result, we show that any constraint language Γ where MinCostCSP(Γ) admits a constant-factor approximation must have a *near-unanimity* (NU) polymorphism unless P = NP, extending a similar result by Dalmau et al. on MinCSPs. These results imply a dichotomy of constant-factor approximability for constraint languages that contain all permutation relations (a natural generalization for Boolean CSPs that allow variable negation): either MinCostCSP(Γ) has an NU polymorphism and is $|D|$-approximable, or it does not have any NU polymorphism and is NP-hard to approximate within any constant factor. Finally, we present a constraint language which has a majority polymorphism, but is nonetheless NP-hard to approximate within any constant factor assuming the Unique Games Conjecture, showing that the condition of having an NU polymorphism is in general not sufficient unless UGC fails.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis

Keywords and phrases Constraint satisfaction problems, approximation algorithms, polymorphisms

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2025.19

Category APPROX

Related Version *Full Version*: https://arxiv.org/abs/2507.08693

Funding *Euiwoong Lee*: Supported in part by NSF grant CCF-2236669.

Acknowledgements The authors would like to thank anonymous reviewers for their helpful comments.

1 Introduction

Constraint satisfaction problems (CSPs) are a central topic of study in theoretical computer science. In an instance of a CSP, we are given a finite set of variables taking values in a finite domain and a finite set of constraints on these variables, and our goal is to find an assignment to the variables so that all constraints are satisfied. CSPs provide a very expressive framework that encompasses many natural combinatorial problems, including satisfiability problems, graph coloring, and solving linear systems. CSPs in their full generality are NP-hard, and therefore it is natural to consider restrictions which lead to interesting tractable subclasses of CSPs. One very influential type of restrictions is to restrict the *constraint language*, that is, to restrict the set of relations that can be used as constraints. In this line of work, the ultimate goal is to obtain a dichotomy, if it exists, which characterizes the boundary between tractable and NP-hard constraint languages. The first such result, obtained by Schaffer [41], gave a complete classification of tractable Boolean CSPs. In their landmark paper [22], Feder and

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2025).
Editors: Alina Ene and Eshan Chattopadhyay; Article No. 19; pp. 19:1–19:24
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Vardi conjectured that a dichotomy exists for CSPs over general domains. This conjecture, known as the CSP Dichotomy Conjecture, led to a series of fruitful work culminating in the proof of the conjecture obtained independently by Bulatov [12] and by Zhuk [46].

Compared to the standard decision variant of CSPs, there are many natural optimization CSP variants whose complexity landscape is less understood. The most well-studied optimization variant is arguably the maximum constraint satisfaction problem (MaxCSP), where the objective is to find an assignment that maximizes the number of satisfied constraints. Interesting examples in this class include maximum cut and maximum satisfiability problems. For MaxCSPs, Raghavendra showed that the optimal approximation ratio[1] can be obtained by solving and rounding a generic *semidefinite programming* relaxation of the problem [40] (note that a constant approximation ratio for MaxCSP can be trivially obtained by uniform random assignment), assuming that the Unique Games Conjecture (UGC) [31] holds. However, the exact approximation ratio is not explicit in Raghavendra's result, and the ratios for many interesting MaxCSPs are still open (see e.g., [8, 9]). For MinCSPs, the objective is to minimize the number of unsatisfied constraints. MinCSP can be much harder than the corresponding MaxCSP in terms of the approximation ratio. In particular, it is at least as hard as the decision problem since an approximation algorithm has to satisfy every constraint when the instance is satisfiable. Valued CSPs generalize MinCSP by replacing 0-1 constraints with valued constraints, so that for any constraint different partial assignments can incur different costs. Thapper and Živný obtained a complexity dichotomy for the task of exact minimization for finite-valued CSPs [45], but the approximability question for this problem is still poorly understood. Ene et al. showed that under some mild technical assumption, there is a generic *linear programming* relaxation for finite-valued CSPs that is optimal for constant-factor approximation, unless UGC fails [21]. But unlike for MaxCSPs, it is unknown how to round this linear program. Dalmau et al. gave some algebraic conditions which indicate where the boundary of constant-factor approximability (or the lack thereof) for valued CSPs may lie [18], but a full characterization of constant-factor approximability is still unresolved.

In this work, we consider an optimization CSP variant called *minimum cost* CSP (MinCostCSP). In this variant, assigning any value to a variable comes with a cost, and the cost is a function of the variable-value pair. Our goal is to find a satisfying assignment that minimizes the total cost. MinCostCSP can be seen as a mixed variant between decision and optimization problems, in that we are still required to find a satisfying assignment. It can also be thought of as a special case of valued CSP, where we have some unary constraints representing the variable costs and all other constraints incur 0 cost if satisfied or infinite cost otherwise. MinCostCSP is a very natural CSP variant which avoids the full generality of valued CSPs, yet still includes many interesting problems, such as graph and hypergraph vertex cover, min-ones CSP [30] and minimum solution CSP [29].

We study the approximability, and in particular constant-factor approximability of MinCostCSP. Like many aforementioned results, our study is based on the universal-algebraic approach (see e.g., [34] for a survey on this approach applied to the exact optimization of valued CSPs), where we investigate the algebraic structure of any constraint language via its *polymorphisms*, which can be thought of certain high-dimensional symmetry that exists in the space of satisfying assignments. More specifically, we seek to algorithmically exploit the existence of desirable polymorphisms or show hardness results based on the lack thereof.

[1] For a maximization (resp. minimization) problem, an approximation algorithm achieves an approximation ratio of α, if on an input instance with global optimal Opt, it produces a solution whose objective value is at least $\alpha \cdot \mathrm{Opt}$ (resp. at most $\alpha \cdot \mathrm{Opt}$).

Our contribution

We obtain constant-factor approximability for constraint languages that have the *dual discriminator* operation as a polymorphism. These constraint languages can be thought of as generalization of 2-SAT to arbitrary finite domains. We give two algorithms for this class of problems, one using a greedy approach and the other based on a natural linear programming relaxation of the problem. Both algorithms crucially use the consistency notion called *(2,3)-minimality* [4].

▶ **Theorem 1.** *Let Γ be a constraint language over some domain D that has the dual discriminator operation as its polymorphism. Then* MinCostCSP(Γ) *can be $|D|$-approximated in polynomial time.*

Complementing our algorithmic result, we obtain the following hardness condition which says that constant-factor approximation is NP-hard for any constraint language which does not have a *near-unanimity* (NU) polymorphism.

▶ **Theorem 2.** *Let Γ be a constraint language such that* MinCostCSP(Γ) *has a constant-factor approximation, then Γ has a NU polymorphism, unless P = NP.*

Near unanimity operations are well-studied in universal algebra (see e.g., [1]), and they have also appeared in the study of CSPs [22, 18, 16]. In particular, Dalmau et al. showed that for valued CSPs the existence of NU polymorphisms is also a necessary condition for constant-factor approximability [18]. It can be verified that for MinCostCSP over the Boolean domain, the condition of having an NU polymorphism is not only necessary but also sufficient for constant-factor approximability (see Remark 40 for more discussion). However, as soon as the domain has at least 3 elements, there exist constraint languages which have NU polymorphisms yet does not admit constant-factor approximation, unless the UGC fails. We present such an example in Section 4.3.

Finally, as an application of our hardness and algorithmic results, we fully classify the constant-factor approximability of constraint languages that include all *permutation relations*, showing that the existence of an NU polymorphism is also a sufficient condition for this class. These languages can be thought of as a natural generalization of Boolean CSPs where we are allowed to apply constraints to negated variables. Our classification relies on the classification of *homogeneous algebras* by Marchenkov [36].

▶ **Theorem 3.** *Let Γ be a constraint language over some domain D that contains all permutation relations over D. Then* MinCostCSP(Γ) *can be $|D|$-approximated if Γ has an NU polymorphism, and is NP-hard to approximate within any constant factor otherwise.*

Related work

The approximability of MaxCSP, MinCSP, as well as MinCostCSP over the Boolean domain was fully classified by Khanna et al. [30]. In particular, for MinCostCSP they obtained the following complete classification: MinCostCSP(Γ) can be solved to optimality in polynomial time if Γ is "width-2 affine", that is, Γ can be expressed as a conjunction of linear equations over $\mathbb{F}_2$ where each equation has at most 2 variables; the problem can be approximated within a constant factor in polynomial time if Γ can be expressed as a 2CNF-formula, or if Γ is IHB-B+ (expressible as a CNF formula where each clause is of the form $x_1 \vee \cdots \vee x_k$, $\neg x_1 \vee x_2$, or $\neg x_1$ where $k \leq K$ for some K depending on Γ), or if Γ is IHB-B− (defined analogously to IHB-B+ with every literal replaced by its negation). Otherwise, MinCostCSP(Γ) is NP-hard to approximate within any constant factor. (See Remark 40 for a more detailed discussion in the context of our results.)

Over the general domain, a dichotomy for solving MinCostCSP(Γ) optimally was obtained by Takhanov [44]. Takhanov's characterization is based on local algebraic conditions satisfied by polymorphisms of Γ.

Kumar et al. showed that for a large class of covering and packing problems that can be expressed as MinCostCSP (they called it "Strict-CSP") over the general domain, a generic linear programming relaxation gives the optimal approximation ratio achievable in polynomial time, assuming the Unique Games Conjecture [35]. Their result generalizes the earlier UGC-based hardness results for vertex cover [33] and the k-uniform hypergraph vertex cover problems [2].

An important special case for MinCostCSP(Γ) where Γ consists of one single binary relation has been studied in the literature under the name "min-cost graph homomorphism" (in the case where the binary relation is symmetric) or "min-cost di-graph homomorphism" (in the general binary case). Dichotomy results for optimally solving these problems are known based on graph-theoretic properties [25, 27]. Hell et al. gave a similar dichotomy for constant-factor approximability for the min-cost graph homomorphism problem in the case where the graph (equivalently, the binary relation) is reflexive (every vertex has a self-loop) or irreflexive (no vertex has a self-loop) [26].[2]

Another CSP variant closely related to MinCostCSP is ListCSP which can be thought of as a special case of MinCostCSP where the costs take values in $\{0, \infty\}$. Bulatov obtained a complete classification for this problem (under the name "conservative CSP") based on the algebraic approach [10] (see also [3, 11]).

Organization of the paper

The rest of the paper is organized as follows. In Section 2, we formally define the problems and introduce some algebraic concepts that are needed throughout the paper. In Section 3, we present our main algorithmic results, proving Theorem 1. In Section 4, we prove some algebraic conditions sufficient for reductions between MinCostCSPs, and use them to prove Theorem 2. Finally, in Section 5, we use our results to give a dichotomy of constant-factor approximability for MinCostCSPs that contain all permutation relations, proving Theorem 3.

2 Preliminaries

2.1 CSP, ListCSP, and MinCostCSP

Let D be a finite set. A *relation* over D is a subset $R \subseteq D^k$ for some positive integer k, where D is called the *domain* of R and k is called the *arity* of R. A set of relations Γ over the same domain D is called a *constraint language*. Throughout this paper, any constraint language we consider will be assumed to contain finitely many relations whose common domain will be denoted by D. The elements in D will be referred to as *labels*.

▶ **Definition 4.** *Let Γ be a constraint language. An instance of CSP(Γ) is a tuple $I = (V, \mathcal{C})$, where V is a finite set of variables and $\mathcal{C}$ a finite set of constraints. Each constraint $C \in \mathcal{C}$ is of the form (R, S), where R is a relation in Γ with some arity k and $S \in V^k$ a k-tuple*

[2] We note that an ICALP'19 paper [39] claimed that the following dichotomy for constant-factor approximability holds over all (undirected) graphs: either a graph G has a conservative majority polymorphism and is constant-factor approximable, or it does not and is NP-hard to approximate within any constant factor. Our Theorem 45 contradicts this claim under the Unique Games Conjecture and $P \neq NP$.

of variables. An assignment *for I is a function $A : V \to D$. We say that A satisfies a constraint $C = (R, (x_1, \ldots, x_k))$ if $(A(x_1), \ldots, A(x_k)) \in R$, and we say that A is a satisfying assignment for $I = (V, \mathcal{C})$ if A satisfies every constraint in $\mathcal{C}$.*

One closely related variant of CSP is the following ListCSP problem. This problem has also been referred to as *conservative* CSP in the literature.

▶ **Definition 5.** *Let Γ be a constraint language. An instance of ListCSP(Γ) is a tuple $I = (V, \mathcal{C}, \{R_x\}_{x \in V})$, where V is a finite set of variables and $\mathcal{C}$ a finite set of constraints, as in the definition of CSP(Γ). In addition to V and $\mathcal{C}$, we are also given a subset $R_x \subseteq D$ for every variable $x \in V$. We say that A is a satisfying assignment for $I = (V, \mathcal{C}, \{R_x\}_{x \in V})$ if A satisfies every constraint in $\mathcal{C}$ and $A(x) \in R_x$ for every $x \in V$.*

ListCSP may be equivalently viewed as the ordinary CSP where all unary constraints are allowed in addition. In this paper, we will mainly focus on MinCostCSP, which further generalizes ListCSP.

▶ **Definition 6.** *Let Γ be a constraint language. An instance of MinCostCSP(Γ) is a tuple $I = (V, \mathcal{C}, \mathtt{cost})$, where V and $\mathcal{C}$ are the same as they are in the definition of CSP(Γ), and we are also given a function $\mathtt{cost} : V \times D \to \mathbb{R}^{\geq 0} \cup \{+\infty\}$. For any assignment A for I, the cost of A is defined to be $\mathtt{cost}_I(A) = \sum_{x \in V} \mathtt{cost}(x, A(x))$. The goal is find a satisfying assignment with the minimum cost.*

For any MinCostCSP instance I, the cost of any optimal solution is denoted by $\mathrm{Opt}(I) := \min_A \mathtt{cost}_I(A)$, where A ranges over all satisfying assignments for I. Although we allowed infinite costs in the definition of MinCostCSP, this is not essential in the context of approximability. In fact, we may simulate infinite cost by setting costs to be prohibitively high, say larger than $|V|$ times the maximum of any other finite cost, so that no approximation algorithm will pick the label. The inclusion of the infinite cost conveniently allows us to assume without loss of generality that Γ contains all unary relations.

▶ **Observation 7.** *Let Γ be a constraint language over some domain D and $\Gamma' = \Gamma \cup \{S \subseteq D \mid S \neq \varnothing\}$. Then MinCostCSP($\Gamma$) and MinCostCSP($\Gamma'$) are equivalent. Namely, any instance of MinCostCSP(Γ) can be solved as an instance of MinCostCSP(Γ') and vice versa.*

Proof. One direction is clear since $\Gamma \subseteq \Gamma'$. For the other direction, for any unary constraint $S(x)$ where $S \subseteq D$ (the constraint which says the label of x must be in S), we simply set $\mathtt{cost}(x, a) = \infty$ for any $a \notin S$. ◀

Observation 7 implies that MinCostCSP(Γ) contains ListCSP(Γ) as a special case. In particular, if ListCSP(Γ) is NP-hard, then for MinCostCSP(Γ) finding any satisfying assignment regardless of cost is NP-hard as well.

2.2 Polymorphisms

▶ **Definition 8.** *Let $f : D^k \to D$ be a k-ary operation on the domain D, and R some m-ary relation over the same domain. We say that f preserves R (or f is a* polymorphism *of R) if for every $(a_{1,1}, \ldots, a_{1,m}), \ldots, (a_{k,1}, \ldots, a_{k,m}) \in R$ we have $(b_1, \ldots, b_m) \in R$, where $b_i = f(a_{1,i}, \ldots, a_{k,i})$ for every $i \in [m]$. Given a constraint language Γ we say that f preserves Γ (or f is a* polymorphism *of Γ) if f preserves every $R \in \Gamma$. We use $\mathsf{Pol}(R)$ to denote the set of all polymorphisms of R, and (abusing notation slightly) $\mathsf{Pol}(\Gamma) = \bigcap_{R \in \Gamma} \mathsf{Pol}(R)$ to denote the set of all polymorphisms of Γ.*

It follows directly from the definition that for any $n \geq 1$, $i \in [n]$, the projection operation $\mathrm{proj}_i^n : D^n \to D, (x_1, \ldots, x_n) \mapsto x_i$ is a polymorphism. Also, any composition of polymorphisms is still a polymorphism. A set of operations satisfying these two properties is known as a *clone*.

▶ **Definition 9.** *A* clone *over D is a set of operations $\mathcal{F} \subseteq \bigcup_{n\geq 1}\{f : D^n \to D\}$ for which the following holds:*

- *$\mathcal{F}$ contains the projection operation proj_i^n for every $n \geq 1$, $i \in [n]$.*
- *For every m-ary $g \in \mathcal{F}$ and n-ary $f_1, \ldots, f_m \in \mathcal{F}$, the n-ary function g' defined by*

$$g' : (x_1, \ldots, x_n) \mapsto g(f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n))$$

is also in $\mathcal{F}$.

Clones have been studied extensively in the universal algebra community, and many complexity-theoretic classification results for CSPs depend on classification results for their corresponding family of polymorphism clones (see e.g. [6] for more on polymorphism clones).

We say that a function $f : D^k \to D$ is *conservative*, if $f(x_1, \ldots, x_k) \in \{x_1, \ldots, x_k\}$ for every $x_1, \ldots, x_k \in D$. It is immediate from the definition that if Γ is constraint language that contains all unary relations, then every $f \in \mathsf{Pol}(\Gamma)$ is conservative.

▶ **Definition 10.** *Let $f : D^k \to D$ be a k-ary operation for some $k \geq 3$. We say that f is a* near-unanimity (NU) *operation if for every $a, b \in D$,*

$$f(a, a, \ldots, a, b) = f(a, a, \ldots, b, a) = \cdots = f(b, a, \ldots, a, a) = a.$$

In other words, if all but one inputs are equal to some $a \in D$, then f outputs a. If $k = 3$, then f is also called a majority *operation.*

▶ **Example 11.** Let *the dual discriminator operation* $d : D^3 \to D$ be defined by

$$d(x_1, x_2, x_3) = \begin{cases} a & \text{if } |\{i \mid x_i = a\}| \geq 2, \\ x_1 & \text{otherwise.} \end{cases}$$

Then d is a majority operation. This operation will be featured heavily in our algorithmic results.

Constraint languages that are preserved by some NU operation enjoy the following nice property.

▶ **Definition 12.** *Let R be a n-ary relation. Let $S_{n,k} = \{(i_1, \ldots, i_k) \mid 1 \leq i_1 < \cdots < i_k \leq n\}$ be the set of k-tuples whose entries are in $[n]$ and in increasing order. For any $s = (i_1, \ldots, i_k) \in S_{n,k}$ and $x = (x_1, \ldots, x_n)$, let $\Pi_s x = (x_{i_1}, \ldots, x_{i_k})$ and $\Pi_s R = \{\Pi_s x \mid x \in R\}$. We say that R is k-decomposable if for every $x = (x_1, \ldots, x_n)$, we have*

$$x \in R \quad \Leftrightarrow \quad \forall s \in S_{n,k}, \Pi_s x \in \Pi_s R.$$

We say that Γ is k-decomposable if every $R \in \Gamma$ is k-decomposable.

▶ **Theorem 13** (Theorem 3.5 in [28])**.** *Let Γ be a constraint language and $k \geq 2$. If Γ is preserved by some $(k + 1)$-ary NU operation, then Γ is k-decomposable.*

3 Algorithms for the dual discriminator

In this section, we give two $|D|$-approximation algorithms for MinCostCSP(Γ) where Γ is a constraint language over D preserved by the dual discriminator operation, thus proving Theorem 1. Both algorithms will assume that the input instance is a satisfiable *binary*, *(2,3)-minimal* instance. We introduce and justify this assumption in Section 3.1, and then present a greedy algorithm in Section 3.2 and a LP-based algorithm in Section 3.3.

3.1 Reducing to satisfiable binary (2,3)-minimal instances

A CSP instance $I = (V, \mathcal{C})$ is *binary*, if every constraint in $\mathcal{C}$ has arity at most 2. Theorem 13 allows us to assume that the input instance is binary, since it is preserved by the dual discriminator operation which is a 3-ary NU operation. For any binary instance I, we may also write it as a triple $I = (V, \{R_u\}_{u\in V}, \{R_{u,v}\}_{u\neq v\in V})$, where we have one unary relation R_x for every $u \in V$ and one binary relation $R_{u,v}$ for every pair $(u, v) \in V$. Note that to write I in this form we may take the intersection of relations with the same scope or add complete relation on some variable(s) if none exists.

▶ **Definition 14** (See e.g., [6]). *A binary CSP instance $I = (V, \{R_u\}_{u\in V}, \{R_{u,v}\}_{u\neq v\in V})$ over domain D is* $(2,3)$-minimal *if*

(a) *For every distinct $u, v \in V$, $R_{v,u} = \{(b, a) \mid (a, b) \in R_{u,v}\}$.*

(b) *For every distinct $u, v \in V$, $R_u = \{a \in D \mid \exists b \in D, (a, b) \in R_{u,v}\}$.*

(c) *For every pairwise distinct $u, v, w \in V$ and $(a, b) \in R_{u,v}$, there exists $c \in R_w$ such that $(a, c) \in R_{u,w}$ and $(b, c) \in R_{v,w}$.*

Informally, the definition says that I is (2,3)-minimal if any partial satisfying assignment (that is, a partial assignment that does not immediately falsify any constraint) to two variables can be extended to a partial satisfying assignment to three variables. Given a binary CSP instance $I = (V, \{R_u\}_{u\in V}, \{R_{u,v}\}_{u\neq v\in V})$, we may transform it into a $(2,3)$-minimal instance using the following procedure:

repeat
  for every distinct $u, v, w \in V$ **do**
    $R_{u,v} \leftarrow \{(a, b) \in R_{u,v} \mid \exists c \in D, (a, c) \in R_{u,w} \wedge (b, c) \in R_{v,w}\}$
    $R_{u,v} \leftarrow \{(a, b) \in R_{u,v} \mid (b, a) \in R_{v,u}\}$
    $R_u \leftarrow \{a \in D \mid \exists b \in D, (a, b) \in R_{u,v}\}$
    $R_v \leftarrow \{b \in D \mid \exists a \in D, (a, b) \in R_{u,v}\}$
until none of the relations change

Clearly, when the above procedure stops, the instance must be $(2,3)$-minimal (otherwise the loop would have continued). The new instance may have constraints that are not in our original constraint language. However, since the new constraints are all obtained by pp-definitions from the original constraint language, they are preserved by the same polymorphisms (see Definition 27 and Theorem 28). In particular, the new instance is still preserved by the dual discriminator operation.

We say that a $(2,3)$-minimal instance is *trivial* if at least one of the unary relations is empty, and it is *nontrivial* otherwise. A trivial $(2,3)$-minimal instance has no satisfying assignment since no assignment can satisfy an empty relation. On the other hand, if the instance is nontrivial and its constraint language has *bounded width*, then such an instance is always guaranteed to have a satisfying assignment which we can find in polynomial time.

▶ **Theorem 15** ([4, 5]). *Let I be a nontrivial $(2,3)$-minimal instance whose constraint language has bounded width. Then I has a satisfying assignment which can be found in polynomial time.*

We will not define the term "bounded width" formally here (see Theorem 31 for a characterization). For us it is sufficient to note that any constraint language that is preserved by an NU polymorphism has bounded width [22], so any unsatisfiable instance will necessarily give rise to a trivial (2,3)-minimal instance. From now on we will assume that the (2,3)-minimal instance is nontrivial, and therefore satisfiable. Also, observe any satisfying assignment for the original instance will remain satisfying for the new (2,3)-minimal instance. This means any assignment we obtain on the new instance will give an approximation ratio at least as good on the original instance, and therefore we may shift our attention to this new instance instead.

We crucially use the following characterization for binary relations preserved by the dual discriminator operation. The same characterization was used by [16] in the context of robust satisfiability (these relations have also been studied under the name of *0/1/all relations* [14]). We sketch a short proof here for completeness.

▶ **Lemma 16** (See e.g. [6, 16]). *Let $R \subset D^2$ be a binary relation. Then R is preserved by the dual discriminator operation $d : D^3 \to D$ if and only if R is of one of the following forms:*
- *$R = P \times Q$ for some $P, Q \subseteq D$.*
- *$R = (\{u\} \times Q) \cup (P \times \{v\})$ for some $u \in P \subseteq D$ and $v \in Q \subseteq D$.*
- *$R = \{(u, \pi(u)) \mid u \in P\}$ for some $P, Q \subseteq D$ and bijective $\pi : P \to Q$.*

Proof. It is easy to verify that if R is one of these three types then it is preserved by d. Let us prove the other direction. Let $P = \{x \in D \mid \exists y \in D \text{ s.t. } (x, y) \in R\}$ and $Q = \{y \in D \mid \exists x \in D \text{ s.t. } (x, y) \in R\}$. For any $u \in P$, if there exist two distinct $v_1, v_2 \in Q$ such that $(u, v_1), (u, v_2) \in R$, then for any $v \in Q$, we have also $(u, v) \in R$. In other words, we have $\{u\} \times Q \subseteq R$. This is because we can find some $(u_1, v) \in R$ by the definition of Q, and apply f to the three pairs $(u_1, v), (u, v_1), (u, v_2) \in R$ to obtain (u, v) (observe that $f(u_1, u, u) = u$ and $f(v, v_1, v_2) = v$).

Now if we have two distinct $u_1, u_2 \in P$ such that $\{u_1\} \times Q \subseteq R$ and $\{u_2\} \times Q \subseteq R$, then for every $v \in Q$ there are two distinct u_1, u_2 such that $(u_1, v), (u_2, v) \in R$. By applying the above argument with P and Q reversed we get that for every $v \in Q$, $P \times \{v\} \subseteq R$, so it must be the case that $R = P \times Q$.

Now assume that there exists exactly one $u \in P$ such that $\{u\} \times Q \subseteq R$ and $R \neq P \times Q$. Then there exists $(u_1, v) \in R$ such that $u_1 \neq u$, so we have $P \times \{v\} \subseteq R$. Note that in this case we must have $R = (\{u\} \times Q) \cup (P \times \{v\})$, for the existence of any other pair would imply that $R = P \times Q$.

Finally, if $R \neq P \times Q$ and there is no $u \in P$ such that $\{u\} \times Q \subseteq R$, then there can also be no $v \in Q$ such that $P \times \{v\} \subseteq R$. So for every $u \in P$ there is a unique v such that $(u, v) \in R$, and we can find some bijective $\pi : P \to Q$ such that $R = \{(u, \pi(u)) \mid u \in P\}$. ◀

3.2 A greedy algorithm

We now present a greedy algorithm which is a generalization of a 2-approximation algorithm for MinOnes 2-SAT due to Gusfield and Pitt [24]. In this algorithm, we will greedily pick labels for the variables one by one, and each label we pick may potentially restrict the set of feasible labels for some other variables. For general CSPs, this restriction can be rather arbitrary and difficult to control. However, for binary $(2,3)$-minimal instances preserved by the dual discriminator operation, the restriction is very simple: either the set of feasible labels is unchanged, or it is restricted to a singleton set (as is guaranteed by Lemma 16). This means for variables where proper restriction happens we can simply fix it to the label in the singleton set. On the other hand, the variables whose set of feasible labels didn't change induce a sub-instance of the original instance and the algorithm may recurse on this sub-instance. This property guarantees that we will always produce a satisfying assignment, as is formalized in the following lemma.

▶ **Definition 17.** *Let $I = (V, \{R_u\}_{u\in V}, \{R_{u,v}\}_{u,v\in V})$ be a (2,3)-minimal instance. For any $u, v \in V$, $a \in R_u$ and $b \in R_v$, we say that v is fixed to b by assigning a to u if $u = v$ and $a = b$, or $u \neq v$ and $R_{u,v} \cap (\{a\} \times R_v) = \{(a, b)\}$. In other words, b is the only feasible label left for v if we assign the label a to u.*

▶ **Lemma 18.** *Let $I = (V, \{R_u\}_{u\in V}, \{R_{u,v}\}_{u,v\in V})$ be a nontrivial (2,3)-minimal instance preserved by the dual discriminator operation. Let $u_0 \in V$, $a \in R_{u_0}$, S be the set of variables fixed by assigning a to u_0. For every $v \in S$, let $A_S(v)$ be the unique label that v is fixed to by assigning a to u_0. Let $A' : V\backslash S \to D$ be any satisfying assignment for the induced (2,3)-minimal instance $I' = (V\backslash S, \{R_u\}_{u\in V\backslash S}, \{R_{u,v}\}_{u,v\in V\backslash S})$, then*

$$A : V \to D, \quad A(v) = \begin{cases} A_S(v) & \text{if } v \in S, \\ A'(v) & \text{otherwise} \end{cases}$$

is a satisfying assignment for I.

Proof. Clearly all unary constraints in I are satisfied. Any binary constraint $R_{u,v}$ where $u, v \in V\backslash S$ is satisfied since A' is a satisfying assignment. The remaining two cases, where $u \in S, v \in V\backslash S$ or $u, v \in S$, follow from the following claim:

▷ Claim 19. For every distinct $v \in S$ and $w \in V$, if $(a, c) \in R_{u_0,w}$, then we have $(A_S(v), c) \in R_{v,w}$.

Proof. This claim clearly holds for $v = u_0$. Suppose $v \neq u_0$. Since $(a, c) \in R_{u_0,w}$, by (2,3)-minimality we can find $b \in R_v$ such that $(a, b) \in R_{u_0,v}, (b, c) \in R_{v,w}$. This b must coincide with $A_S(v)$, since v is fixed by assigning a to u_0. Thus, we have $(A_S(v), c) \in R_{v,w}$. ◁

For $u \in S, v \in V\backslash S$, since $(a, A'(v)) \in R_{u_0,v}$, we may apply the claim and obtain that $(A_S(u), A'(v)) \in R_{u,v}$, so this constraint is satisfied. For $u, v \in S$, again since since $(a, A_S(v)) \in R_{u_0,v}$, we may apply the claim and obtain that $(A_S(u), A_S(v)) \in R_{u,v}$, so this constraint is also satisfied. It follows that all binary constraints in I are satisfied by A, so it is indeed a satisfying assignment for I. ◀

To guarantee a constant-factor approximation, we need to pick labels in a clever way. One naive idea is to compute the total cost incurred by variables fixed by each label and pick the label that minimizes this cost. However, this does not work because the optimum assignment may incur more cost but save by fixing more variables. One way to fix this naive idea is to consider a *derived cost* t, instead of the original cost `cost`. In particular, when we face a decision for some variable u (i.e., $|R_u| \geq 2$), we pick the label that minimizes the total derived cost of fixed variables. We then pay $|R_u| \leq |D|$ times this cost towards reducing the derived cost of *all* variable-label pairs that are or could've been fixed by assignment to u. The key idea here is that one of the labels in R_u will be taken by the optimal assignment, so the amount we pay are always within a factor of $|D|$ from the total cost of the optimal assignment.

We now formally present the algorithm and its analysis. The pseudocode for the algorithm can be found in Algorithm 1.

Algorithm 1 Greedy algorithm for MINCOSTCSP.

Input: $I = (V, \{R_u\}_{u \in V}, \{R_{u,v}\}_{u,v \in V})$ a (2,3)-minimal instance, $\texttt{cost} : V \times D \to \mathbb{R} \cup \{+\infty\}$
Output: $A : V \to D$ a satisfying assignment for I.
Initialization: $t : V \times D \to \mathbb{R}^{\geq 0} \cup \{+\infty\}, t(u,a) \leftarrow \texttt{cost}(u,a)$

1: **procedure** MINCOSTHOM-GREEDY(I)
2: $S \leftarrow \{u \in V \mid |R_u| \geq 2\}$ ▷ The set of undetermined variables
3: **for** $u \in V \backslash S$ **do**
4: $A(u) \leftarrow$ the unique label a in R_u
5: $t(u,a) \leftarrow 0$
6: **while** $S \neq \varnothing$ **do**
7: Pick an arbitrary $v \in S$
8: **for** $a \in R_v$ **do**
9: $F_a \leftarrow \{u \in S \mid u$ is fixed by assigning $v = a\}$
10: **for** $u \in F_a$ **do**
11: $A_a(u) \leftarrow$ the label u is fixed to if assigning $v = a$
12: $c_a \leftarrow \sum_{u \in F_a} t(u, A_a(u))$ ▷ Calculate the potential cost for every label a
13: $a_0 \leftarrow \arg\min_{a \in R_v} c_a$
14: Initialize an array Δt indexed by $V \times D$ to all zeros.
15: ▷ Δt stores the amount by which we will decrease t this round.
16: ▷ Intuitively, this is the cost we need to pay in this round.
17: **for** $a \in R_v$ **do**
18: Find $0 \leq t_a(u) \leq t(u, A_a(u))$ for every $u \in F_a$ such that $\sum_{u \in F_a} t_a(u) = c_{a_0}$
19: ▷ Always possible since $\sum_{u \in F_a} t(u, A_a(u)) = c_a \geq c_{a_0}$
20: ▷ In particular, $t_{a_0}(u) = t(u, A_{a_0}(u))$ for every $u \in F_{a_0}$
21: **for** $u \in F_a$ **do**
22: $\Delta t(u, A_a(u)) \leftarrow \max(\Delta t(u, A_a(u)), t_a(u))$ ▷ Choose the largest decrease
23: **for** $u \in V$ **do**
24: **for** $a \in D$ **do**
25: $t(u,a) \leftarrow t(u,a) - \Delta t(u,a)$ ▷ Update the costs
26: **for** $u \in F_{a_0}$ **do**
27: $A(u) \leftarrow A_{a_0}(u)$ ▷ Set values to the variables fixed by assigning a_0 to u
28: $S \leftarrow S \backslash F_{a_0}$ ▷ Remove variables fixed in this step from the set of undetermined variables
29: **return** A

We start by making the following observations.

▶ **Observation 20.** *Let t^{end} be the values of t when A is returned. For every $u \in V$, $t^{\text{end}}(u, A(u)) = 0$.*

Proof. For variables fixed outside the while loop, this follows from Line 5. For variables that obtained its assignment inside the while loop, this follows from Lines 18 and 25 (since $\Delta t(u, A_{a_0}(u)) = t(u, A_{a_0}(u))$ for every $u \in F_{a_0}$). ◀

▶ **Observation 21.** *The value of t never increases during the algorithm. In particular, for every $u \in V$ and $a \in D$, we have $\texttt{cost}(u,a) - t^{\text{end}}(u,a) \geq 0$.*

Proof. This is because Δt is always nonnegative. ◀

▶ **Observation 22.** *For every $u \in V$ and $a \in D$, we have $t(u,a) \geq 0$ at any point of the algorithm.*

Proof. This is because when an update happens at Line 22, the updated value for $\Delta t(u, A_a(u))$ is equal to $t_a(u) \leq t(u, A_a(u))$. ◀

The guarantee on the approximation ratio is based on the following claim.

▷ Claim 23. Let $A' : V \to D$ be any satisfying assignment for I. For each iteration of the while loop beginning at Line 6, let $t^{\text{before}}, t^{\text{after}}$ be the values of t at the beginning and the end of the iteration respectively. Then we have

$$\sum_{u \in V, a \in D} (t^{\text{before}}(u,a) - t^{\text{after}}(u,a)) \leq |D| \cdot \sum_{u \in V} (t^{\text{before}}(u, A'(u)) - t^{\text{after}}(u, A'(u)))$$

Proof. Let the variables be defined as they are in the algorithm. For every $u \in V$ and $a \in D$ define the auxiliary variables

$$\Delta'(u,a) = \sum_{a' : a' \in R_v, u \in F_{a'}, a = A_{a'}(u)} t_{a'}(u).$$

Note that the quantity $(t^{\text{before}}(u,a) - t^{\text{after}}(u,a))$ is nonzero only if there exists some $a' \in R_v$ such that $u \in F_{a'}$ and $a = A_{a'}(u)$, and this quantity is given by the value stored in $\Delta t(u,a)$ at the beginning for Line 23. By construction (see Line 22), for any such a' we have

$$t_{a'}(u) \leq \Delta t(u,a) \leq \Delta'(u,a),$$

since $\Delta t(u,a)$ is equal to the largest summand in the sum defining $\Delta'(u,a)$. It follows that

$$\begin{aligned}
\sum_{u \in V, a \in D} (t^{\text{before}}(u,a) - t^{\text{after}}(u,a)) &\leq \sum_{u \in V, a \in D} \Delta'(u,a) \\
&= \sum_{u \in V, a \in D} \left(\sum_{a' : a' \in R_v, u \in F_{a'}, a = A_{a'}(u)} t_{a'}(u) \right) \\
&= \sum_{a' \in R_v} \sum_{u \in F_{a'}} t_{a'}(u) = |R_v| \cdot c_{a_0} \leq |D| \cdot c_{a_0}.
\end{aligned}$$

Here, the last equality follows from $\sum_{u \in F_{a'}} t_{a'}(u) = c_{a_0}$ (see Line 18). On the other hand, for $a = A'(v)$ we must have $A_a(u) = A'(u)$ for every $u \in F_a$, and therefore we have

$$\begin{aligned}
\sum_{u \in V} (t^{\text{before}}(u, A'(u)) - t^{\text{after}}(u, A'(u))) &\geq \sum_{u \in F_a} (t^{\text{before}}(u, A_a(u)) - t^{\text{after}}(u, A_a(u))) \\
&= \sum_{u \in F_a} \Delta t(u, A_a(u)) \geq \sum_{u \in F_a} t_a(u) = c_{a_0}.
\end{aligned}$$

This establishes the claim. ◁

▶ **Theorem 24.** *Algorithm 1 returns a satisfying assignment $A : V \to D$ whose cost is at most $|D|$ times the optimal cost.*

Proof. The algorithm returns a satisfying assignment by Lemma 18.

Let $A' : V \to D$ be an arbitrary assignment. We have the following chain of inequalities:

$$\begin{aligned}
\sum_{v\in V} \texttt{cost}(v, A(v)) &\le \sum_{v\in V} \left(\texttt{cost}(v, A(v)) - t^{\mathrm{end}}(v, A(v))\right) && \text{(Observation 20)}\\
&\le \sum_{v\in V}\sum_{a\in D} \left(\texttt{cost}(v, a) - t^{\mathrm{end}}(v, a)\right) && \text{(Observation 21)}\\
&\le |D| \cdot \sum_{v\in V} \left(\texttt{cost}(v, A'(v)) - t^{\mathrm{end}}(v, A'(v))\right) && \text{(Claim 23)}\\
&\le |D| \cdot \sum_{v\in V} \texttt{cost}(v, A'(v)) && \text{(Observation 22)}
\end{aligned}$$

Since A' is arbitrary, the theorem follows by taking A' to be an optimal assignment. ◀

3.3 An LP-based algorithm

We now present another approximation algorithm that is based on the *basic linear programming* (BLP) relaxation for the MinCostCSP problem [42, 35]. Given an MinCostCSP(Γ) instance $I = (V, \mathcal{C}, \texttt{cost})$, its BLP relaxation can be formulated as follows.

$$\begin{array}{lll}
\text{minimize} & \displaystyle\sum_{v\in V}\sum_{a\in D} p_{v,a}\cdot \texttt{cost}(v,a) & \\
\text{subject to} & \displaystyle\sum_{a\in D} p_{v,a} = 1, & \forall v \in V,\\
& \displaystyle\sum_{x\in R} p_{C,x} = 1, & \forall C = (R,S) \in \mathcal{C},\\
& \displaystyle\sum_{x\in R} p_{C,x}\cdot \mathbf{1}[x(v) = a] = p_{v,a}, & \forall C = (R,S) \in \mathcal{C},\ \forall v \in S,\ \forall a \in D,\\
& p_{v,a},\ p_{C,x} \ge 0, & \forall v \in V, a \in D, C = (R,S) \in \mathcal{C}, x \in R.
\end{array}$$

Figure 1 BLP relaxation for MinCostCSP(Γ).

Recall that each constraint C is represented by a pair (R, S) where R is some k-ary relation and S is a k-tuple of variables to which R is applied. Here in the LP formulation we abuse the notation slightly and think of each satisfying tuple $x \in R$ also as a partial assignment $S \to D^k$, and $x(v)$ is the label that x assigns to v. Informally, the linear program maintains a distribution of satisfying partial assignments for each constraint, and it requires that if a variable appears in multiple constraints, then the marginal distribution on this variable should be consistent across these constraints. The objective function to minimize is then the expected cost under these marginal distributions. It is easy to see that any integral assignment to I corresponds to distributions supported on a single element, so this is indeed a relaxation of the original problem. We remark that in polynomial time we can actually enforce the marginal consistency requirement on any constant-sized set of variables, and thus obtain tighter relaxations on the so-called *Sherali-Adams hierarchy* ([42]). However, this will not be needed for our purpose.

For any MinCostCSP instance I, let LP(I) be the optimal value of the BLP relaxation for I. Our LP-based algorithm is as follows. Note that as before, we assume that I is given as a non-trivial $(2, 3)$-minimal instance.

Algorithm 2 LP-based algorithm for MinCostCSP.

Input: $I = (V, \{R_u\}_{u\in V}, \{R_{u,v}\}_{u,v\in V})$ a (2,3)-minimal instance
Output: $A : V \to D$ a satisfying assignment for I.

```
procedure MinCostHom-LP(I)
    Solve the Basic LP relaxation and obtain {p_{v,a} | (v,a) ∈ V × D}.
    for v ∈ V do
        R'_v ← {a ∈ R_v | p_{v,a} ≥ 1/|D|}
    for every distinct u, v ∈ V do
        R'_{u,v} ← R_{u,v} ∩ (R'_u × R'_v)
    Return any satisfying solution to the (2,3)-minimal instance I' = (V, {R'_u}_{u∈V}, {R'_{u,v}}_{u,v∈V}).
```

In words, we remove labels that receive tiny LP probabilities (less than $1/|D|$) and solve the new instance. Observe that R'_v is nonempty for every $v \in V$, since at least one of the $|D|$ labels will get a probability that is at least $1/|D|$.

▶ **Theorem 25.** *Let I be a nontrivial (2,3)-minimal instance. Then on input I, Algorithm 2 returns in polynomial time a satisfying assignment to I whose cost is at most $|D| \cdot \mathrm{LP}(I)$.*

Proof. We first show that I' is indeed a (2,3)-minimal instance. We verify the conditions in Definition 14 one by one. For readers' convenience, we restate the conditions in italic.

(a) *For every distinct $u, v \in V$, $R'_{v,u} = \{(b,a) \mid (a,b) \in R'_{u,v}\}$.* This is equivalent to the statement $(a,b) \in R'_{u,v} \Leftrightarrow (b,a) \in R'_{v,u}$. By symmetry, it is sufficient to show that $(a,b) \in R'_{u,v} \Rightarrow (b,a) \in R'_{v,u}$. Let $(a,b) \in R'_{u,v} \subseteq R_{u,v}$, then $a \in R'_u$ and $b \in R'_v$. Since I is (2,3)-minimal, we have that $(b,a) \in R_{v,u}$, so $(b,a) \in R_{v,u} \cap (R'_v \times R'_u) = R'_{v,u}$.

(b) *For every distinct $u, v \in V$, $R'_u = \{a \mid \exists b \in R'_v, (a,b) \in R'_{u,v}\}$.* By construction we have $\{a \mid \exists b \in R'_v, (a,b) \in R'_{u,v}\} \subseteq R'_u$. To show the other inclusion, let us pick $a \in R'_u$. By Lemma 16, we have the following two cases:
- $\{a\} \times R_v \subseteq R_{u,v}$. In this case, we have $\{a\} \times R'_v \subseteq R_{u,v} \cap (R'_u \times R'_v) = R'_{u,v}$. Since R'_v is nonempty, we have that $a \in \{a \mid \exists b \in R'_v, (a,b) \in R'_{u,v}\}$.
- $\{a\} \times R_v \not\subseteq R_{u,v}$. In this case, there is a unique $b_0 \in R_v$ such that $(a, b_0) \in R_{u,v}$. By the LP constraint, we have $p_{v,b_0} \geq p_{u,a} \geq 1/|D|$, and therefore $b_0 \in R'_v$ and $a \in \{a \mid \exists b \in R'_v, (a,b) \in R'_{u,v}\}$.

(c) *For every pairwise distinct $u, v, w \in V$ and $(a,b) \in R'_{u,v}$, there exists $c \in R'_w$ such that $(a,c) \in R'_{u,w}$ and $(b,c) \in R'_{v,w}$.* Similar to part (b), we again have two cases:
- $\{a\} \times R_w \not\subseteq R_{u,w}$ or $\{b\} \times R_w \not\subseteq R_{v,w}$. Since I is (2,3)-minimal, there exists $c \in R_w$ such that $(a,c) \in R_{u,w}$ and $(b,c) \in R_{v,w}$. By Lemma 16, c is either the unique element in R_w such that $(a,c) \in R_{u,w}$, or the unique element in R_w such that $(b,c) \in R_{v,w}$. In either case, we can conclude that $p_{w,c} \geq \min(p_{u,a}, p_{v,b}) \geq 1/|D|$, so $c \in R'_w$ as required.
- $\{a\} \times R_w \subseteq R_{u,w}$ and $\{b\} \times R_w \subseteq R_{v,w}$. In this case, since R'_w is non-empty, there exists some $c_0 \in R'_w \subseteq R_w$ so we have $(a, c_0) \in R'_{u,w}$ and $(b, c_0) \in R'_{v,w}$.

This establishes that I' is (2,3)-minimal. Note that every binary relation in I' is also of the three types described in Lemma 16, so it is also preserved by the dual discriminator operation. By Theorem 15, we can find a satisfying assignment $A : V \to D$ for I' in polynomial time. Note that for this assignment, we have

$$\begin{aligned}\sum_{v\in V} \texttt{cost}(v, A(v)) &\leq \sum_{v\in V} |D| \cdot p_{v,A(v)} \cdot \texttt{cost}(v, A(v)) \\ &\leq |D| \cdot \sum_{v\in V}\sum_{a\in D} p_{v,a} \cdot \texttt{cost}(v,a) = |D| \cdot \mathrm{LP}(I).\end{aligned}$$ ◀

4 NU polymorphism as necessary condition for constant-factor approximability

In this section, we establish the following necessary condition for the constant-factor approximability of MinCostCSP(Γ) problem. We note that Dalmau et al. [18] obtained a similar necessary condtion for the problem of MinCSP. We will closely follow their proof.

▶ **Theorem 26** (Theorem 2 restated). *Let Γ be a constraint language. If MinCostCSP(Γ) has a constant-factor approximation, then* $\mathsf{Pol}(\Gamma)$ *contains a conservative NU polymorphism, unless P = NP.*

4.1 Gadget reductions and primitive positive interpretations

Let us first establish some sufficient condition for reductions between MinCostCSPs. Given two constraint languages Γ_1 and Γ_2, we write MinCostCSP(Γ_1) $\leq_{\mathrm{CF}}$ MinCostCSP(Γ_2) if the constant-factor approximability for MinCostCSP(Γ_2) implies the constant-factor approximability for MinCostCSP(Γ_1). By definition, $\leq_{\mathrm{CF}}$ is transitive.

▶ **Definition 27** (pp-definition). *Let Γ be a constraint language over D and R a k-ary relation over the same domain. We say that Γ* pp-defines *R, if there exist some $m > 0$ and a conjunction $\mathfrak{C}$ over variables $x_1, \ldots, x_k, y_1, \ldots, y_m$ consisting of relations in Γ and the equality relation ($\mathrm{eq}_D := \{(u, v) \in D^2 \mid u = v\}$) over D, such that*

$$R(x_1, \ldots, x_k) \equiv \exists y_1 \cdots \exists y_m \mathfrak{C}.$$

If Γ' is another constraint language over the same domain D, then we say that Γ pp-defines Γ' if Γ pp-defines every relation in Γ'.

The following theorem establishes a connection (often referred to as *Galois correspondence* in the literature) between polymorphisms and pp-definitions. We say that a relation R with arity k is *irreducible*, if for every distinct $i, j \in [k]$, there exists $(x_1, \ldots, x_k) \in R$ such that $x_i \neq x_j$.

▶ **Theorem 28** ([7, 23]). *Let Γ be a constraint language and R some k-ary relation over the same domain. Then we have*

- $\mathsf{Pol}(\Gamma) \subseteq \mathsf{Pol}(R)$ *if and only Γ pp-defines R.*
- *If R is irreducible, then* $\mathsf{Pol}(\Gamma) \subseteq \mathsf{Pol}(R)$ *if and only Γ pp-defines R without using the equality relation.*

▶ **Definition 29** (pp-interpretation). *Let Γ_1 and Γ_2 be constraint languages over domains D and E respectively. We say that Γ_1 pp-interprets Γ_2 if there exist $n \in \mathbb{N}$, $F \subseteq D^n$, and a surjective function $f : F \to E$ such that Γ_1 pp-defines the following relations:*

- *F as an n-ary relation over D.*
- *For every $R \in \Gamma_2$ with some arity k, the relation*

$$f^{-1}(R) = \{(x^{(1)}, x^{(2)}, \ldots, x^{(k)}) \in D^{kn} \mid x^{(i)} \in F \text{ for } i = 1, \ldots, k, (f(x^{(1)}), \ldots, f(x^{(k)})) \in R\}$$

  Here each $x^{(i)}$ is a n-tuple over D and we are thinking of $(x^{(1)}, x^{(2)}, \ldots, x^{(k)})$ as a flattened kn-tuple over D.
- *The relation*

$$f^{-1}(\mathrm{eq}_E) = \left\{(x^{(1)}, x^{(2)}) \in D^{2n} \mid x^{(i)} \in F \text{ for } i = 1, 2,\ f(x^{(1)}) = f(x^{(2)})\right\}$$

  Here again $x^{(1)}$ and $x^{(2)}$ are n-tuples over D and $(x^{(1)}, x^{(2)})$ is a flattened $2n$-tuple.

We say that Γ_1 pp-interpretes Γ_2 in the first power if $n = 1$ in the above definition. It is known in the case of standard decision CSP that the existence of a pp-interpretation implies a gadget reduction, where we simply replace constraints in Γ' with constraints in Γ using the pp-definitions. However, in the case of MinCostCSP, for the purpose of the reduction we would also need to translate the costs between the two instances. This is straightforward for $n = 1$, but there seems to be no natural way of doing this if we are using a pp-interpretation with $n \geq 2$.

▶ **Lemma 30.** *Let Γ_1 be a constraint language over D and Γ_2 a constraint language over E. If Γ_1 pp-interpretes Γ_2 in the first power and one of the following holds:*
- *The equality relation eq_D over D is in Γ_1.*
- *Every $R \in \Gamma_2$ is irreducible.*

Then MinCostCSP$(\Gamma_2) \leq_{\mathrm{CF}}$ MinCostCSP(Γ_1)

Proof. Let $F \subseteq D$ and $f : F \to E$ be as in the definition of pp-interpretation. Let $I_2 = (V, \mathcal{C}, \mathtt{cost})$ be a MinCostCSP(Γ_2) instance. We define a MinCostCSP(Γ_1) instance $I_1 = (V, \mathcal{C}', \mathtt{cost}')$ as follows:
- I_1 has the same set of variables V as I_2.
- For each constraint $C = (R, S) \in \mathcal{C}$, we would like to add a constraint $C' = (f^{-1}(R), S)$ to $\mathcal{C}'$. To do this, without loss of generality assuming $S = \{x_1, \ldots, x_k\}$, we take the pp-definition of $f^{-1}(R)$, which is of the form

  $$\exists y_1 \cdots \exists y_m \mathfrak{C}.$$

  Here $y_1, \ldots, y_m$ are auxiliary variables with zero costs and $\mathfrak{C}$ is a conjunction of constraints, each being either a relation from Γ_1 or eq_D applied to some of the variables in $\{x_1, \ldots, x_k, y_1, \ldots, y_m\}$. Now if $\mathrm{eq}_D \in \Gamma_1$, then we can think of $\mathfrak{C}$ as a CSP(Γ_1) instance with variables being $\{x_1, \ldots, x_k, y_1, \ldots, y_m\}$, and this instance can be satisfied if and only $(x_1, \ldots, x_k) \in f^{-1}(R)$, so we add (the constraints and the auxiliary variables of) this instance to I_2. If $\mathrm{eq}_D \notin \Gamma_1$, but every $R \in \Gamma_2$ is irreducible, then by Theorem 28, we may assume that $\mathfrak{C}$ does not contain eq_D and therefore we can still write it as an instance of CSP(Γ_1) and add it to I_2.
- For each $x \in V$ and $a \in D$, let $\mathtt{cost}'(x, b) := \begin{cases} \mathtt{cost}(x, a) & \text{if } \exists a \in E \text{ s.t. } b \in f^{-1}(a), \\ +\infty & \text{otherwise.} \end{cases}$

Clearly, for every satisfying assignment $A : V \to E$ to I_2, we may define $A' : V \to F$ such that $A'(x) \in f^{-1}(A(x))$, and A' will be a satisfying assignment to I_1 with the same cost. In particular, this means that $\mathrm{Opt}(I_1) \leq \mathrm{Opt}(I_2)$.

Now if we have a constant-factor approximation algorithm for MinCostCSP(Γ_1), we can use it to obtain a solution $A_1 : V \to D$ such that $\mathtt{cost}_{I_1}(A_1) \leq t \cdot \mathrm{Opt}(I_1)$ for some constant t independent of I_1. In fact, we may assume $A_1 : V \to F$, since every label not in F has infinite cost. Take $A_2 : V \to E, x \mapsto f(A_1(x))$, then by construction A_2 is a satisfying assignment for I_2, and

$$\mathtt{cost}_{I_2}(A_2) = \mathtt{cost}_{I_1}(A_1) \leq t \cdot \mathrm{Opt}(I_1) \leq t \cdot \mathrm{Opt}(I_2).$$

Thus we obtain a constant-factor approximation algorithm for MinCostCSP(Γ_2) as well. ◀

We refer interested readers to the survey by Barto el al. [6] which contains a more detailed exposition on pp-interpretation (and its generalization *pp-construction*) in the context of decision CSPs.

4.2 Proof of Theorem 26

The proof contains two cases: either Γ has unbounded width, or it has bounded width. We use the following characterization for bounded-widthness of constraint languages.

▶ **Theorem 31** ([5, 17]). *Let Γ be a constraint language that contains all singleton relations. Γ is not bounded width if and only if there exists some nontrivial finite abelian group G such that Γ pp-interprets $\Gamma(G)$ in the first power using pp-definitions without equality.*

Here G being nontrivial means it has at least 2 elements, and $\Gamma(G)$ is the set of relations $\{R_{abc} = \{(x, y, z) \in G^3 \mid ax + by + cz = 0\} \mid a, b, c \in \mathbb{Z}\}$[3] over G.

In the unbounded-width case we shall use a reduction from the Nearest Codeword problem, and in the bounded-width case we reduce from the hypergraph vertex cover problem.

▶ **Definition 32.** *In the Nearest Codeword problem over a finite field $\mathbb{F}_p$, we are given a matrix $A \in \mathbb{F}_p^{m\times n}$ and a vector $x \in \mathbb{F}_p^n$, and we are asked to find a vector $y \in \mathbb{F}_p^n$ such that $Ay = 0$ and the number of nonzero entries in $x - y$ is minimized.*

▶ **Definition 33.** *In the k-uniform hypergraph vertex cover problem, we are given a k-uniform hypergraph (namely, each hyperedge is contains k vertices), and our goal is to choose a minimum number of vertices so that from each hyperedge we have chosen at least one vertex.*

The following theorems give the best known NP-hardness results for approximating these two problems.

▶ **Theorem 34** ([20, 13]). *The Nearest Codeword problem over any finite field $\mathbb{F}_p$ is NP-hard to approximate within a factor of $2^{\log^{1-\epsilon}(n)}$, for any constant $\epsilon > 0$.*

▶ **Theorem 35** ([19]). *The k-uniform hypergraph vertex cover problem is NP-hard to approximate within a factor of $k - 1 - \epsilon$, for any $k \geq 3$ and $\epsilon > 0$.*

We remark that if we further assume the Unique Games Conjecture, then we can improve the hardness factor for k-uniform hypergraph vertex cover from $k - 1 - \epsilon$ to $k - \epsilon$ [2], but this difference does not matter for us here.

The following is a simple corollary from the hardness of the Nearest Codeword problem.

▶ **Corollary 36.** *Let Γ_p be the set of all relations of the form $R_{abc} = \{(x, y, z) \in \mathbb{F}_p^3 \mid ax + by + cz = 0\}$[4] where $a, b, c \in \mathbb{F}_p \backslash \{0\}$ over some finite field $\mathbb{F}_p$. Then* MinCostCSP(Γ_p) *is NP-hard to approximate within a factor of $2^{\log^{1-\epsilon}(n)}$.*

Proof. We show how to cast the Nearest Codeword problem over $\mathbb{F}_p$ as a MinCostCSP(Γ_p) instance. We first add the variables $y_1, \ldots, y_n$ denoting entries of y. For each y_i, we impose a cost of 1 if it is not equal to x_i, and 0 if it is equal to x_i. This models the minimum Hamming weight requirement. To model the constraint $Ay = 0$, we first note that Γ_p can also be used to simulate $R_{abc} = \{(x, y, z) \in \mathbb{F}_p^3 \mid ax + by + cz = 0\}$ if one of a, b, c is zero. This can be achieved as follows: to obtain the constraint $ax + by = 0$, we create a dummy variable z and add the constraint $R_{abc}(x, y, z)$ for an arbitrary nonzero c, and then we set the non-zero label costs for z to be all infinite and its zero label cost to be just 0, effectively forcing this variable to be zero. This can be extended easily if two of a, b, c are zeros.

---

[3] Here ax denotes the sum of a copies of x. Note that this is a finite set of relations, since G is a finite group.

[4] Here ax denotes the $\mathbb{F}_p$ multiplication between field elements $a, x \in \mathbb{F}_p$.

Now we need to transform each linear equation of the form $\sum_{i=1}^{m} A_{ki}y_i = 0$ so that left hand side contains at most 3 variables. This can be done via the standard trick: by introducing a new auxiliary variable z (which has zero cost for any label), we may rewrite $\sum_{i=1}^{m} A_{ki}y_i = 0$ equivalently as $A_{k1}y_1 + A_{k2}y_2 + z = 0$ and $-z + \sum_{i=3}^{m} A_{ki}y_i = 0$, and thereby reducing the number of variables on the left hand side by 1. We repeat this procedure until all equations have at most 3 variables on the left hand side. It is clear that the resulting instance is a MinCostCSP(Γ_p) instance which is equivalent to the original Nearest Codeword instance. By Theorem 34, we can therefore conclude that MinCostCSP(Γ_p) is also NP-hard to approximate within a factor of $2^{\log^{1-\epsilon}(n)}$. ◀

▶ **Lemma 37.** *Let Γ be a constraint language such that MinCostCSP(Γ) has a constant-factor approximation, then Γ is bounded-width unless P = NP.*

Proof. We prove the contrapositive. Let Γ be a constraint language with unbounded width. Without loss of generality, assume that Γ contains all unary relations. Then by Theorem 31, there exists some nontrivial finite abelian group G such that Γ pp-interprets $\Gamma(G)$ in the first power. In particular, Γ pp-interprets the set of all irreducible relations $\Gamma(G)^{\text{irr}} \subseteq \Gamma(G)$ in the first power. By Lemma 30, we have MinCostCSP($\Gamma(G)^{\text{irr}}$) $\leq_{\text{CF}}$ MinCostCSP(Γ).

We now claim that MinCostCSP($\Gamma(G)^{\text{irr}}$) contains MinCostCSP(Γ_p) as a special case for some prime p. Note that G must contain some cyclic subgroup of prime order: one may find this subgroup by taking a subgroup of prime order of some cyclic subgroup generated by a single nonzero element in G. We may identify this subgroup of order p with the finite field $\mathbb{F}_p$. Note that any relation $R_{abc} = \{(x, y, z) \in \mathbb{F}_p^3 \mid ax + by + cz = 0\}$ over $\mathbb{F}_p$ is irreducible if a, b, c are all nonzero, so these relations are contained in $\Gamma(G)^{\text{irr}}$. So we have that MinCostCSP($\Gamma(G)^{\text{irr}}$) contains MinCostCSP(Γ_p) as a subproblem (where we set the cost of any label outside $\mathbb{F}_p$ to be infinite). It follows that MinCostCSP($\Gamma(G)^{\text{irr}}$), and therefore MinCostCSP(Γ), do not have a constant-factor approximation unless P = NP. ◀

For the bounded-width case, we use the following reduction from the hypergraph vertex cover problem.

▶ **Lemma 38** ([18])**.** *Let Γ be a bounded-width constraint language which is not preserved by any NU operation. If Γ contains all unary singleton relations, then for every $k \geq 1$, there is a k-ary relation R pp-definable from Γ and $a, b \in D$ such that*

$$R \cap \{a, b\}^k = \{a, b\}^k \backslash \{(a, a, \ldots, a)\}.$$

▶ **Lemma 39.** *Let Γ be a bounded-width constraint language which is not preserved by any NU operation. Then MinCostCSP(Γ) does not have a constant-factor approximation unless P = NP.*

Proof. Assume that Γ has all unary relations without loss of generality. Note that the k-uniform hypergraph vertex cover problem is the MinCostCSP problem with a single relation $R_k = \{0, 1\}^k \backslash \{(0, 0, \ldots, 0)\}$, which is pp-definable from Γ by Lemma 38 (by thinking of a as 0 and b as 1, and the assumption that $\{a, b\}$ as a unary relation is in Γ). Observe that R_k is irreducible, so the reduction in Lemma 30 implies that MinCostCSP(Γ) is as hard to approximate as the k-uniform hypergraph vertex cover problem, in particular, by Theorem 35, it is NP-hard to approximate MinCostCSP(Γ) within a factor of $k - 1 - \epsilon$ for any $\epsilon > 0$. Since k can be arbitrarily large, this implies that MinCostCSP(Γ) does not have a constant-factor approximation, unless P = NP. ◀

Theorem 26 can now be obtained by combining these two cases.

Proof of Theorem 26. Let Γ be a constraint language such that MinCostCSP(Γ) has a constant-factor approximation and assume that P $\neq$ NP, then by Lemma 37, Γ must be bounded-width. It then follows from Lemma 39 that Γ must be preserved by some NU operation. ◀

▶ Remark 40. In the Boolean case ($|D| = 2$), it follows from Khanna et al.'s classification [30] as well as Post's classification of Boolean clones [38] that the necessary condition of having a conservative NU polymorphism is sufficient as well. Recall that we have the following three classes of constant-factor approximable Boolean MinCostCSPs:

- Γ can be expressed as a 2CNF-formula. In this case, Γ is preserved by the (unique) majority operation. (This case also includes constraint languages that are width-2 affine for which MinCostCSP can be solved to optimality.)
- Γ is expressible as a CNF formula where each clause is of the form $x_1 \vee \cdots \vee x_k$, $\neg x_1 \vee x_2$, or $\neg x_1$ where $k \leq K$ for some K depending on Γ. In this case Γ is preserved by the $(K+1)$-ary NU operation th_2^{K+1}, where

$$\mathrm{th}_p^n(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if } |\{i \in [n] \mid x_i = 1\}| \geq p, \\ 0 & \text{otherwise.} \end{cases}$$

- Γ is expressible as a CNF formula where each clause is of the form $\neg x_1 \vee \cdots \vee \neg x_k$, $x_1 \vee \neg x_2$, or x_1 where $k \leq K$ for some K depending on Γ. In this case Γ is preserved by the $(K+1)$-ary NU operation th_K^{K+1}.

It can be easily verified using Post's Lattice [38] that any constraint language whose polymorphism clone contains an NU operation can be reduced to one of the three cases above. However, as soon as $|D| \geq 3$, the condition of being preserved by some NU operation is no longer sufficient (for example, see Theorem 45 in the following subsection).

4.3 A hard predicate with a majority polymorphism

We now present a binary relation P_H which has a conservative majority polymorphism, but nonetheless MinCostCSP(P_H) is hard to approximate within any constant factor, unless UGC fails. This implies that the existence of an NU polymorphism is in general not sufficient for constant-factor approximability assuming UGC.

▶ **Definition 41.** *Let P_H be the binary relation on domain $A = \{0, 1, 2\}$ such that $P_H(x, y)$ holds if and only if $x \neq y$ or $x = y = 2$.*

The constraint satisfaction problem defined by P_H is equivalent to the graph homomorphism problem to the undirected graph shown in Figure 2. Intuitively, P_H is the XOR predicate with a "wildcard" element 2 such that the predicate is also satisfied if some input is 2.

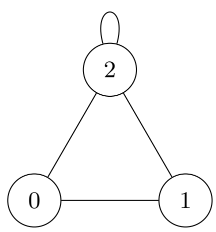

Figure 2 The undirected graph H corresponding to P_H.

We now verify that P_H is preserved by a conservative majority operation.

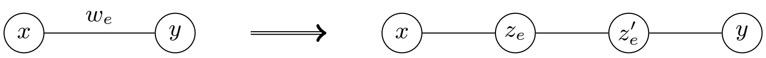

Figure 3 The reduction from MIN UNCUT to MINCOSTCSP(P_H). The nonzero costs are $c(x, 2) = c(y, 2) = 1$, $c(z_e, 2) = c(z'_e, 2) = w_e$.

▷ Claim 42. Let $f : A^3 \to A$ be defined as follows

$$f(a_1, a_2, a_3) = \begin{cases} a & \text{if } |\{i \in [3] \mid a_i = a\}| \geq 2, \\ 2 & \text{otherwise.} \end{cases}$$

Then $f \in \mathsf{Pol}(P_H)$.

Proof. Let $(a_1, b_1), (a_2, b_2), (a_3, b_3) \in P_H$. We verify that $(f(a_1, a_2, a_3), f(b_1, b_2, b_3)) \in P_H$. This is always true if at least one of $f(a_1, a_2, a_3)$ and $f(b_1, b_2, b_3)$ is 2. If neither is 2, then there is a majority in (a_1, a_2, a_3) as well as in (b_1, b_2, b_3). It follows by the pigeonhold principle that there must be some $i \in [3]$ such that a_i is equal to the majority element in (a_1, a_2, a_3) and b_i is equal to the majority element in (b_1, b_2, b_3), so we have $(f(a_1, a_2, a_3), f(b_1, b_2, b_3)) = (a_i, b_i) \in P_H$. ◁

Note that f is conservative since when there isn't a majority we must have $\{a_1, a_2, a_3\} = \{0, 1, 2\} \ni 2$.

To prove that MINCOSTCSP(P_H) is hard to approximate, we use a reduction from the MIN UNCUT problem.

▶ **Definition 43.** *In the MIN UNCUT problem, the input is a weighted undirected graph $G = (V, E, \{w_e\}_{e \in E})$ where $w_e \geq 0$ for every $e \in E$, and we are asked to remove a subset $E' \subseteq E$ of the edges such that the remaining graph $G' = (V, E \setminus E')$ is bipartite. The goal is to minimize the total weight of removed edges $\sum_{e \in E'} w_e$.*

We use $\mathrm{Opt}(G)$ to denote the value of an optimum solution to MIN UNCUT(G). Without loss of generality, we may assume that the total edge weight in a MIN UNCUT instance is normalized to be 1, i.e., $\sum_{e \in E} w_e = 1$.

▶ **Theorem 44** ([32]). *Assuming UGC, there exists some constant $c > 0$ such that for all sufficiently small $\epsilon > 0$ it is NP-hard to distinguish instances of MIN UNCUT with value at most ϵ and instances with value at least $c \cdot \sqrt{\epsilon}$. In particular, it is NP-hard to approximate MIN UNCUT within any constant factor, assuming UGC.*

▶ **Theorem 45.** *Assuming UGC, it is NP-hard to approximate MINCOSTCSP(P_H) within any constant factor.*

Proof. Given any MIN UNCUT instance $G = (V, E, \{w_e\}_{e \in E})$, we construct an instance I of MINCOSTCSP(P_H) such that $\mathrm{Opt}(G) = \mathrm{Opt}(I)$. This reduction combined with Theorem 44 will establish our theorem. The reduction is as follows. The variable set of I will be $V \cup \{z_e, z'_e \mid e \in E\}$ where we take vertices in G plus two distince auxiliary variables z_e, z'_e for every edge $e \in E$. For every $e = \{x, y\} \in E$, we add three constraints $P_H(x, z_e), P_H(z_e, z'_e), P_H(z'_e, y)$ to I (note that the order of x and y does not matter). For the cost function c, we define $c(x, 0) = c(x, 1) = 0$, $c(x, 2) = 1$ for every $x \in V$, and $c(z_e, 0) = c(z_e, 1) = c(z'_e, 0) = c(z'_e, 1) = 0$, $c(z_e, 2) = c(z'_e, 2) = w_e$ for the auxiliary variables. This completes the construction. See Figure 3 for an illustration.

We claim that $\mathrm{Opt}(G) = \mathrm{Opt}(I)$. We first show that $\mathrm{Opt}(G) \geq \mathrm{Opt}(I)$. Take any optimal assignment for G. We take the same assignment for the vertex variables in I which generates no cost. For any edge $e = \{x, y\}$ that is satisfied by the assignment, we can set $z_e = 1 - x$,

$z'_e = 1 - y$ and satisfy all three constraints $P_H(x, z_e), P_H(z_e, z'_e), P_H(z'_e, y)$ with no cost. For any edge $e = \{x, y\}$ that is not satisfied, we can set $z_e = 2$ and $z'_e = 1 - y$, satisfying all three constraints $P_H(x, z_e), P_H(z_e, z'_e), P_H(z'_e, y)$ with cost w_e. So we obtain an assignment for I that has value $\mathrm{Opt}(G)$.

The other direction can be shown similarly. First observe that for each edge e we may assume at most one of the two auxiliary variables z_e, z'_e is set to 2. Also, since for any vertex variable x we have $c(x, 2) = 1 = \sum_{e \in E} w_e$, we may assume that no vertex variable x is set to 2. Now if we take an optimal assignment A for I with these two assumptions, A restricted to the vertex variables is a valid assignment for G whose the total weight of violated edges is at most the cost of A, which implies that $\mathrm{Opt}(G) \le \mathrm{Opt}(I)$. ◀

5 Application: Dichotomy for MinCostCSP with permutation constraints

As an application of our results, we give a complete classification for MinCostCSP(Γ) where Γ contains all *permutation relations*.

▶ **Definition 46.** *A binary relation $R \subseteq D^2$ over D is called a* permutation relation *if $R = \{(a, \sigma(a)) \mid a \in D\}$ for some bijective $\sigma : D \to D$.*

▶ **Theorem 47** (Theorem 3 restated)**.** *Let Γ be a set of relations over D such that it contains all permutation relations. Then MinCostCSP(Γ) is $|D|$-approximable if Γ is preserved by a conservative majority operation. Otherwise, if Γ is not preserved by any conservative majority operation, then it is also not preserved by any conservative NU operation and MinCostCSP(Γ) is not constant-factor approximable, assuming P $\neq$ NP.*

A constraint language that contains all permutation relations can be seen as a natural generalization of Boolean constraint languages that allow negation of variables. Our classification relies on the classification of *homogeneous algebras*. To state the result, we first need some definitions.

▶ **Definition 48.** *An algebra $(D, \mathcal{F})$ consists of a set D (called the* universe*) and a set of operations $\mathcal{F}$ (called the* basic operations*) which are functions from finite powers of D to D. The symbols and arities of the basic operations are called the* signature *of $(D, \mathcal{F})$. A* term operation *is an operation obtained by composition of operations in $\mathcal{F}$.*

The set of all term operations of a given algebra $(D, \mathcal{F})$ form a clone (recall Definition 9). We denote this clone by $\langle \mathcal{F} \rangle$. When $\mathcal{F} = \{s_1, \ldots, s_k\}$ consists of finitely many operations, we may also write $\langle s_1, \ldots, s_k \rangle$ in place of $\langle \mathcal{F} \rangle$.

▶ **Definition 49.** *Let $(D, \mathcal{F})$ and $(D', \mathcal{F}')$ be two algebras with the same signature. A function $f : D \to D'$ is called a homomorphism from $(D, \mathcal{F})$ to $(D', \mathcal{F}')$, if f commutes with all basic operations. That is, for every k-ary function symbol t in the signature, we have $t_{D'}(f(a_1), \ldots, f(a_k)) = f(t_D(a_1, \ldots, a_k))$, where t_D and $t_{D'}$ are the functions t represents in $(D, \mathcal{F})$ and $(D', \mathcal{F}')$ respectively. When $(D, \mathcal{F}) = (D', \mathcal{F}')$, we also say that f is an* automorphism*.*

▶ **Definition 50.** *An algebra $(D, \mathcal{F})$ is called a homogeneous algebra if every bijection $D \to D$ is an automorphism.*

The following claim follows directly from the definition.

▷ Claim 51. Let Γ be a constraint language which contains all permutation relations. Then $(D, \mathsf{Pol}(\Gamma))$ is a homogeneous algebra.

The study of homogenous algebras was initiated by Marczewski [37], and a complete classification was first obtained by Marchenkov [36]. Dalmau used Marchenkov's result to give a complete classification for decision CSP where the constraint language contains all permutation relations [15]. The following theorem is taken from [43] (see also [15]).

▶ **Theorem 52** (Theorem 5.9 in [43]). *Let D be a finite domain such that $|D| \geq 5$. Let $(D, \mathcal{F})$ be a homogeneous algebra, then either the dual discriminator operation d is a term operation, or its clone of term operations $\langle \mathcal{F} \rangle$ is equal to one of the followings:*

- $E_1^0 = \langle s \rangle$, $E_1^1 = \langle s, r_n \rangle$,
- $E_i^0 = \langle l_i \rangle$ *for* $2 \leq i \leq n-1$, $E_n^0 = \mathcal{J}$.
- $E_i^1 = \langle l_i, r_n \rangle$ *for* $2 \leq i \leq n-3$, $E_{n-2}^1 = \langle r_n \rangle$.

Here $\mathcal{J}$ is the clone of projection operations. s is the switching *operation, defined by*

$$s(x_1, x_2, x_3) = \begin{cases} x_3 & \text{if } x_1 = x_2, \\ x_2 & \text{if } x_1 = x_3, \\ x_1 & \text{otherwise.} \end{cases}$$

For $2 \leq k \leq n-1$, l_k is the k-ary near projection *operation defined by*

$$l_k(x_1, x_2, \ldots, x_k) = \begin{cases} x_1 & \text{if } |\{x_1, \ldots, x_k\}| < k, \\ x_k & \text{otherwise.} \end{cases}$$

And finally, r_n is the $(n-1)$-ary operation defined by

$$r_n(x_1, x_2, \ldots, x_{n-1}) = \begin{cases} x_1 & \text{if } |\{x_1, \ldots, x_{n-1}\}| < n-1, \\ x_n & \text{otherwise, where } x_n \in D \backslash \{x_1, \ldots, x_{n-1}\}. \end{cases}$$

The $|D| \geq 5$ assumption is not essential. When $|D| \geq 5$, the above algebras are all distinct. When $2 \leq |D| \leq 4$, some of these algebras become non-distinct, but the only exceptional case not covered by the classification above is the Klein 4-group (the unique 4-element group with exponent 2) with the operation $x + y + z$. However, this is not a conservative operation so we may safely ignore it for our purpose.

We are now ready to prove Theorem 47.

Proof of Theorem 47. First observe that if $\mathsf{Pol}(\Gamma)$ contains some majority function f, then there must be some $i \in [3]$ such that $f(x_1, x_2, x_3) = x_i$ when x_1, x_2, x_3 are pairwise distinct: if not, then there exist distinct $i, j \in [3]$ and two triples $(x_1, x_2, x_3), (y_1, y_2, y_3)$ with pairwise distinct elements within each triple such that $f(x_1, x_2, x_3) = x_i, f(y_1, y_2, y_3) = y_j$. Then, let $\pi : D \to D$ be a permutation such that $y_i = \pi(x_i)$ for every $i \in [3]$, f does not preserve the permutation relation $\{(a, \pi(a)) \mid a \in D\}$, which is a contradiction. By potentially permuting the input coordinates in f, we get that the dual discriminator operation d is also contained in $\mathsf{Pol}(\Gamma)$, and therefore $\textsc{MinCostCSP}(\Gamma)$ is $|D|$-approximable by Theorem 1.

Now suppose $\mathsf{Pol}(\Gamma)$ does not contain a conservative majority function, then in particular it does not contain the dual discriminator operation. Note that since Γ can be assumed to contain all unary operations (see Observation 7), every polymorphism of Γ is conservative. It is easy to see that r_n is not conservative. If $\mathsf{Pol}(\Gamma) = E_n^0 = \mathcal{J}$, then $\mathrm{CSP}(\Gamma)$ is NP-complete. Furthermore, Dalmau [15] showed that if $\mathsf{Pol}(\Gamma) = E_i^0$ for some $2 \leq i \leq n-1$, then $\mathrm{CSP}(\Gamma)$ is also NP-complete. So by Theorem 52, the only remaining possibility is $\mathsf{Pol}(\Gamma) = \langle s \rangle$. However, as is observed by Dalmau [15], $\langle s \rangle$ does not contain any NU operation, so by Theorem 26, there is no constant factor approximation for $\textsc{MinCostCSP}(\Gamma)$, assuming $\mathrm{P} \neq \mathrm{NP}$. ◀

References

1 Kirby A Baker and Alden F Pixley. Polynomial interpolation and the chinese remainder theorem for algebraic systems. *Mathematische Zeitschrift*, 143:165–174, 1975.

2 Nikhil Bansal and Subhash Khot. Inapproximability of hypergraph vertex cover and applications to scheduling problems. In *International Colloquium on Automata, Languages, and Programming*, pages 250–261. Springer, 2010. doi:10.1007/978-3-642-14165-2_22.

3 Libor Barto. The dichotomy for conservative constraint satisfaction problems revisited. In *2011 IEEE 26th Annual Symposium on Logic in Computer Science*, pages 301–310. IEEE, 2011. doi:10.1109/LICS.2011.25.

4 Libor Barto. The collapse of the bounded width hierarchy. *Journal of Logic and Computation*, 26(3):923–943, 2014.

5 Libor Barto and Marcin Kozik. Constraint satisfaction problems solvable by local consistency methods. *Journal of the ACM (JACM)*, 61(1):1–19, 2014. doi:10.1145/2556646.

6 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 für Informatik, Dagstuhl, Germany, 2017. doi:10.4230/DFU.Vol7.15301.1.

7 VG Bondarchuk, LA Kaluzhnin, VN Kotov, and BA Romov. Galois theory for post algebras. i-ii. *Kibernetika*, 3:1–10, 1969.

8 Joshua Brakensiek, Neng Huang, Aaron Potechin, and Uri Zwick. On the mysteries of max nae-sat. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 484–503. SIAM, 2021. doi:10.1137/1.9781611976465.30.

9 Joshua Brakensiek, Neng Huang, Aaron Potechin, and Uri Zwick. Separating max 2-and, max di-cut and max cut. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 234–252. IEEE, 2023. doi:10.1109/FOCS57990.2023.00023.

10 Andrei A. Bulatov. Tractable conservative constraint satisfaction problems. In *18th Annual IEEE Symposium of Logic in Computer Science, 2003. Proceedings.*, pages 321–330. IEEE, 2003.

11 Andrei A. Bulatov. Conservative constraint satisfaction re-revisited. *Journal of Computer and System Sciences*, 82(2):347–356, 2016. doi:10.1016/J.JCSS.2015.07.004.

12 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. doi:10.1109/FOCS.2017.37.

13 Qi Cheng and Daqing Wan. A deterministic reduction for the gap minimum distance problem. *IEEE Transactions on Information Theory*, 58(11):6935–6941, 2012. doi:10.1109/TIT.2012.2209198.

14 Martin C. Cooper, David A. Cohen, and Peter G. Jeavons. Characterising tractable constraints. *Artificial Intelligence*, 65(2):347–361, 1994. doi:10.1016/0004-3702(94)90021-3.

15 Victor Dalmau. A new tractable class of constraint satisfaction problems. *Annals of Mathematics and Artificial Intelligence*, 44:61–85, 2005. doi:10.1007/S10472-005-1810-9.

16 Víctor Dalmau, Marcin Kozik, Andrei Krokhin, Konstantin Makarychev, Yury Makarychev, and Jakub Oprsal. Robust algorithms with polynomial loss for near-unanimity csps. *SIAM Journal on Computing*, 48(6):1763–1795, 2019. doi:10.1137/18M1163932.

17 Víctor Dalmau and Andrei Krokhin. Robust satisfiability for csps: Hardness and algorithmic results. *ACM Transactions on Computation Theory (TOCT)*, 5(4):1–25, 2013. doi:10.1145/2540090.

18 Víctor Dalmau, Andrei Krokhin, and Rajsekar Manokaran. Towards a characterization of constant-factor approximable finite-valued csps. *Journal of Computer and System Sciences*, 97:14–27, 2018. doi:10.1016/J.JCSS.2018.03.003.

19 Irit Dinur, Venkatesan Guruswami, Subhash Khot, and Oded Regev. A new multilayered pcp and the hardness of hypergraph vertex cover. *SIAM Journal on Computing*, 34(5):1129–1146, 2005. doi:10.1137/S0097539704443057.

20 I Dumer, D. Micciancio, and M. Sudan. Hardness of approximating the minimum distance of a linear code. *IEEE Transactions on Information Theory*, 49(1):22–37, 2003. doi:10.1109/TIT.2002.806118.

21 Alina Ene, Jan Vondrak, and Yi Wu. Local distribution and the symmetry gap: Approximability of multiway partitioning problems. *arXiv preprint*, 2015. arXiv:1503.03905.

22 Tomás Feder and Moshe Y Vardi. The computational structure of monotone monadic snp and constraint satisfaction: A study through datalog and group theory. *SIAM Journal on Computing*, 28(1):57–104, 1998. doi:10.1137/S0097539794266766.

23 David Geiger. Closed systems of functions and predicates. *Pacific journal of mathematics*, 27(1):95–100, 1968.

24 Dan Gusfield and Leonard Pitt. A bounded approximation for the minimum cost 2-sat problem. *Algorithmica*, 8(1):103–117, 1992. doi:10.1007/BF01758838.

25 Gregory Gutin, Pavol Hell, Arash Rafiey, and Anders Yeo. A dichotomy for minimum cost graph homomorphisms. *European Journal of Combinatorics*, 29(4):900–911, 2008. doi:10.1016/J.EJC.2007.11.012.

26 Pavol Hell, Monaldo Mastrolilli, Mayssam Mohammadi Nevisi, and Arash Rafiey. Approximation of minimum cost homomorphisms. In *Algorithms–ESA 2012: 20th Annual European Symposium, Ljubljana, Slovenia, September 10-12, 2012. Proceedings 20*, pages 587–598. Springer, 2012. doi:10.1007/978-3-642-33090-2_51.

27 Pavol Hell and Arash Rafiey. The dichotomy of minimum cost homomorphism problems for digraphs. *SIAM Journal on Discrete Mathematics*, 26(4):1597–1608, 2012. doi:10.1137/100783856.

28 Peter Jeavons, David Cohen, and Martin C Cooper. Constraints, consistency and closure. *Artificial Intelligence*, 101(1-2):251–265, 1998. doi:10.1016/S0004-3702(98)00022-8.

29 Peter Jonsson and Gustav Nordh. Introduction to the maximum solution problem. *Complexity of Constraints: An Overview of Current Research Themes*, pages 255–282, 2008. doi:10.1007/978-3-540-92800-3_10.

30 Sanjeev Khanna, Madhu Sudan, Luca Trevisan, and David P Williamson. The approximability of constraint satisfaction problems. *SIAM Journal on Computing*, 30(6):1863–1920, 2001. doi:10.1137/S0097539799349948.

31 Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 767–775, 2002. doi:10.1145/509907.510017.

32 Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for MAX-CUT and other 2-variable CSPs? *SIAM Journal on Computing*, 37(1):319–357, 2007. doi:10.1137/S0097539705447372.

33 Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within 2- ε. *Journal of Computer and System Sciences*, 74(3):335–349, 2008.

34 Andrei Krokhin and Stanislav Zivny. The Complexity of Valued CSPs. In Andrei Krokhin and Stanislav Zivny, editors, *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. doi:10.4230/DFU.Vol7.15301.233.

35 Amit Kumar, Rajsekar Manokaran, Madhur Tulsiani, and Nisheeth K Vishnoi. On lp-based approximability for strict csps. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1560–1573. SIAM, 2011. doi:10.1137/1.9781611973082.121.

36 S.S. Marchenkov. Homogeneous algebras. *Problemy Kibernetiki*, 39:85–106, 1982.

37 E. Marczewski. Homogeneous algebras and homogeneous operations. *Fund. Math*, 56(8):103, 1964.

38 E.L. Post. The two-valued iterative systems of mathematical logic. *Annals of Mathematics Studies*, 1941.

39 Akbar Rafiey, Arash Rafiey, and Thiago Santos. Toward a dichotomy for approximation of H-coloring. In *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, pages 91:1–91:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ICALP.2019.91.

40 Prasad Raghavendra. Optimal algorithms and inapproximability results for every csp? In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 245–254, 2008. doi:10.1145/1374376.1374414.

41 Thomas J Schaefer. The complexity of satisfiability problems. In *Proceedings of the tenth annual ACM symposium on Theory of computing*, pages 216–226, 1978. doi:10.1145/800133.804350.

42 Hanif D Sherali and Warren P Adams. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM Journal on Discrete Mathematics*, 3(3):411–430, 1990. doi:10.1137/0403036.

43 Ágnes Szendrei. *Clones in universal algebra*. Presses de l'Université de Montréal, 1986.

44 Rustem Takhanov. A dichotomy theorem for the general minimum cost homomorphism problem. In *27th International Symposium on Theoretical Aspects of Computer Science (2010)*, pages 657–668. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2010. doi:10.4230/LIPICS.STACS.2010.2493.

45 Johan Thapper and Stanislav Živný. The complexity of finite-valued csps. *Journal of the ACM (JACM)*, 63(4):1–33, 2016. doi:10.1145/2974019.

46 Dmitriy Zhuk. A proof of the csp dichotomy conjecture. *Journal of the ACM (JACM)*, 67(5):1–78, 2020. doi:10.1145/3402029.