CLAP: A New Algorithm for Promise CSPs*†

Lorenzo Ciardo‡ Stanislav Živný§

Abstract

We propose a new algorithm for Promise Constraint Satisfaction Problems (PCSPs). It is a combination of the **C**onstraint Basic **L**P relaxation and the **A**ffine I**P** relaxation (CLAP). We give a characterisation of the power of CLAP in terms of a minion homomorphism. Using this characterisation, we identify a certain weak notion of symmetry which, if satisfied by infinitely many polymorphisms of PCSPs, guarantees tractability.

We demonstrate that there are PCSPs solved by CLAP that are not solved by any of the existing algorithms for PCSPs; in particular, not by the BLP + AIP algorithm of Brakensiek and Guruswami [SODA'20] and not by a reduction to tractable finite-domain CSPs.

1 Introduction

Constraint Satisfaction Constraint Satisfaction Problems (CSPs) have driven some of the most influential developments in theoretical computer science, from NP-completeness to the PCP theorem [2, 1, 36] to semidefinite programming algorithms [58] to the Unique Games Conjecture [49].

A CSP over domain A is specified by a finite collection $\mathbf{A}$ of relations over A, and is denoted by CSP($\mathbf{A}$). Given on input a set of variables and a set of constraints, each of which uses relations from $\mathbf{A}$, the task is to decide the existence of an assignment of values from A to the variables that satisfies all the constraints. Classic examples of CSPs include 2-SAT, graph 3-colouring, and linear equations of fixed width over finite groups.

For Boolean CSPs, which are CSPs with $|A| = 2$, Schaefer proved that every such CSP is either solvable in polynomial time or is NP-complete [59]. Feder and Vardi famously conjectured that the same holds true for CSPs over arbitrary finite domains [38]. Furthermore, they realised the importance of considering closure properties of solution spaces of CSPs [38], which initiated the algebraic approach [46, 45, 26]. The key notion in the algebraic approach is that of *polymorphisms*, which are operations that take solutions to a CSP and are guaranteed to return, by a coordinatewise application, a solution to the same CSP. All CSPs admit projections (also known as dictators) as polymorphisms. However, the presence of less trivial polymorphisms, satisfying some notion of symmetry, is necessary for tractability. For instance, the set of solutions to 2-SAT is closed under the ternary majority operation $\mathrm{maj} : \{0,1\}^3 \to \{0,1\}$ that satisfies the following notion of symmetry: $\mathrm{maj}(a,a,b) = \mathrm{maj}(a,b,a) = \mathrm{maj}(b,a,a) = a$ for any $a, b \in \{0,1\}$. Similarly, the set of solutions to Horn-SAT is closed under the binary minimum operation $\min : \{0,1\}^2 \to \{0,1\}$ that satisfies a different notion of symmetry: $\min(a,a) = a$, $\min(a,b) = \min(b,a)$, and $\min(a, \min(b,c)) = \min(\min(a,b), c)$ for any $a, b, c \in \{0,1\}$. The binary max operation – which is a polymorphism of dual Horn-SAT – has the same notion of symmetry, called semilattice [10]. Together with the ternary minority polymorphism, which captures linear equations on $\{0,1\}$, this gives all non-trivial tractable cases from Schaefer's dichotomy result.[1]

The polymorphisms of any CSP form a clone, in that they include all projections and are closed under composition. For instance, since Horn-SAT has min as a polymorphism, it also has the 4-ary minimum operation $\min_4(a,b,c,d) = \min(a, \min(b, \min(c,d)))$ as a polymorphism. Building on the connection to universal algebra, the algebraic approach has been tremendously successful beyond decision CSPs, e.g. for robust satisfiability of CSPs [34, 9, 33], for exact optimisation of CSPs [53, 60, 50], and for characterising the power of

*The research leading to these results has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 714532). Stanislav Živný was supported by a Royal Society University Research Fellowship. The paper reflects only the authors' views and not the views of the ERC or the European Commission. The European Union is not liable for any use that may be made of the information contained therein.

†The full version of the paper can be accessed at `https://arxiv.org/abs/2107.05018`.

‡Department of Computer Science, University of Oxford, UK.

§Department of Computer Science, University of Oxford, UK.

[1]The trivial cases, called 0- and 1-valid, are captured by the constant-0 and constant-1 polymorphisms, respectively.

algorithms [55, 8, 13, 51, 52, 61, 62]. The culmination of the algebraic approach is the positive resolution of the dichotomy conjecture by Bulatov [28] and Zhuk [65]. We refer the reader to [10] for a survey on the algebraic approach.

Promise Constraint Satisfaction In this paper, we study Promise Constraint Satisfaction Problems (PCSPs), whose systematic study was initiated by Austrin, Guruswami, and Håstad [4], and Brakensiek and Guruswami [18]. PCSPs form a vast generalisation of CSPs. In PCSP($\mathbf{A}, \mathbf{B}$), each constraint comes in two forms, a strict one in $\mathbf{A}$ and a weak one in $\mathbf{B}$. The goal is to distinguish between (i) the case in which (the strong form of) the constraints can be simultaneously satisfied in $\mathbf{A}$ and (ii) the case in which (even the weak form of) the constraints cannot be simultaneously satisfied in $\mathbf{B}$. The promise is that it is never the case that the PCSP is not satisfiable in the strict sense but is satisfiable in the weak sense. If the strict and weak forms coincide in every constraint (i.e., if $\mathbf{A} = \mathbf{B}$) we get the (non-promise) CSPs. However, PCSPs include many fundamental problems that are inexpressible as CSPs.

The simplest example of strict vs. weak constraints is when the weak constraints are supersets of the strict constraints on the same domain (the first two examples below) or on a larger domain (the third example below); the notion of homomorphism from $\mathbf{A}$ to $\mathbf{B}$ formalises this for any PCSP.

First, can we distinguish a g-satisfiable k-SAT instance (in the sense that there is an assignment that satisfies at least g literals in each clause) from an instance that is not even 1-satisfiable? This problem was studied in [4], where it was shown to be solvable in polynomial time if $\frac{g}{k} \geq \frac{1}{2}$ and NP-complete otherwise. Recently, this result has been generalised to arbitrary finite domains [23].

Second, can we distinguish a 3-SAT formula that admits an assignment satisfying exactly 1 literal in each clause (i.e., a satisfiable instance of **1-in-3**-SAT) from one that does not admit an assignment satisfying 1 or 2 literals in each clause (i.e., a non-satisfiable instance of **N**ot-**A**ll-**E**qual-3-SAT)? Remarkably, while both **1-in-3** and **NAE** are NP-hard, this promise version is solvable in polynomial time [18, 19].

Third, can we distinguish a k-colourable graph from a graph that is not even ℓ-colourable, where $k \leq \ell$? This is the *approximate graph colouring* problem, which is believed to be NP-hard for any fixed $3 \leq k \leq \ell$, but has been elusive since the 1970s [40]. The current state of the art is NP-hardness for $k = 3$ and $\ell = 5$ [7] and for any $k \geq 4$ and $\ell = \ell(k) = \binom{k}{\lfloor k \rfloor / 2} - 1$ [64].

While a systematic study of PCSPs was initiated only recently [4, 18], concrete PCSPs have been considered for a while, e.g. approximate graph [40, 63, 15, 47, 48, 41] and hypergraph colouring [37]. A highlight result is the dichotomy of Boolean symmetric PCSPs [39] (in which all constraint relations are symmetric), following an earlier classification of Boolean symmetric PCSPs with disequalities [18]. Very recent works have investigated certain Boolean non-symmetric PCSPs [24] and certain non-Boolean symmetric PCSPs [6]. Other recent results include, e.g., [3, 42, 21].

Most of the recent progress, including results on the approximate graph colouring problem [7, 64] and on the approximate graph homomorphism problem [54, 64], rely on the algebraic approach to PCSPs [7]. In particular, the breakthrough results in [7], building on [11], established that the complexity of PCSPs is captured by the polymorphism minions and certain types of symmetries these minions satisfy – these are non-nested identities on polymorphisms, such as the majority example but not the semilattice example. Crucially, minions are less structured than clones: A minion (of functions) is a set of operations closed under permuting coordinates, identifying coordinates, and introducing dummy coordinates, but not under composition.[2] Thus, unlike in our earlier CSP example (corresponding to Horn-SAT), a binary minimum polymorphism of a PCSP cannot in general be used to generate a 4-ary minimum polymorphism of the same PCSP.

Despite the momentous results in [7], there is a long way to go to classify all PCSPs, and it is not even clear whether a dichotomy for all PCSPs should be expected. When Feder and Vardi conjectured a CSP dichotomy [38], the Boolean case [59] and the graph case [43] had been fully classified. We seem quite far from these two cases being classified for PCSPs. Thus, further progress is needed on both the hardness and tractability part. This paper focusses on the latter.

Finite tractability Although PCSPs are (much) more general than CSPs, some PCSPs can be reduced to tractable CSPs. This idea was introduced in [19] under the name of homomorphic sandwiching (cf. Section 2 for a precise definition); PCSPs that are reducible to tractable (finite-domain) CSPs are called *finitely tractable*. Finite tractability is not sufficient to explain tractability of all tractable PCSPs. In particular, Barto et al. showed [7]

[2] In this work, we shall use the more abstract notion of minion introduced in [22], cf. Definition 2.3.

that the above-mentioned example **1-in-3** vs. **NAE** is not finitely tractable, despite being a tractable PCSP [18]. We remark that it is not inconceivable (and in fact was conjectured in [19]) that every tractable (finite-domain) PCSP could be reducible to a tractable CSP possibly over an infinite domain; this is the case for the **1-in-3** vs. **NAE** problem [19]. However, while certain infinite-domain CSPs are amenable to algebraic methods, the complexity of infinite-domain CSPs is far from understood, cf. [16, 17, 12] for recent work.

Since finite tractability does not capture all tractable PCSPs, there is need for other algorithmic tools. One possibility is to attempt to extend algorithmic techniques developed for CSPs.

There are two main algorithmic approaches for CSPs. On the one hand, there are local consistency methods [38], which have been studied in theoretical computer science but also in artificial intelligence, logic, and database theory. The power of local consistency for CSPs has been characterised in [25, 8], and it is known that the third level of consistency solves all so-called bounded-width CSPs [5]. On the other hand, there are CSPs solvable by algorithms based on generalisations of Gaussian elimination, most notably CSPs with a Mal'tsev polymorphism [29]. This method has been pushed to its limit, in a way, in [44, 13]. While the NP-hardness part of the CSP dichotomy has been known since [26], the challenge in proving the algorithmic part is the complicated interaction of these two very different algorithmic approaches. Although this interaction does not occur in Boolean CSPs, it occurs already in CSPs on three-element domains [27].

The characterisation of the power of the first level of the consistency methods, 1-consistency (also known as arc-consistency [57]), has been lifted from CSPs [38] to PCSPs in [7]. Rather than establishing 1-consistency combinatorially, one can employ convex relaxations.

Relaxations A canonical analogue of 1-consistency is the *basic linear programming relaxation* (BLP) [55], which in fact is stronger than 1-consistency [56]. The characterisation of the power of BLP has been lifted from CSPs [55] to PCSPs in [7], both in terms of a minion and a property of polymorphisms. The power of BLP is captured by a minion consisting of rational stochastic vectors or, equivalently, by the presence of symmetric polymorphisms of all arities; these are polymorphisms invariant under any permutation of the coordinates. For example, we have seen that Horn-SAT, a classic CSP, has a binary symmetric polymorphism, namely min. We have also seen that min can generate a 4-ary operation $\min_4$, which is symmetric. Similarly, min can generate (via composition) symmetric operations of all arities, and thus Horn-SAT is solved by BLP.

A different relaxation of PCSPs is the *basic affine integer programming relaxation* (AIP) [19]. The power of AIP has been characterised, both in terms of a minion and a property of polymorphisms, in [7]. The minion capturing AIP consists of integer affine vectors. Concerning polymorphisms, AIP is captured by polymorphisms of all odd arities that are invariant under permutations that only permute odd and even coordinates separately, and additionally satisfy that adjacent coordinates cancel each other out. The **1-in-3** vs. **NAE** problem is solved by AIP (cf. Example 2.3).

Brakensiek and Guruswami [20] proposed a combination of the two above-mentioned relaxations, called BLP + AIP. Their algorithm has many interesting features. Firstly, it solves PCSPs that admit only infinitely many symmetric polymorphisms (i.e., not all arities are required as in the case of BLP). Secondly, it solves all tractable Boolean CSPs, thus demonstrating how research on PCSPs can shed new light on (non-promise) CSPs. Indeed, although there are no Boolean CSPs that mix bounded width and linear equations, there previously had not been a single algorithm solving both cases. The BLP + AIP algorithm does that. A follow-up work [22] established the power of BLP + AIP in terms of a minion and (a property of) polymorphisms. The minion capturing BLP + AIP is essentially a product of the BLP and AIP minions [22]. Concerning polymorphisms, BLP + AIP is captured by polymorphisms of all odd arities that are invariant under permutations that only permute odd and even coordinates.

At present, all known tractable PCSPs are solved either by BLP + AIP or are finitely tractable. It may well be that BLP + AIP is the only algorithm needed for all tractable Boolean PCSPs. However, as already observed in [22], BLP + AIP does not solve some rather simple, tractable PCSPs. Motivated by this, we investigate algorithms that are stronger than BLP + AIP.

Contributions Building on the work of Brakensiek et al. [22], we study stronger relaxations for PCSPs and give three main contributions.

(1) CLAP Our first contribution is the introduction of CLAP to the study of PCSPs. Our goal was to design an algorithm that, unlike BLP + AIP, solves all CSPs of bounded width. While all bounded-width CSPs can be solved by 3-consistency [5], and thus also by the third level of the Sherali-Adams hierarchy for BLP (e.g., by [61]), Kozik showed that already (a special case of) the singleton arc-consistency (SAC) algorithm, introduced in [35]

(cf. [14, 30]), solves all bounded-width CSPs [52]. Thus, we study the LP relaxation that we call the *singleton* BLP (SBLP), which is at least as strong as SAC. A special case of SBLP (without this name) implicitly appeared in the literature, e.g. in [4, 18] for Boolean PCSPs. The idea behind SBLP is essentially to run SAC but replace the arc-consistency check by the BLP; i.e., the algorithm repeatedly takes a variable-value pair (x, a) and tests the feasibility of the BLP with the requirement that x should be assigned the value a. If this LP is infeasible then a is removed from the domain of x. This is repeated until convergence. If any variable ends up with an empty domain then SBLP rejects, otherwise it accepts. Overall, the number of BLP calls occurring for an instance of $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ with variable-set X is at most polynomial in the size of X. As mentioned above, this simple algorithm solves all bounded-width CSPs [52].

We adopt a modification of SBLP that turns out to be more naturally captured by a minion-oriented analysis: the *constraint* BLP (CBLP). This (possibly) stronger algorithm is a generalisation of SBLP in which we do not consider only variable-value pairs (x, a), but rather the constraint-assignment pairs $(\mathbf{x}, \mathbf{a})$ for every constraint in the instance. As in SBLP, if fixing a (local) assignment to a constraint yields an infeasible BLP then the assignment is removed from the constraint relation. Upon convergence, which takes at most polynomially many BLP calls, if any constraint ends up with an empty relation then CBLP rejects, otherwise it accepts.

Our algorithm CLAP first runs CBLP and then runs AIP upon termination of CBLP. If one believes the suggestion in [22] that constantly many rounds of the Sherali-Adams hierarchy for BLP + AIP could solve all tractable (non-promise) CSPs, then it is not outrageous to believe that the same could be true for CLAP, and CLAP might be easier to analyse.

(2) Characterisation Our second contribution is a minion characterisation of the power of CLAP, stated as Theorem 3.1. The objects in the minion are essentially matrices with a particular structure, which we call skeletal (cf. Definition 3.1). These matrices capture the CBLP part of CLAP and together with certain integer affine vectors form the minion (cf. Definition 3.2).

(3) H-symmetric polymorphisms Building on the minion characterisation, our third contribution is the identification of a sufficient condition for CLAP to work in terms of the symmetries of the polymorphisms. This is stated as Theorem 3.2, using the notion of H-symmetricity. For a matrix H, a polymorphism f is H-symmetric if f is invariant under permutations of the coordinates but only on a specific set of inputs determined by H (cf. Definition 3.3). For instance, if H is a row vector then we obtain the requirement that f be symmetric on all inputs. If H is the identity matrix then we require that f be symmetric only on inputs in which different entries occur with different multiplicities. In general, the intuition is that we capture "symmetry with exceptions that depend on multiplicities". We refer the reader to the discussion in Section 3 for details. Thanks to the AIP part of CLAP, only infinitely many (as opposed to of all arities) H-symmetric polymorphisms suffice for CLAP to work. The link between CLAP and H-symmetricity lies in the notion of skeletal matrices and in their key "tiebreak property" stated as [31, Lemma 32] in the full version: Finitely many skeletal matrices can be simultaneously reduced to vectors that avoid ties. Finally, we give an example of a PCSP that is neither finitely tractable nor is it solved by BLP + AIP, but is solved by our new algorithm.

After necessary background material in Section 2, our algorithm CLAP and the main results are presented in Section 3. The details of all results and proofs can be found in the full version of this paper [31].

2 Preliminaries

We let $\mathbb{N} = \{1, 2, \ldots\}$ and $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$. The cardinality of $\mathbb{N}$ shall be denoted by $\aleph_0$. For $k \in \mathbb{N}$, $[k]$ denotes the set $\{1, \ldots, k\}$. For a set A, $\mathcal{P}(A)$ denotes the set of all subsets of A. We denote by $\leq_p$ many-one polynomial-time reductions. We shall use standard notation for vectors and matrices. Vectors will be treated as column vectors and whenever convenient identified with the corresponding (row) tuples. Both tuples and vectors will be typed in bold font. We denote by $\mathbf{e}_i$ the i-th standard unit vector of the appropriate size (which will be clear from the context); i.e., $\mathbf{e}_i$ is equal to 1 in the i-th coordinate and 0 elsewhere. We denote by $\mathbf{0}_p$ and by $\mathbf{1}_p$ the all-zero and all-one vector, respectively, of size p; if the size is clear, we occasionally drop the subscript. The *support* of a vector $\mathbf{v} = (v_i)$ of size p is the set $\mathrm{supp}(\mathbf{v}) = \{i \in [p] : v_i \neq 0\}$. I_p denotes the identity matrix of order p, while O denotes an all-zero matrix of suitable size.

Promise CSPs A *signature* σ is a finite set of relation symbols R, each with its arity $\mathrm{ar}(R) \geq 1$. A *relational structure* over a signature σ, or a σ-structure, is a finite universe A, called the *domain* of $\mathbf{A}$, and a relation $R^{\mathbf{A}} \subseteq A^{\mathrm{ar}(R)}$ for each symbol $R \in \sigma$. For two σ-structures $\mathbf{A}$ and $\mathbf{B}$, a mapping $f : A \to B$ is called a

homomorphism from $\mathbf{A}$ to $\mathbf{B}$, denoted by $f : \mathbf{A} \to \mathbf{B}$, if f preserves all relations; that is, for every $R \in \sigma$ and every tuple $\mathbf{a} \in R^{\mathbf{A}}$, we have $f(\mathbf{a}) \in R^{\mathbf{B}}$, where f is applied coordinatewise. The existence of a homomorphism from $\mathbf{A}$ to $\mathbf{B}$ is denoted by $\mathbf{A} \to \mathbf{B}$. A PCSP *template* is a pair $(\mathbf{A}, \mathbf{B})$ of relational structures over the same signature such that $\mathbf{A} \to \mathbf{B}$. Without loss of generality, we will often assume that A, the domain of $\mathbf{A}$, is $[n]$.

DEFINITION 2.1. *Let $(\mathbf{A}, \mathbf{B})$ be a PCSP template. The* decision version *of* $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ *is the following problem: Given as input a relational structure $\mathbf{X}$ over the same signature as $\mathbf{A}$ and $\mathbf{B}$, output* YES *if $\mathbf{X} \to \mathbf{A}$ and* NO *if $\mathbf{X} \not\to \mathbf{B}$. The* search version *of* $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ *is the following problem: Given as input a relational structure $\mathbf{X}$ over the same signature as $\mathbf{A}$ and $\mathbf{B}$ and such that $\mathbf{X} \to \mathbf{A}$, find a homomorphism from $\mathbf{X}$ to $\mathbf{B}$.*

For a relational structure $\mathbf{A}$, the *constraint satisfaction problem* (CSP) with template $\mathbf{A}$ [38], denoted by $\mathrm{CSP}(\mathbf{A})$, is $\mathrm{PCSP}(\mathbf{A}, \mathbf{A})$.

EXAMPLE 2.1. *For $k \geq 2$, let $\mathbf{K}_k$ be the structure with domain $[k]$ and a binary relation $\{(i, j) \mid i \neq j\}$. Then,* $\mathrm{CSP}(\mathbf{K}_k)$ *is the standard graph k-colouring problem. For $k \leq \ell$,* $\mathrm{PCSP}(\mathbf{K}_k, \mathbf{K}_\ell)$ *is the* approximate graph colouring problem *[40]. In the decision version, the task is to decide whether a graph is k-colourable or not even ℓ-colourable. In the search version, given a k-colourable graph G, the task is to find an ℓ-colouring of G. It is widely believed that for any fixed $3 \leq k \leq \ell$,* $\mathrm{PCSP}(\mathbf{K}_k, \mathbf{K}_\ell)$ *is NP-hard; i.e., constantly many colours do not help. The current most general NP-hardness result is known for $k = 3$ and $\ell = 5$ by Bulín, Krokhin, and Opršal [7] and for $k \geq 4$ and $\ell = \ell(k) = \binom{k}{\lfloor k \rfloor / 2} - 1$ by Wrochna and Živný [64].*

We call a PCSP template $(\mathbf{A}, \mathbf{B})$ *tractable* if any instance of $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ can be solved in polynomial time in the size of the input structure $\mathbf{X}$. It is easy to show that the decision version reduces to the search version [7] (but the converse is not known in general); for CSPs, the two versions are equivalent [32, 26]. Our results are for the decision version.

EXAMPLE 2.2. *Let* **1-*in*-3** *be the Boolean structure with domain $\{0, 1\}$ and a single ternary relation $\{(0, 0, 1), (0, 1, 0), (1, 0, 0)\}$. Let* **NAE** *be the structure with domain $\{0, 1\}$ and a single ternary relation $\{0, 1\}^3 \setminus \{(0, 0, 0), (1, 1, 1)\}$. Then,* $\mathrm{CSP}(\mathbf{1\text{-}in\text{-}3})$ *is the (positive)* 1-*in*-3-*SAT problem and* $\mathrm{CSP}(\mathbf{NAE})$ *is the (positive) Not-All-Equal-*3-*SAT problem. Since both of these problems are NP-hard [59], the PCSP templates* $(\mathbf{1\text{-}in\text{-}3}, \mathbf{1\text{-}in\text{-}3})$ *and* $(\mathbf{NAE}, \mathbf{NAE})$ *are both intractable. However, the PCSP template* $(\mathbf{1\text{-}in\text{-}3}, \mathbf{NAE})$ *is tractable, as shown by Brakensiek and Guruswami [18].*

DEFINITION 2.2. *Let $(\mathbf{A}, \mathbf{B})$ be a PCSP template with signature σ. An operation $f : A^L \to B$, where $L \geq 1$, is a* polymorphism *of arity L of $(\mathbf{A}, \mathbf{B})$ if for every $R \in \sigma$ of arity $k = \mathrm{ar}(R)$ and for any possible $L \times k$ matrix whose rows are tuples in $R^{\mathbf{A}}$, the application of f on the columns of the matrix gives a tuple in $R^{\mathbf{B}}$. We denote by* $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ *the set of all polymorphisms of $(\mathbf{A}, \mathbf{B})$.*

EXAMPLE 2.3. *The unary operation $\neg : \{0, 1\} \to \{0, 1\}$ defined by $\neg(a) = 1 - a$ is a polymorphism of $(\mathbf{NAE}, \mathbf{NAE})$ but not a polymorphism of $(\mathbf{1\text{-}in\text{-}3}, \mathbf{1\text{-}in\text{-}3})$. For any odd L, the L-ary operation $f : \{0, 1\}^L \to \{0, 1\}$ defined by $f(a_1, \ldots, a_L) = 1$ if $a_1 - a_2 + a_3 - \cdots + a_L > 0$ and $f(a_1, \ldots, a_L) = 0$ otherwise is a polymorphism of $(\mathbf{1\text{-}in\text{-}3}, \mathbf{NAE})$.*

Minions Polymorphisms of CSPs form clones; i.e., $\mathrm{Pol}(\mathbf{A}, \mathbf{A})$ contains all projections (also known as dictators) and is closed under composition [10]. Polymorphisms of the (more general) PCSPs form *minions*; i.e, they are closed under taking minors.[3] Formally, given an L-ary function $f : A^L \to B$, its *minor* relative to a map $\pi : [L] \to [L']$ is the L'-ary function $f_{/\pi} : A^{L'} \to B$ defined by

$$f_{/\pi}(a_1, \ldots, a_{L'}) = f(a_{\pi(1)}, \ldots, a_{\pi(L)}). \tag{2.1}$$

Equivalently, a minor of f is a function obtained from f by identifying variables, permuting variables, and introducing dummy variables. Rather than focussing on minions of functions, we consider here abstract minions, as described and used in [22].

[3] We remark that clones are also closed under taking minors.

DEFINITION 2.3. *A* minion $\mathscr{M}$ *consists in the disjoint union of sets* $\mathscr{M}^{(L)}$ *for* $L \in \mathbb{N}$ *equipped with operations* $(\cdot)_{/\pi} : \mathscr{M}^{(L)} \to \mathscr{M}^{(L')}$ *for all functions* $\pi : [L] \to [L']$*, which satisfy*

- $(M_{/\pi})_{/\tilde{\pi}} = M_{/\tilde{\pi}\circ\pi}$ *for* $\pi : [L] \to [L']$*,* $\tilde{\pi} : [L'] \to [L'']$ *and*
- $M_{/\,\mathrm{id}} = M$

for all $M \in \mathscr{M}^{(L)}$*.*

DEFINITION 2.4. *For two minions* $\mathscr{M}$ *and* $\mathscr{N}$*, a* minion homomorphism $\xi : \mathscr{M} \to \mathscr{N}$ *is a map that preserves arities and minors: Given* $M \in \mathscr{M}^{(L)}$ *and* $\pi : [L] \to [L']$*,* $\xi(M) \in \mathscr{N}^{(L)}$ *and* $\xi(M_{/\pi}) = \xi(M)_{/\pi}$*.*

For any PCSP template $(\mathbf{A}, \mathbf{B})$, the set $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ of its polymorphisms equipped with the operations described by (2.1) is a minion [7]. One of the results in [7] established that minion homomorphisms give rise to polynomial-time reductions: If there is a minion homomorphism from $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ to $\mathrm{Pol}(\mathbf{A}', \mathbf{B}')$, then $\mathrm{PCSP}(\mathbf{A}', \mathbf{B}') \leq_p \mathrm{PCSP}(\mathbf{A}, \mathbf{B})$. Minions are also useful for characterising the power of algorithms, as we will discuss later.

Existing algorithms One way to establish tractability of PCSPs is to reduce to CSPs. Let $(\mathbf{A}, \mathbf{B})$ be a PCSP template. A structure $\mathbf{C}$ is called a (homomorphic) *sandwich* if $\mathbf{A} \to \mathbf{C} \to \mathbf{B}$. It is known that, in this case, $\mathrm{PCSP}(\mathbf{A}, \mathbf{B}) \leq_p \mathrm{CSP}(\mathbf{C})$.[4] Thus, if $\mathbf{C}$ is a tractable CSP template then $(\mathbf{A}, \mathbf{B})$ is a tractable PCSP template. If $\mathbf{C}$ has a finite domain, we say that $(\mathbf{A}, \mathbf{B})$ is *finitely tractable*.

EXAMPLE 2.4. *The PCSP template* $(\mathbf{1\text{-}in\text{-}3}, \mathbf{NAE})$ *from Example 2.2 is tractable [18] but not finitely tractable unless P=NP, as shown in [7].*

Another way to establish tractability for PCSPs is to leverage convex relaxations. In the introduction, we mentioned three studied relaxations: BLP [55], AIP [18], and BLP + AIP [20]. Their powers have been characterised in [7, 22] in terms of certain minions and polymorphism identities. The details of these relaxations and the characterisations are provided in Appendix A of the full version [31].

All known tractable PCSPs are solved by finite tractability (i.e., by a reduction to a tractable finite-domain CSP) or by BLP + AIP. The next example identifies a simple PCSP template not captured by either of these two methods.

EXAMPLE 2.5. *Consider the relational structures* $\mathbf{A} = (A; R_1^{\mathbf{A}}, R_2^{\mathbf{A}})$ *and* $\mathbf{B} = (B; R_1^{\mathbf{B}}, R_2^{\mathbf{B}})$ *on the domain* $A = B = \{0, \dots, 6\}$ *with the following relations:* $R_1^{\mathbf{A}} = \{(0,0,1), (0,1,0), (1,0,0)\}$ *is* $\mathbf{1\text{-}in\text{-}3}$ *on* $\{0,1\}$*,* $R_1^{\mathbf{B}} = \{0,1\}^3 \setminus \{(0,0,0), (1,1,1)\}$ *is* $\mathbf{NAE}$ *on* $\{0,1\}$*, and* $R_2^{\mathbf{A}} = R_2^{\mathbf{B}} = \{(2,3), (3,2), (4,5), (5,6), (6,4)\}$*. The identity mapping is a homomorphism from* $\mathbf{A}$ *to* $\mathbf{B}$*, so* $(\mathbf{A}, \mathbf{B})$ *is a PCSP template. Since the directed graph corresponding to* $R_2^{\mathbf{A}} = R_2^{\mathbf{B}}$ *is a disjoint union of a directed* 2*-cycle and a directed* 3*-cycle, [22, Example 6.1] shows that the* BLP + AIP *algorithm does not solve* $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$*. We claim that the template* $(\mathbf{A}, \mathbf{B})$ *is not finitely tractable. For contradiction, assume that there is a finite relational structure* $\mathbf{C} = (C; R_1^{\mathbf{C}}, R_2^{\mathbf{C}})$ *such that* $\mathbf{A} \to \mathbf{C} \to \mathbf{B}$ *and* $\mathrm{CSP}(\mathbf{C})$ *is tractable. We will argue that this would imply finite tractability of* $(\mathbf{1\text{-}in\text{-}3}, \mathbf{NAE})$*, which contradicts the result in [7] (unless P=NP); cf. Example 2.4. Indeed, the existence of such* $\mathbf{C}$ *gives the following chain of homomorphisms:*

$$\mathbf{1\text{-}in\text{-}3} = (\{0,1\}; R_1^{\mathbf{A}}) \to (A; R_1^{\mathbf{A}}) \to (C; R_1^{\mathbf{C}}) \to (B; R_1^{\mathbf{B}}) \to (\{0,1\}; R_1^{\mathbf{B}}) = \mathbf{NAE} \tag{2.2}$$

where the first map is the inclusion of $\{0,1\}$ *in* A*, the second and the third are the maps witnessing* $\mathbf{A} \to \mathbf{C} \to \mathbf{B}$*, and the fourth is any map* $g : B \to \{0,1\}$ *such that* $g(0) = 0$ *and* $g(1) = 1$*. Let* $\tilde{\mathbf{C}} = (C; R_1^{\mathbf{C}})$*. Observe that* $\tilde{\mathbf{C}}$ *is tractable since the inclusion map gives a minion homomorphism* $\mathrm{Pol}(\mathbf{C}, \mathbf{C}) \to \mathrm{Pol}(\tilde{\mathbf{C}}, \tilde{\mathbf{C}})$*, and thus* $\mathrm{CSP}(\tilde{\mathbf{C}}) = \mathrm{PCSP}(\tilde{\mathbf{C}}, \tilde{\mathbf{C}}) \leq_p \mathrm{PCSP}(\mathbf{C}, \mathbf{C}) = \mathrm{CSP}(\mathbf{C})$ *by [7, Theorem 3.1]. This proves the claim, as (2.2) established* $\mathbf{1\text{-}in\text{-}3} \to \tilde{\mathbf{C}} \to \mathbf{NAE}$*.*

The template from Example 2.5 will be proved tractable later (in Example 3.1) using our new algorithm, which we will present next.

[4] This is a special case of homomorphic relaxation [7], which we do not need here.

3 The CLAP algorithm

Let $(\mathbf{A}, \mathbf{B})$ be a PCSP template with signature σ and let $\mathbf{X}$ be an instance of PCSP$(\mathbf{A}, \mathbf{B})$. Without loss of generality, we assume that σ contains a unary symbol R_{u} such that $R_{\mathrm{u}}^{\mathbf{X}} = X$, $R_{\mathrm{u}}^{\mathbf{A}} = A$, and $R_{\mathrm{u}}^{\mathbf{B}} = B$. If this is not the case, the signature and the instance can be extended without changing the set of solutions. Our algorithm – the combined **C**B**L**P+**AIP** algorithm (CLAP), presented in Algorithm 1 – builds on BLP [7] and BLP + AIP [22]. As in Appendix A of the full version [31], where BLP, AIP, and BLP + AIP are presented in full detail for completeness, by $\lambda_{\mathbf{x},R}(\mathbf{a})$ we denote the variable of BLP$(\mathbf{X}, \mathbf{A})$ associated with $\mathbf{x} \in R^{\mathbf{X}}$ and $\mathbf{a} \in R^{\mathbf{A}}$, where $R \in \sigma$. The algorithm has polynomial time complexity in the size of the input instance: Letting $g = \sum_{R \in \sigma} |R^{\mathbf{X}}||R^{\mathbf{A}}|$, $\mathcal{O}(g^2)$ BLP calls and $\mathcal{O}(g)$ BLP + AIP calls occur. We say that CLAP *solves* PCSP$(\mathbf{A}, \mathbf{B})$ if, for every instance $\mathbf{X}$ of PCSP$(\mathbf{A}, \mathbf{B})$, we have (i) if $\mathbf{X} \to \mathbf{A}$ then CLAP accepts $\mathbf{X}$, and (ii) if $\mathbf{X}$ is accepted by CLAP then $\mathbf{X} \to \mathbf{B}$.

Algorithm 1 The CLAP algorithm

Input: an instance $\mathbf{X}$ of PCSP$(\mathbf{A}, \mathbf{B})$ of signature σ
Output: YES if $\mathbf{X} \to \mathbf{A}$ and NO if $\mathbf{X} \not\to \mathbf{B}$

for $R \in \sigma$, $\mathbf{x} \in R^{\mathbf{X}}$
  set $S_{\mathbf{x}} := R^{\mathbf{A}}$;
end for
repeat
  for $R \in \sigma, \mathbf{x} \in R^{\mathbf{X}}$, $\mathbf{a} \in S_{\mathbf{x}}$
    if BLP$(\mathbf{X}, \mathbf{A})$ with $\lambda_{\mathbf{x},R}(\mathbf{a}) = 1$ and $\lambda_{\mathbf{x}',R'}(\mathbf{a}') = 0$ for every $R' \in \sigma$, $\mathbf{x}' \in R'^{\mathbf{X}}$, and $\mathbf{a}' \notin S_{\mathbf{x}'}$ is not feasible
      remove $\mathbf{a}$ from $S_{\mathbf{x}}$;
    end if
  end for
until no set $S_{\mathbf{x}}$ is changed;
if some $S_{\mathbf{x}}$ is empty
  return NO;
else
  for $R \in \sigma, \mathbf{x} \in R^{\mathbf{X}}$, $\mathbf{a} \in S_{\mathbf{x}}$
    if BLP + AIP$(\mathbf{X}, \mathbf{A})$ with $\lambda_{\mathbf{x},R}(\mathbf{a}) = 1$ and $\lambda_{\mathbf{x}',R'}(\mathbf{a}') = 0$ for every $R' \in \sigma$, $\mathbf{x}' \in R'^{\mathbf{X}}$, and $\mathbf{a}' \notin S_{\mathbf{x}'}$ is feasible
      return YES;
    end if
  end for
  return NO;
end if

Characterisation Our first main result – Theorem 3.1 – is a minion-theoretic characterisation of the power of the CLAP algorithm. In particular, we will introduce in Definition 3.2 a minion $\mathscr{C}$ such that, for any PCSP template $(\mathbf{A}, \mathbf{B})$, the CLAP algorithm solves PCSP$(\mathbf{A}, \mathbf{B})$ if and only if there is a minion homomorphism from $\mathscr{C}$ to Pol$(\mathbf{A}, \mathbf{B})$. The two directions are proved in the full version [31]. Combining the sufficiency condition with our second main result – Theorem 3.2, proved in the full version [31] – will then yield a sufficient condition for CLAP to solve a given PCSP template, in terms of a weak notion of symmetry for the polymorphisms of the template.

The L-ary objects of the minion $\mathscr{C}$ are pairs $(M, \boldsymbol{\mu})$, where M is a matrix with L rows and infinitely many columns[5] encoding the BLP computations of CLAP and $\boldsymbol{\mu}$ is an L-ary vector of integers encoding the AIP computation of CLAP. The matrices M in $\mathscr{C}$ have a special structure, which we call "skeletal".

DEFINITION 3.1. *Let M be a $p \times \aleph_0$ matrix with $p \in \mathbb{N}$. We say that M is* skeletal *if, for each $j \in [p]$, either $\mathbf{e}_j^T M = \mathbf{0}_{\aleph_0}^T$ or $M\mathbf{e}_i = \mathbf{e}_j$ for some $i \in \mathbb{N}$.*

[5] This is a notational convenience which could be replaced with finitely many columns but less elegant statements.

In other words, either the j-th row of M is the zero vector or some column of M is the j-th standard unit vector. Equivalently, M is skeletal if there exist permutation matrices $P \in \mathbb{R}^{p,p}$ and $Q \in \mathbb{R}^{\aleph_0,\aleph_0}$ such that $PMQ = \begin{bmatrix} I_k & \tilde{M} \\ O & O \end{bmatrix}$ for some $k \leq p$ and some $\tilde{M} \in \mathbb{R}^{k,\aleph_0}$. The name indicates that the "body" of a skeletal matrix (the nonzero rows) is completely supported by a "skeleton" (the identity block).

We are now ready to define the minion $\mathscr{C}$. The L-ary objects of $\mathscr{C}$ are pairs $(M, \boldsymbol{\mu})$, where M is a skeletal matrix of size $L \times \aleph_0$ and $\boldsymbol{\mu}$ is an affine vector (i.e., an integer vector whose entries sum up to one) of size L. We require that every column of M should be stochastic and M should have only finitely many different columns. Moreover, we require a particular relationship between M and $\boldsymbol{\mu}$ formalised in (c_4) in Definition 3.2.

DEFINITION 3.2. *For $L \in \mathbb{N}$, let $\mathscr{C}^{(L)}$ be the set of pairs $(M, \boldsymbol{\mu})$ such that $M \in \mathbb{Q}^{L,\aleph_0}$, $\boldsymbol{\mu} \in \mathbb{Z}^L$, and the following requirements are met:*

(c_1) *M is entrywise nonnegative*;
(c_2) $\mathbf{1}_L^T M = \mathbf{1}_{\aleph_0}^T$;
(c_3) $\mathbf{1}_L^T \boldsymbol{\mu} = 1$;
(c_4) $\mathrm{supp}(\boldsymbol{\mu}) \subseteq \mathrm{supp}(M\mathbf{e}_1)$;
(c_5) $\exists t \in \mathbb{N}$ *such that* $M\mathbf{e}_i = M\mathbf{e}_t \quad \forall i \geq t$;
(c_6) *M is skeletal.*

We define $\mathscr{C}$ as the disjoint union of L-ary parts, $\mathscr{C} \coloneqq \bigcup_{L \geq 1} \mathscr{C}^{(L)}$.

We defined $\mathscr{C}$ as a set. For $\mathscr{C}$ to be a minion, we need to define the minor operation on $\mathscr{C}$ and verify that it preserves the structure of $\mathscr{C}$. This is easy and done in the full version [31].

Our first result is the following characterisation of the power of CLAP.

THEOREM 3.1. *Let $(\mathbf{A}, \mathbf{B})$ be a PCSP template. Then,* CLAP *solves* $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ *if and only if there is a minion homomorphism from $\mathscr{C}$ to* $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$.

H-symmetry Our second main result is a sufficient condition on a PCSP template $(\mathbf{A}, \mathbf{B})$ to guarantee that CLAP solves $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$. The condition is through symmetries satisfied by polymorphisms of the template. In particular, in Theorem 3.2 we will show that if $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ contains infinitely many operations that are "H-symmetric" for a suitable matrix H, then there is a minion homomorphism from $\mathscr{C}$ to $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$, and thus CLAP solves $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ by (the sufficiency part of) Theorem 3.1.

In order to define the notion of H-symmetricity, we need a few auxiliary definitions. A vector $\mathbf{w} = (w_i) \in \mathbb{R}^p$ is *tieless* if, for any two indices $i \neq i' \in [p]$, $w_i \neq 0 \Rightarrow w_i \neq w_{i'}$. A *tie matrix* is a matrix having integer nonnegative entries, each of whose columns is a tieless vector. Given an $m \times p$ tie matrix H, we say that a vector $\mathbf{v} \in \mathbb{R}^p$ is *H-tieless* if $H\mathbf{v}$ is tieless.

Let A be a finite set, let $L \in \mathbb{N}$, and take $\mathbf{a} = (a_1, \ldots, a_L) \in A^L$. We define the (multiplicity) vector $\mathbf{a}^{\#} \in \mathbb{N}_0^{|A|}$ whose a-th entry is $|\{i \in [L] : a_i = a\}|$ for each $a \in A$.

DEFINITION 3.3. *Let A, B be finite sets, and consider a function $f : A^L \to B$ for some $L \in \mathbb{N}$. Given an $m \times |A|$ tie matrix H, we say that f is* H-symmetric *if*

$$f_{/\pi}(\mathbf{a}) = f(\mathbf{a}) \qquad \forall \pi : [L] \to [L] \text{ permutation}, \quad \forall \mathbf{a} \in A^L \text{ such that } \mathbf{a}^{\#} \text{ is } H\text{-tieless}.$$

Our second result is the following sufficient condition for tractability of PCSPs.

THEOREM 3.2. *Let $(\mathbf{A}, \mathbf{B})$ be a PCSP template and suppose* $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ *contains H-symmetric operations of arbitrarily large arity for some $m \times |A|$ tie matrix H, $m \in \mathbb{N}$. Then there exists a minion homomorphism from $\mathscr{C}$ to* $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$.

Recall from Definition 3.1 the notion of a skeletal matrix. The "skeleton" represents the link between CLAP and the above-defined notion of H-symmetricity. Indeed, on the one hand the presence of the identity block captures the fact that each BLP solution computed by CLAP gives probability 1 to some constraint assignment (cf. line 6 of Algorithm 1). On the other hand, the "Tiebreak Lemma" [31, Lemma 32] – stated and proved in the full version – shows that, by virtue of this feature, finitely many skeletal matrices can always be simultaneously reduced to H-tieless vectors; through some technicalities detailed in the proof of Theorem 3.2, this last property mirrors the behaviour of H-symmetric operations.

We now mention some consequences of Theorem 3.2. First, observe that a vector of size 1 is always tieless. Hence, if we take any $1 \times |A|$ integer nonnegative matrix as H, we see that H is a tie matrix and that $\mathbf{a}^{\#}$ is H-tieless for each tuple $\mathbf{a}$ in the domain of f; therefore, for such an H, f being H-symmetric reduces to f being symmetric. On the other hand, having Definition 3.3 in mind, adding rows to H increases the chance that $H\mathbf{a}^{\#}$ has some ties, in which case f is released from the requirement of being symmetric on $\mathbf{a}$. In this sense, H encodes the "exceptions to symmetry" that f is allowed to have: The more rows H has, the stronger Theorem 3.2 becomes. If, for instance, H is the identity matrix of order $|A|$, then an H-symmetric operation needs to be symmetric only on those tuples where each entry occurs with a different multiplicity. A very special example of such an $I_{|A|}$-symmetric operation is a function f that returns (the homomorphic image of) the *most-frequent* entry in the input tuple whenever it is unique, and, in any other case, f is, say, (the homomorphic image of) a projection. Other, more creative choices for H allow capturing operations having more complex exceptions to symmetry, as shown in Example 3.1.

Theorems 3.1 and 3.2 together establish that the CLAP algorithm solves any PCSP template admitting arbitrarily large polymorphisms having some exceptions to symmetry that can be encoded via a tie matrix.

EXAMPLE 3.1. *Recall the PCSP template* $(\mathbf{A}, \mathbf{B})$ *from Example 2.5, where it was shown that* $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ *is not finitely tractable and not solved by the* $\mathrm{BLP} + \mathrm{AIP}$ *algorithm from [20]. We will show that* $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ *is solved by* CLAP.

Take $L \in \mathbb{N}$ *and consider the function* $f : A^L \to B$ *defined as follows: For* $\mathbf{a} = (a_1, \dots, a_L) \in A^L$,

- *if* $\mathbf{a} \in \{0,1\}^L$, *look at* $\mathbf{a}_1^{\#}$, *i.e., the multiplicity of* $1 \in A$ *in the tuple* $\mathbf{a}$;[6]
  - *if* $\mathbf{a}_1^{\#} < \frac{L}{3}$, *set* $f(\mathbf{a}) = 0$;
  - *if* $\mathbf{a}_1^{\#} > \frac{L}{3}$, *set* $f(\mathbf{a}) = 1$;
  - *if* $\mathbf{a}_1^{\#} = \frac{L}{3}$, *set* $f(\mathbf{a}) = a_1$;
- *if* $\mathbf{a} \in \{2,3,4,5,6\}^L$,
  - *if there is a unique element* $a \in A$ *having maximum multiplicity in* $\mathbf{a}$, *set* $f(\mathbf{a}) = a$;
  - *if there is more than one element of* A *having maximum multiplicity in* $\mathbf{a}$, *set* $f(\mathbf{a}) = a_1$;
- *otherwise, set* $f(\mathbf{a}) = 0$.[7]

We claim that $f \in \mathrm{Pol}(\mathbf{A}, \mathbf{B})$. *To see that* f *preserves* R_1, *consider a tuple* $\boldsymbol{\rho} = (\mathbf{r}_1, \dots, \mathbf{r}_L)$ *of elements of* $R_1^{\mathbf{A}}$, *where* $\mathbf{r}_i = (a_i, b_i, c_i)$ *for* $i \in [L]$. *We shall let* $\mathbf{a} = (a_1, \dots, a_L)$, $\mathbf{b} = (b_1, \dots, b_L)$, *and* $\mathbf{c} = (c_1, \dots, c_L)$. *Notice that*

$$\mathbf{a}_1^{\#} + \mathbf{b}_1^{\#} + \mathbf{c}_1^{\#} = L. \tag{3.3}$$

If $f(\mathbf{a}) = f(\mathbf{b}) = f(\mathbf{c}) = 0$, *then* $\mathbf{a}_1^{\#} \le \frac{L}{3}$, $\mathbf{b}_1^{\#} \le \frac{L}{3}$, *and* $\mathbf{c}_1^{\#} \le \frac{L}{3}$; *by* (3.3), *this implies that* $\mathbf{a}_1^{\#} = \mathbf{b}_1^{\#} = \mathbf{c}_1^{\#} = \frac{L}{3}$. *Hence,* $(0,0,0) = (f(\mathbf{a}), f(\mathbf{b}), f(\mathbf{c})) = (a_1, b_1, c_1) = \mathbf{r}_1 \in R_1^{\mathbf{A}}$, *a contradiction. Similarly,* $f(\mathbf{a}) = f(\mathbf{b}) = f(\mathbf{c}) = 1$ *would yield* $\mathbf{a}_1^{\#} \ge \frac{L}{3}$, $\mathbf{b}_1^{\#} \ge \frac{L}{3}$, *and* $\mathbf{c}_1^{\#} \ge \frac{L}{3}$; *again by* (3.3), *this implies that* $\mathbf{a}_1^{\#} = \mathbf{b}_1^{\#} = \mathbf{c}_1^{\#} = \frac{L}{3}$, *hence* $(1,1,1) = (f(\mathbf{a}), f(\mathbf{b}), f(\mathbf{c})) = (a_1, b_1, c_1) = \mathbf{r}_1 \in R_1^{\mathbf{A}}$, *also a contradiction. We conclude that* $f(\boldsymbol{\rho}) = (f(\mathbf{a}), f(\mathbf{b}), f(\mathbf{c})) \in R_1^{\mathbf{B}}$, *thus showing that* f *preserves* R_1.

As for R_2, *let* $\boldsymbol{\rho} = (\mathbf{r}_1, \dots, \mathbf{r}_L)$ *be a tuple of elements of* $R_2^{\mathbf{A}}$, *where* $\mathbf{r}_i = (a_i, b_i)$ *for* $i \in [L]$, *and let* $\mathbf{a} = (a_1, \dots, a_L)$ *and* $\mathbf{b} = (b_1, \dots, b_L)$. *By definition, the directed graph having vertex set* $\{2,3,4,5,6\}$ *and edge set* $R_2^{\mathbf{A}} = R_2^{\mathbf{B}}$ *consists of the disjoint union of a directed 2-cycle and a directed 3-cycle and, hence, all of its vertices have in-degree and out-degree one. As a consequence, the multiplicity of a directed edge* (a, b) *in the tuple* $\boldsymbol{\rho}$ *equals both the multiplicity of* a *in* $\mathbf{a}$ *and the multiplicity of* b *in* $\mathbf{b}$. *Therefore, if the tuple* $\boldsymbol{\rho}$ *has a unique element* $\mathbf{r} = (a, b)$ *with maximum multiplicity, then* $f(\boldsymbol{\rho}) = (f(\mathbf{a}), f(\mathbf{b})) = (a, b) = \mathbf{r} \in R_2^{\mathbf{B}}$. *Otherwise,* $f(\boldsymbol{\rho}) = (a_1, b_1) = \mathbf{r}_1 \in R_2^{\mathbf{B}}$. *This shows that* f *preserves* R_2, *too, and is thus a polymorphism of* $(\mathbf{A}, \mathbf{B})$.

Consider the matrix $H = \mathrm{diag}(1,2,1,1,1,1,1)$, *and observe that* H *is a tie matrix. We claim that* f *is* H*-symmetric. Let* $\pi : [L] \to [L]$ *be a permutation, and take a tuple* $\mathbf{a} = (a_1, \dots, a_L) \in A^L$ *such that* $\mathbf{a}^{\#}$ *is* H*-tieless;*

[6] Since the elements of A are numbered starting from 0, $\mathbf{a}_1^{\#}$ is the second entry of the vector $\mathbf{a}^{\#}$.

[7] Assigning any value in $\{0, \dots, 6\}$ to $f(\mathbf{a})$ would work here.

i.e., the vector $H\mathbf{a}^{\#} = (\mathbf{a}_0^{\#}, 2\mathbf{a}_1^{\#}, \mathbf{a}_2^{\#}, \mathbf{a}_3^{\#}, \mathbf{a}_4^{\#}, \mathbf{a}_5^{\#}, \mathbf{a}_6^{\#})$ *is tieless. Write* $\tilde{\mathbf{a}} = (a_{\pi(1)}, \dots, a_{\pi(L)})$*, and observe that* $\tilde{\mathbf{a}}^{\#} = \mathbf{a}^{\#}$*.*

- *If* $\mathbf{a} \in \{0,1\}^L$*, we get* $\mathbf{a}_0^{\#} \neq 2\mathbf{a}_1^{\#}$*; since* $\mathbf{a}_0^{\#} + \mathbf{a}_1^{\#} = L$*, this gives* $2\mathbf{a}_1^{\#} \neq L - \mathbf{a}_1^{\#}$ *so that* $\mathbf{a}_1^{\#} \neq \frac{L}{3}$*. As a consequence,* $f(\mathbf{a}) = f(\tilde{\mathbf{a}})$*.*
- *If* $\mathbf{a} \in \{2,3,4,5,6\}^L$*, the condition above implies that the tuple* $(\mathbf{a}_2^{\#}, \mathbf{a}_3^{\#}, \mathbf{a}_4^{\#}, \mathbf{a}_5^{\#}, \mathbf{a}_6^{\#})$ *has a unique maximum element and, hence, there is a unique element* a *of* A *having maximum multiplicity in* $\mathbf{a}$ *(and in* $\tilde{\mathbf{a}}$*). Therefore,* $f(\mathbf{a}) = a = f(\tilde{\mathbf{a}})$*.*
- *If* $\mathbf{a} \notin \{0,1\}^L \cup \{2,3,4,5,6\}^L$*, then* $f(\mathbf{a}) = 0 = f(\tilde{\mathbf{a}})$*.*

We conclude that, in each case, $f(\mathbf{a}) = f(\tilde{\mathbf{a}}) = f_{/\pi}(\mathbf{a})$*, which means that* f *is* H*-symmetric. By Theorems 3.1 and 3.2,* CLAP *solves* $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$*.*[8]

REMARK 3.1. *Consider the minion* $\mathscr{M}_{\mathrm{BLP+AIP}}$ *from [22] (cf. [31, Appendix A.3]). A direct consequence of Example 3.1, Theorem 3.1, and [22, Lemma 5.4] is that there is no minion homomorphism from* $\mathscr{M}_{\mathrm{BLP+AIP}}$ *to* $\mathscr{C}$*. On the other hand, the function*

$$\begin{aligned} \vartheta : \mathscr{C} &\to \mathscr{M}_{\mathrm{BLP+AIP}} \\ (M, \boldsymbol{\mu}) &\mapsto (M\mathbf{e}_1, \boldsymbol{\mu}) \end{aligned}$$

is readily seen to be a minion homomorphism. It follows that CLAP *solves any PCSP template solved by* BLP + AIP *(as is also clear from the description of the two algorithms).*

REMARK 3.2. *Similar to [20], the assumption in Theorem 3.2 can be weakened as follows: Instead of requiring* H*-symmetric polymorphisms of arbitrarily large arity, it turns out to be enough requiring* H*-block-symmetric polymorphisms of arbitrarily large width, where the definition of an* H*-block-symmetric operation mirrors that of a block-symmetric operation in [20]. The proof of this stronger result is very similar to that of Theorem 3.2. The details can be found in the full version [31, Appendix C].*

REMARK 3.3. *A (possibly) stronger version of the* CLAP *algorithm consists in running* BLP + AIP *(instead of just* BLP*) at each iteration in the* ***for*** *loop in lines* 5–9 *of Algorithm 1, and then removing the additional* ***for*** *loop in lines* 14–18*. This algorithm can be called C(*BLP + AIP*). An analysis entirely analogous to the one presented in this paper shows that the power of C(*BLP + AIP*) is captured by the minion* $\tilde{\mathscr{C}}$ *defined like* $\mathscr{C}$ *with the following difference: The* L*-ary elements of* $\tilde{\mathscr{C}}$ *are pairs* (M, N)*, where* M *is as in* $\mathscr{C}$ *while* N *is an integer matrix of the same size as* M *taking the role of* $\boldsymbol{\mu}$ *(in particular,* N *satisfies the "refinement condition"* $\mathrm{supp}(Ne_i) \subseteq \mathrm{supp}(Me_i)$ $\forall i \in \mathbb{N}$*, analogous to* (c_4) *in Definition 3.2). A possible direction for future research is to investigate whether the richer structure of* $\tilde{\mathscr{C}}$ *can be exploited to obtain a stronger version of Theorem 3.2.*

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998.

[2] S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. *J. ACM*, 45(1):70–122, 1998.

[3] P. Austrin, A. Bhangale, and A. Potukuchi. Improved inapproximability of rainbow coloring. In *Proceedings of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'20)*, pages 1479–1495, 2020, arXiv:1810.02784.

[4] P. Austrin, V. Guruswami, and J. Håstad. $(2+\epsilon)$-Sat is NP-hard. *SIAM J. Comput.*, 46(5):1554–1573, 2017, eccc:2013/159.

[5] L. Barto. The collapse of the bounded width hierarchy. *J. Log. Comput.*, 26(3):923–943, 2016.

[8] The discussion in Example 3.1 shows that the PCSP template (**1-in-3**, **NAE**) admits H-symmetric polymorphisms of *all* arities; hence, the AIP part of the CLAP algorithm is not needed in this case (cf. [31, Remark 33]). In fact, the template is already solved by SBLP, since it admits alternating-threshold polymorphisms of all odd arities (cf. Example 2.3 and [18]).

[6] L. Barto, D. Battistelli, and K. M. Berg. Symmetric Promise Constraint Satisfaction Problems: Beyond the Boolean Case. In *Proceedings of the 38th International Symposium on Theoretical Aspects of Computer Science (STACS'21)*, volume 187 of *LIPIcs*, pages 10:1–10:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, arXiv:2010.04623.
[7] L. Barto, J. Bulín, A. A. Krokhin, and J. Opršal. Algebraic approach to promise constraint satisfaction. *Journal of the ACM*, 68(4):28:1–28:66, 2021, arXiv:1811.00970.
[8] L. Barto and M. Kozik. Constraint Satisfaction Problems Solvable by Local Consistency Methods. *J. ACM*, 61(1), 2014. Article No. 3.
[9] L. Barto and M. Kozik. Robustly solvable constraint satisfaction problems. *SIAM J. Comput.*, 45(4):1646–1669, 2016, arXiv:1512.01157.
[10] L. Barto, A. Krokhin, and R. Willard. Polymorphisms, and how to use them. In A. Krokhin and S. Živný, 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.
[11] L. Barto, J. Opršal, and M. Pinsker. The wonderland of reflections. *Isr. J. Math*, 223(1):363–398, Feb 2018, arXiv:1510.04521.
[12] L. Barto and M. Pinsker. Topology is irrelevant (in a dichotomy conjecture for infinite domain constraint satisfaction problems). *SIAM J. Comput.*, 49(2):365–393, 2020, arXiv:1909.06201.
[13] J. Berman, P. Idziak, P. Marković, R. McKenzie, M. Valeriote, and R. Willard. Varieties with few subalgebras of powers. *Trans. Am. Math. Soc.*, 362(3):1445–1473, 2010.
[14] C. Bessiere and R. Debruyne. Theoretical analysis of singleton arc consistency and its extensions. *Artif. Intell.*, 172(1):29–41, 2008.
[15] A. Blum. New approximation algorithms for graph coloring. *J. ACM*, 41(3):470–516, 1994.
[16] M. Bodirsky, B. Martin, and A. Mottet. Discrete temporal constraint satisfaction problems. *J. ACM*, 65(2):9:1–9:41, 2018, arXiv:1503.08572.
[17] M. Bodirsky, A. Mottet, M. Olšák, J. Opršal, M. Pinsker, and R. Willard. ω-categorical structures avoiding height 1 identities. *Transactions of the American Mathematical Society*, 374(1):327–350, 2021, arXiv:2006.12254.
[18] J. Brakensiek and V. Guruswami. Promise Constraint Satisfaction: Structure Theory and a Symmetric Boolean Dichotomy. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'18)*, pages 1782–1801. SIAM, 2018, arXiv:1704.01937.
[19] J. Brakensiek and V. Guruswami. An algorithmic blend of LPs and ring equations for promise CSPs. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'19)*, pages 436–455, 2019, arXiv:1807.05194.
[20] J. Brakensiek and V. Guruswami. Symmetric polymorphisms and efficient decidability of promise CSPs. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms (SODA'20)*, pages 297–304, 2020, arXiv:1907.04383.
[21] J. Brakensiek, V. Guruswami, and S. Sandeep. Conditional Dichotomy of Boolean Ordered Promise CSPs. In *Proceedings of the 48th International Colloquium on Automata, Languages, and Programming (ICALP'21)*, volume 198 of *LIPIcs*, pages 37:1–37:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, 2102.11854.
[22] J. Brakensiek, V. Guruswami, M. Wrochna, and S. Živný. The power of the combined basic LP and affine relaxation for promise CSPs. *SIAM J. Comput.*, 49:1232–1248, 2020, arXiv:1907.04383.
[23] A. Brandts, M. Wrochna, and S. Živný. The complexity of promise SAT on non-Boolean domains. *ACM Trans. Comput. Theory*, 13(4):26:1–26:20, 2021, arXiv:1911.09065.
[24] A. Brandts and S. Živný. Beyond PCSP(1-in-3,NAE). In *Proceedings of the 48th International Colloquium on Automata, Languages, and Programming (ICALP'21)*, volume 198 of *LIPIcs*, pages 121:1–121:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, arXiv:2104.12800.
[25] A. Bulatov. Bounded relational width. Unpublished manuscript, 2009.
[26] A. Bulatov, P. Jeavons, and A. Krokhin. Classifying the complexity of constraints using finite algebras. *SIAM J. Comput.*, 34(3):720–742, 2005.
[27] A. A. Bulatov. A dichotomy theorem for constraint satisfaction problems on a 3-element set. *J. ACM*, 53(1):66–120, 2006.
[28] A. A. Bulatov. A dichotomy theorem for nonuniform CSPs. In *Proceedings of the 58th Annual IEEE Symposium on Foundations of Computer Science (FOCS'17)*, pages 319–330, 2017, arXiv:1703.03021.
[29] A. A. Bulatov and V. Dalmau. A Simple Algorithm for Mal'tsev Constraints. *SIAM J. Comput.*, 36(1):16–27, 2006.
[30] H. Chen, V. Dalmau, and B. Grußien. Arc consistency and friends. *J. Log. Comput.*, 23(1):87–108, 2013.
[31] L. Ciardo and S. Živný. CLAP: A New Algorithm for Promise CSPs. 2021, arXiv:2107.05018.
[32] D. A. Cohen. Tractable decision for a constraint language implies tractable search. *Constraints*, 9(3):219–229, 2004.
[33] V. Dalmau, M. Kozik, A. A. Krokhin, K. Makarychev, Y. Makarychev, and J. Oprsal. Robust algorithms with polynomial loss for near-unanimity CSPs. *SIAM J. Comput.*, 48(6):1763–1795, 2019, arXiv:1607.04787.
[34] V. Dalmau and A. A. Krokhin. Robust Satisfiability for CSPs: Hardness and Algorithmic Results. *ACM Trans. Comput. Theory*, 5(4):15:1–15:25, 2013.
[35] R. Debruyne and C. Bessière. Some Practicable Filtering Techniques for the Constraint Satisfaction Problem. In

Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI'97), pages 412–417. Morgan Kaufmann, 1997.
[36] I. Dinur. The PCP theorem by gap amplification. *J. ACM*, 54(3):12, 2007.
[37] I. Dinur, O. Regev, and C. Smyth. The hardness of 3-uniform hypergraph coloring. *Comb.*, 25(5):519–535, Sept. 2005.
[38] T. Feder and M. 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, 1998.
[39] M. Ficak, M. Kozik, M. Olšák, and S. Stankiewicz. Dichotomy for Symmetric Boolean PCSPs. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP'19)*, volume 132, pages 57:1–57:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, arXiv:1904.12424.
[40] M. R. Garey and D. S. Johnson. The complexity of near-optimal graph coloring. *J. ACM*, 23(1):43–49, 1976.
[41] V. Guruswami and S. Khanna. On the hardness of 4-coloring a 3-colorable graph. *SIAM J. Discret. Math.*, 18(1):30–40, 2004.
[42] V. Guruswami and S. Sandeep. d-To-1 Hardness of Coloring 3-Colorable Graphs with O(1) Colors. In *Proceedings of the 47th International Colloquium on Automata, Languages, and Programming (ICALP'20)*, volume 168 of *LIPIcs*, pages 62:1–62:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.
[43] P. Hell and J. Nešetřil. On the complexity of H-coloring. *J. Comb. Theory, Ser. B*, 48(1):92–110, 1990.
[44] P. M. Idziak, P. Markovic, R. McKenzie, M. Valeriote, and R. Willard. Tractability and learnability arising from algebras with few subpowers. *SIAM J. Comput.*, 39(7):3023–3037, 2010.
[45] P. G. Jeavons. On the Algebraic Structure of Combinatorial Problems. *Theor. Comput. Sci.*, 200(1-2):185–204, 1998.
[46] P. G. Jeavons, D. A. Cohen, and M. Gyssens. Closure Properties of Constraints. *J. ACM*, 44(4):527–548, 1997.
[47] S. Khanna, N. Linial, and S. Safra. On the hardness of approximating the chromatic number. *Comb.*, 20(3):393–415, 2000.
[48] S. Khot. Improved Inaproximability Results for MaxClique, Chromatic Number and Approximate Graph Coloring. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science (FOCS'01)*, pages 600–609. IEEE Computer Society, 2001.
[49] S. Khot. On the power of unique 2-prover 1-round games. In *Proc. 34th Annual ACM Symposium on Theory of Computing (STOC'02)*, pages 767–775. ACM, 2002.
[50] V. Kolmogorov, A. A. Krokhin, and M. Rolínek. The complexity of general-valued CSPs. *SIAM J. Comput.*, 46(3):1087–1110, 2017, arXiv:1502.07327.
[51] V. Kolmogorov, J. Thapper, and S. Živný. The power of linear programming for general-valued CSPs. *SIAM J. Comput.*, 44(1):1–36, 2015, arXiv:1311.4219.
[52] M. Kozik. Solving CSPs Using Weak Local Consistency. *SIAM J. Comput.*, 50(4):1263–1286, 2021, arXiv:1605.00565.
[53] M. Kozik and J. Ochremiak. Algebraic properties of valued constraint satisfaction problem. In *Proceedings of the 42nd International Colloquium on Automata, Languages, and Programming (ICALP'15)*, volume 9134 of *Lecture Notes in Computer Science*, pages 846–858. Springer, 2015, arXiv:1403.0476.
[54] A. Krokhin and J. Opršal. The complexity of 3-colouring H-colourable graphs. In *Proceedings of the 60th Annual IEEE Symposium on Foundations of Computer Science (FOCS'19)*, pages 1227–1239, 2019, arxiv:1904.03214.
[55] G. Kun, R. O'Donnell, S. Tamaki, Y. Yoshida, and Y. Zhou. Linear programming, width-1 CSPs, and robust satisfaction. In *Proceedings of the 3rd Innovations in Theoretical Computer Science (ITCS'12)*, pages 484–495. ACM, 2012.
[56] G. Kun and M. Szegedy. A new line of attack on the dichotomy conjecture. *Eur. J. Comb.*, 52:338–367, 2016.
[57] A. K. Mackworth. Consistency in networks of relations. *Artif. Intell.*, 8(1):99–118, 1977.
[58] P. Raghavendra. Optimal algorithms and inapproximability results for every CSP? In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC'08)*, pages 245–254, 2008.
[59] T. Schaefer. The complexity of satisfiability problems. In *Proceedings of the 10th Annual ACM Symposium on the Theory of Computing (STOC'78)*, pages 216–226, 1978.
[60] J. Thapper and S. Živný. The complexity of finite-valued CSPs. *J. ACM*, 63(4), 2016, arXiv:1210.2987. Article No. 37.
[61] J. Thapper and S. Živný. The power of Sherali-Adams relaxations for general-valued CSPs. *SIAM J. Comput.*, 46(4):1241–1279, 2017, arXiv:1606.02577.
[62] J. Thapper and S. Živný. The limits of SDP relaxations for general-valued CSPs. *ACM Trans. Comput. Theory*, 10(3):12:1–12:22, 2018, arXiv:1612.01147.
[63] A. Wigderson. Improving the performance guarantee for approximate graph coloring. *J. ACM*, 30(4):729–735, 1983.
[64] M. Wrochna and S. Živný. Improved hardness for H-colourings of G-colourable graphs. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'20)*, pages 1426–1435, 2020, arxiv:1907.00872.
[65] D. Zhuk. A proof of the CSP dichotomy conjecture. *J. ACM*, 67(5):30:1–30:78, 2020, arXiv:1704.01914.