Hard constraint satisfaction problems have hard gaps at location 1 [1]

Peter Jonsson [a], Andrei Krokhin [b], Fredrik Kuivinen [c,*]

[a] *Department of Computer and Information Science, Linköpings Universitet, SE-581 83, Linköping, Sweden, phone: +46 13 282415*

[b] *Department of Computer Science, University of Durham, Science Laboratories, South Road, Durham DH1 3LE, UK, phone: +44 191 334 1743*

[c] *Department of Computer and Information Science, Linköpings Universitet, SE-581 83, Linköping, Sweden, phone: +46 13 286607*

Abstract

An instance of the *maximum constraint satisfaction problem* (Max CSP) is a finite collection of constraints on a set of variables, and the goal is to assign values to the variables that maximises the number of satisfied constraints. Max CSP captures many well-known problems (such as Max k-SAT and Max Cut) and is consequently **NP**-hard. Thus, it is natural to study how restrictions on the allowed constraint types (or *constraint language*) affect the complexity and approximability of Max CSP. The PCP theorem is equivalent to the existence of a constraint language for which Max CSP has a hard gap at location 1, i.e. it is **NP**-hard to distinguish between satisfiable instances and instances where at most some constant fraction of the constraints are satisfiable. All constraint languages, for which the CSP problem (i.e., the problem of deciding whether all constraints can be satisfied) is currently known to be **NP**-hard, have a certain algebraic property. We prove that any constraint language with this algebraic property makes Max CSP have a hard gap at location 1 which, in particular, implies that such problems cannot have a PTAS unless $\mathbf{P} = \mathbf{NP}$. We then apply this result to Max CSP restricted to a single constraint type; this class of problems contains, for instance, Max Cut and Max DiCut. Assuming $\mathbf{P} \neq \mathbf{NP}$, we show that such problems do not admit PTAS except in some trivial cases. Our results hold even if the number of occurrences of each variable is bounded by a constant. Finally, we give some applications of our results.

Key words: constraint satisfaction, optimisation, approximability, universal algebra, computational complexity, dichotomy
1991 MSC: 68Q17, 68Q25, 90C27

---

1 Introduction

Many combinatorial optimisation problems are **NP**-hard so there has been a great interest in constructing approximation algorithms for such problems. For some optimisation problems, there exist powerful approximation algorithms known as *polynomial-time approximation schemes* (PTAS). An optimisation problem Π has a PTAS A if, for any fixed rational $c > 1$ and for any instance $\mathcal{I}$ of Π, $A(\mathcal{I}, c)$ returns a c-approximate (i.e., within c of optimum) solution in time polynomial in $|\mathcal{I}|$. There are some well-known **NP**-hard optimisation problems that have the highly desirable property of admitting a PTAS: examples include KNAPSACK [33], EUCLIDEAN TSP [2], and INDEPENDENT SET restricted to planar graphs [6,46]. It is also well-known that a large number of optimisation problems do not admit PTAS unless some unexpected collapse of complexity classes occurs. For instance, problems like MAX k-SAT [4] and INDEPENDENT SET [5] do not admit a PTAS unless $\mathbf{P} = \mathbf{NP}$. We note that if Π is a problem that does not admit a PTAS, then there exists a constant $c > 1$ such that Π cannot be approximated within c in polynomial time. Throughout the paper, we assume that $\mathbf{P} \neq \mathbf{NP}$.

The *constraint satisfaction problem* (CSP) [53] and its optimisation variants have played an important role in research on approximability. For example, it is well known that the famous PCP theorem has an equivalent reformulation in terms of inapproximability of some CSP [4,26,56], and the recent combinatorial proof of this theorem [26] deals entirely with CSPs. Other important examples include Håstad's first optimal inapproximability results [32] and the work around the unique games conjecture (UGC) of Khot [16,39,40,52].

We will focus on a class of optimisation problems known as the *maximum constraint satisfaction problem* (MAX CSP). The most well-known examples in this class probably are MAX k-SAT and MAX CUT.

We are now ready to formally define our problem. Let D be a finite set. A subset $R \subseteq D^n$ is a *relation* and n is the *arity* of R. Let $R_D^{(k)}$ be the set of all k-ary relations on D and let $R_D = \cup_{i=1}^{\infty} R_D^{(i)}$. A *constraint language* is a finite subset of R_D.

Definition 1 (CSP(Γ)) *The constraint satisfaction problem over the con-*

* Corresponding author.
Email addresses: `petej@ida.liu.se` (Peter Jonsson), `andrei.krokhin@durham.ac.uk` (Andrei Krokhin), `freku@ida.liu.se` (Fredrik Kuivinen).

[1] Preliminary versions of parts of this article appeared in *Proceedings of the 2nd International Computer Science Symposium in Russia (CSR-2007)*, Ekaterinburg, Russia, 2007.

straint language Γ*, denoted* CSP(Γ)*, is defined to be the decision problem with instance* (V, C)*, where*

- V *is a set of variables, and*
- C *is a collection of constraints* $\{C_1, \ldots, C_q\}$*, in which each constraint* C_i *is a pair* $(R_i, \boldsymbol{s_i})$ *with* $\boldsymbol{s_i}$ *a list of variables of length* n_i*, called the constraint scope, and* $R_i \in \Gamma$ *is an* n_i*-ary relation in* R_D*, called the constraint relation.*

The question is whether there exists an assignment $s : V \to D$ *which satisfies all constraints in* C *or not. A constraint* $(R_i, (v_{i_1}, v_{i_2}, \ldots, v_{i_{n_i}})) \in C$ *is* satisfied *by an assignment* s *if the image of the constraint scope is a member of the constraint relation, i.e., if* $(s(v_{i_1}), s(v_{i_2}), \ldots, s(v_{i_{n_i}})) \in R_i$*.*

Many combinatorial problems are subsumed by the CSP framework; examples include problems in graph theory [31], combinatorial optimisation [38], and computational learning [23]. We refer the reader to [18] for an introduction to this framework.

For a constraint language $\Gamma \subseteq R_D$, the optimisation problem MAX CSP(Γ) is defined as follows:

Definition 2 (MAX CSP(Γ)**)** MAX CSP(Γ) *is defined to be the optimisation problem with*

Instance: *An instance* (V, C) *of* CSP(Γ)*.*
Solution: *An assignment* $s : V \to D$ *to the variables.*
Measure: *Number of constraints in* C *satisfied by the assignment* s*.*

We use *collections* of constraints instead of just sets of constraints as we do not have any weights in our definition of MAX CSP. Some of our reductions will make use of copies of one constraint to simulate something which resembles weights. We choose to use collections instead of weights because bounded occurrence restrictions are easier to explain in the collection setting. Note that we prove our hardness results in this restricted setting without weights and with a constant bound on the number of occurrences of each variable.

Throughout the article, MAX CSP(Γ)-k will denote the problem MAX CSP(Γ) restricted to instances with the number of occurrences of each variable is bounded by k. For our hardness results we will write that MAX CSP(Γ)-B is hard (in some sense) to denote that there is a k such that MAX CSP(Γ)-k is hard in this sense. If a variable occurs t times in a constraint which appears s times in an instance, then this would contribute $t \cdot s$ to the number of occurrences of that variable in the instance.

Example 3 *Given a (multi)graph* $G = (V, E)$*, the* MAX k-CUT *problem,* $k \geq 2$*, is the problem of maximising* $|E'|$*,* $E' \subseteq E$*, such that the subgraph*

$G' = (V, E')$ is k-colourable. For $k = 2$, this problem is known simply as MAX CUT. *The problem* MAX k-CUT *is known to be* **APX***-complete for any k (it is Problem GT33 in [6]), and so has no PTAS. Let N_k denote the binary disequality relation on $\{0, 1, \ldots, k-1\}$, $k \geq 2$, that is, $(x, y) \in N_k \iff x \neq y$. To see that* MAX CSP$(\{N_k\})$ *is precisely* MAX k-CUT, *think of vertices of a given graph as of variables, and apply the relation to every pair of variables x, y such that (x, y) is an edge in the graph, with the corresponding multiplicity.*

Most of the early results on the complexity and approximability of CSP and MAX CSP were restricted to the Boolean case, i.e. when $D = \{0, 1\}$. For instance, Schaefer [54] characterised the complexity of CSP(Γ) for all Γ over the Boolean domain, the approximability of MAX CSP(Γ) for all Γ over the Boolean domain have also been determined [20,21,38]. It has been noted that the study of non-Boolean CSP seems to give a better understanding (when compared with Boolean CSP) of what makes CSP easy or hard: it appears that many observations made on Boolean CSP are special cases of more general phenomena. Recently, there has been some major progress in the understanding of non-Boolean CSP: Bulatov has provided a complete complexity classification of the CSP problem over a three-element domain [10] and also given a classification of constraint languages that contain all unary relations [8]. Corresponding results for MAX CSP have been obtained by Jonsson et al. [36] and Deineko et al. [24].

We continue this line of research by studying two aspects of non-Boolean MAX CSP. The complexity of CSP(Γ) is not known for all constraint languages Γ — it is in fact a major open question [13,29]. However, the picture is not completely unknown since the complexity of CSP(Γ) has been settled for many constraint languages [10,11,13,14,34,35].

It has been conjectured [29] that for all constraint languages Γ, CSP(Γ) is either in **P** or is **NP**-complete, and the refined conjecture [13] (which we refer to as the "algebraic CSP Conjecture", see §3.2 for details) also describes the dividing line between the two cases. Recall that if **P** $\neq$ **NP**, then Ladner's Theorem [44] states that there are problems of intermediate complexity, i.e., there are problems in **NP** that are not in **P** and not **NP**-complete. Hence, we cannot rule out a priori if there is a constraint language Γ such that CSP(Γ) is neither in **P** nor **NP**-complete. If the algebraic CSP Conjecture is true, then all **NP**-complete problems CSP(Γ) are already identified; i.e., it is the tractability part of the conjecture that is still open.

In the first part of the article we study the family of all constraint languages Γ such that it is currently known that CSP(Γ) is **NP**-complete. We prove that each constraint language in this family makes MAX CSP(Γ) have a hard gap at location 1, even when the number of variable occurrence in an instance is bounded by a sufficiently large constant (depending on Γ), see Theorem 22.

"Hard gap at location 1" means that it is **NP**-hard to distinguish instances of MAX CSP(Γ) in which all constraints are satisfiable from instances where at most an ε-fraction of the constraints are satisfiable (for some constant ε which depends on Γ) [2]. This property immediately implies approximation hardness (in particular, no PTAS) for the problem, even when restricted to satisfiable instances (Corollary 29). We note that, for the Boolean domain and without the bounded occurrence restriction, Theorem 22 follows from a result of Khanna et al. [38, Theorem 5.14].

Interestingly, the PCP theorem is equivalent to the fact that, for *some* constraint language Γ over some finite set D, MAX CSP(Γ) has a hard gap at location 1 [4,26,56]; clearly, CSP(Γ) cannot be polynomial time solvable in this case. Theorem 22 means that MAX CSP(Γ) has a hard gap at location 1 for *any* constraint language such that CSP(Γ) is known to be **NP**-complete. Moreover, if the above mentioned conjecture holds, then MAX CSP(Γ) has a hard gap at location 1 whenever CSP(Γ) is not in **P**. Another equivalent reformulation of the PCP theorem states that the problem MAX 3-SAT has a hard gap at location 1 [4,56], and our proof consists of a gap preserving reduction from this problem through a version of the algebraic argument from [13].

The second aspect of MAX CSP we study is the case when the constraint language consists of a single relation; this class of problems contains some of the best-studied examples of MAX CSP such as MAX CUT and MAX DICUT. Note that a full complexity classification of single-relation CSP is not known. In fact, Feder and Vardi [29] have proved that by providing such a classification, one has also classified the CSP problem for *all* constraint languages.

It was proved in [37] that, for any non-empty relation R, the problem MAX CSP($\{R\}$) is either trivial (i.e., mapping all variables in any instance to the same fixed value always satisfies all constraints) or **NP**-hard. We strengthen this result by proving approximation hardness (and hence the non-existence of PTAS) instead of **NP**-hardness (see Theorem 33), and again even with a bound on the number of variable occurrences. Our proof uses the first main result, Theorem 22, along with the main result from [7]. Note that, for some Boolean MAX CSP problems, e.g., for MAX CUT, a stronger version of Theorem 33 is known (see, e.g., [32]). We then apply Theorem 33 to generalise some results from [42,43].

Raghavendra [52] recently proved an interesting result regarding the approximability of MAX CSP. He constructed an approximation algorithm such that

[2] Some authors consider the promise problem GAP-CSP[ε, 1] where an instance is a MAX CSP instance (V, C) and the problem is to decide between the following two possibilities: the instance is satisfiable, or at most $\varepsilon \cdot |C|$ constraints are simultaneously satisfiable. Obviously, if a MAX CSP(Γ) has a hard gap at location 1, then there exists an ε such that the corresponding GAP-CSP[ε, 1] problem is **NP**-hard.

for any constraint language Γ the solutions produced by the algorithm is within a factor $\alpha(\Gamma) + \varepsilon$ of the optimal value, for any $\varepsilon > 0$. Furthermore, assuming the UGC and $\mathbf{P} \neq \mathbf{NP}$, he proved that for every constraint language Γ the problem MAX CSP(Γ) cannot be approximated within a factor $\alpha(\Gamma) - \varepsilon$ of the optimal value for any $\varepsilon > 0$ in polynomial time. Raghavendra's result is very strong, assuming the UGC and $\mathbf{P} \neq \mathbf{NP}$ it gives nearly tight approximability results for every constraint language. However, it does not give any direct method for characterising the classes of constraint languages which, e.g., does not admit a PTAS. Our results are less general in the sense that they apply to a smaller class of constraint languages and that they do not give near optimal approximability results. However, we study a different notion of hardness — hardness at gap location 1. Furthermore, there are explicit methods for characterising the class of constraint languages that are "hard". We also do not need any more assumptions than $\mathbf{P} \neq \mathbf{NP}$ to obtain our results.

Here is an overview of the article: In §2 we define some concepts we need. Section 3 contains the proof for our first result and §4 contains the proof of our second result. In §4.3 we strengthen some earlier published results on MAX CSP as mentioned above. We give a few concluding remarks in §5.

2 Preliminaries

A *combinatorial optimisation problem* is defined over a set of *instances* (admissible input data); each instance $\mathcal{I}$ has a set $\mathsf{sol}(\mathcal{I})$ of *feasible solutions* associated with it, and each solution $y \in \mathsf{sol}(\mathcal{I})$ has a value $m(\mathcal{I}, y)$. The objective is, given an instance $\mathcal{I}$, to find a feasible solution of optimum value. The optimal value is the largest one for *maximisation* problems and the smallest one for *minimisation* problems. A combinatorial optimisation problem is said to be an **NP** optimisation (**NPO**) problem if its instances and solutions can be recognised in polynomial time, the solutions are polynomially-bounded in the input size, and the objective function can be computed in polynomial time (see, e.g., [6]).

Definition 4 (Performance ratio) *A solution $s \in \mathsf{sol}(\mathcal{I})$ to an instance $\mathcal{I}$ of an* **NPO** *maximization problem Π is r-approximate if*

$$\max \left\{ \frac{m(\mathcal{I}, s)}{\textsc{opt}(\mathcal{I})}, \frac{\textsc{opt}(\mathcal{I})}{m(\mathcal{I}, s)} \right\} \leq r,$$

where $\textsc{opt}(\mathcal{I})$ is the optimal value for a solution to $\mathcal{I}$. An approximation algorithm for an **NPO** *problem Π has* performance ratio *$R(n)$ if, given any instance $\mathcal{I}$ of Π with $|\mathcal{I}| = n$, it outputs an $R(n)$-approximate solution.*

PO is the class of **NPO** problems that can be solved (to optimality) in polynomial time. An **NPO** problem Π is in the class **APX** if there is a polynomial time approximation algorithm for Π whose performance ratio is bounded by a constant. The following result is well-known (see, e.g., [17, Proposition 2.3]).

Lemma 5 *Let D be a finite set. For every constraint language $\Gamma \subseteq R_D$,* MAX CSP(Γ) *belongs to* **APX**. *Moreover, if a is the maximum arity of any relation in Γ, then there is a polynomial time approximation algorithm with performance ratio $|D|^a$*

Definition 6 (Hard to approximate) *We say that a problem Π is* hard to approximate *if there exists a constant c such that, Π is* **NP**-*hard to approximate within c (that is, the existence of a polynomial-time approximation algorithm for Π with performance ratio c implies* **P** = **NP**).

The following notion has been defined in a more general setting by Petrank [50].

Definition 7 (Hard gap at location α) MAX CSP(Γ) *has a* hard gap at location $\alpha \leq 1$ *if there exists a constant $\varepsilon < \alpha$ and a polynomial-time reduction from an* **NP**-*complete problem Π to* MAX CSP(Γ) *such that,*

- YES *instances of Π are mapped to instances $\mathcal{I} = (V, C)$ such that* $\mathrm{OPT}(\mathcal{I}) \geq \alpha|C|$, *and*
- NO *instances of Π are mapped to instances $\mathcal{I} = (V, C)$ such that* $\mathrm{OPT}(\mathcal{I}) \leq \varepsilon|C|$.

Note that if a problem Π has a hard gap at location α (for any α) then Π is hard to approximate. This simple observation has been used to prove inapproximability results for a large number of optimisation problems. See, e.g., [3,6,56] for surveys on inapproximability results and the related PCP theory.

2.1 Approximation Preserving Reductions

To prove our approximation hardness results we use *AP-reductions*. This type of reduction is most commonly used to define completeness for certain classes of optimisation problems (i.e., **APX**). However, no **APX**-hardness results are actually proven in this article since we concentrate on proving that problems are hard to approximate (in the sense of Definition 6). We will frequently use *AP*-reductions and this is justified by Lemma 9 below. Our definition of *AP*-reductions follows [21,38].

Definition 8 (*AP*-reduction) *Given two* **NPO** *problems Π_1 and Π_2 an AP-*

reduction *from* Π_1 *to* Π_2 *is a triple* (F, G, α) *such that,*

- F *and* G *are polynomial-time computable functions and* $\alpha > 0$ *is a constant;*
- *for any instance* $\mathcal{I}$ *of* Π_1*,* $F(\mathcal{I})$ *is an instance of* Π_2*;*
- *for any instance* $\mathcal{I}$ *of* Π_1*, and any feasible solution* s' *of* $F(\mathcal{I})$*,* $G(\mathcal{I}, s')$ *is a feasible solution of* $\mathcal{I}$*;*
- *for any instance* $\mathcal{I}$ *of* Π_1*, and any* $r \geq 1$*, if* s' *is an* r*-approximate solution of* $F(\mathcal{I})$ *then* $G(\mathcal{I}, s')$ *is an* $(1 + (r-1)\alpha + o(1))$*-approximate solution of* $\mathcal{I}$ *where the* o*-notation is with respect to* $|\mathcal{I}|$*.*

If such a triple exist we say that Π_1 *is* AP*-reducible to* Π_2*. We use the notation* $\Pi_1 \leq_{AP} \Pi_2$ *to denote this fact.*

It is a well-known fact (see, e.g., §8.2.1 in [6]) that AP-reductions compose. The following simple lemma makes AP-reductions useful to us.

Lemma 9 *If* $\Pi_1 \leq_{AP} \Pi_2$ *and* Π_1 *is hard to approximate, then* Π_2 *is hard to approximate.*

Proof. Let $c > 1$ be the constant such that it is **NP**-hard to approximate Π_1 within c. Let (F, G, α) be the AP-reduction which reduces Π_1 to Π_2. We will prove that it is **NP**-hard to approximate Π_2 within

$$r = \frac{1}{\alpha}(c-1) + 1 - \varepsilon'$$

for any $\varepsilon' > 0$.

Let $\mathcal{I}_1$ be an instance of Π_1. Then, $\mathcal{I}_2 = F(\mathcal{I}_1)$ is an instance of Π_2. Given an r-approximate solution to $\mathcal{I}_2$ we can construct an $(1 + (r-1)\alpha + o(1))$-approximate solution to $\mathcal{I}_1$ using G. Hence, we get an $1 + (r-1)\alpha + o(1) = c - \alpha\varepsilon' + o(1)$ approximate solution to $\mathcal{I}_1$, and when the instances are large enough this is strictly smaller than c. As $c > 1$ we can choose ε' such that $\varepsilon' > 0$ and $c - \alpha\varepsilon' > 1$. □

2.2 Reduction Techniques

The basic reduction technique in our approximation hardness proofs is based on *strict implementations* and *perfect implementations*. Those techniques have been used before when studying Max CSP and other CSP-related problems [21,36,38].

Definition 10 (Implementation) *A collection of constraints* $C_1, \ldots, C_m$ *over a tuple of variables* $\boldsymbol{x} = (x_1, \ldots, x_p)$ *called* primary variables *and* $\boldsymbol{y} = (y_1, \ldots, y_q)$

called auxiliary variables *is an* α-implementation *of the* p*-ary relation* R *for a positive integer* $\alpha \leq m$ *if the following conditions are satisfied:*

(1) For any assignment to $\boldsymbol{x}$ *and* $\boldsymbol{y}$*, at most* α *constraints from* $C_1, \ldots, C_m$ *are satisfied.*
(2) For any $\boldsymbol{x}$ *such that* $\boldsymbol{x} \in R$*, there exists an assignment to* $\boldsymbol{y}$ *such that exactly* α *constraints are satisfied.*
(3) For any $\boldsymbol{x}, \boldsymbol{y}$ *such that* $\boldsymbol{x} \notin R$*, at most* $(\alpha - 1)$ *constraints are satisfied.*

Definition 11 (Strict/Perfect Implementation) *An* α*-implementation is a* strict implementation *if for every* $\boldsymbol{x}$ *such that* $\boldsymbol{x} \notin R$ *there exists* $\boldsymbol{y}$ *such that exactly* $(\alpha - 1)$ *constraints are satisfied. An* α*-implementation (not necessarily strict) is a* perfect implementation *if* $\alpha = m$*.*

It will sometimes be convenient for us to view relations as predicates instead. In this case an n-ary relation R over the domain D is a function $r : D^n \to \{0, 1\}$ such that $r(\boldsymbol{x}) = 1 \iff \boldsymbol{x} \in R$. Most of the time we will use predicates when we are dealing with strict implementations and relations when we are working with perfect implementations, because perfect implementations are naturally written as a conjunction of constraints whereas strict implementations may naturally be seen as a sum of predicates. We will write strict α-implementations in the following form

$$g(\boldsymbol{x}) + (\alpha - 1) = \max_{\boldsymbol{y}} \sum_{i=1}^{m} g_i(\boldsymbol{x}_i)$$

where $\boldsymbol{x} = (x_1, \ldots, x_p)$ are the primary variables, $\boldsymbol{y} = (y_1, \ldots, y_q)$ are the auxiliary variables, $g(\boldsymbol{x})$ is the predicate which is implemented, and each $\boldsymbol{x}_i$ is a tuple of variables from $\boldsymbol{x}$ and $\boldsymbol{y}$.

We say that a collection of relations Γ *strictly (perfectly) implements* a relation R if, for some $\alpha \in \mathbb{Z}^+$, there exists a strict (perfect) α-implementation of R using relations only from Γ. It is not difficult to show that if R can be obtained from Γ by a series of strict (perfect) implementations, then it can also be obtained by a single strict (perfect) implementation (for the Boolean case, this is shown in [21, Lemma 5.8]).

The following lemma indicates the importance of strict implementations for Max CSP. It was first proved for the Boolean case, but without the assumption on bounded occurrences, in [21, Lemma 5.17]. A proof of this lemma in our setting can be found in [24, Lemma 3.4] (the lemma is stated in a slightly different form but the proof establishes the required AP-reduction).

Lemma 12 *If* Γ *strictly implements a predicate* f*, then, for any integer* k*, there is an integer* k' *such that* Max CSP$(\Gamma \cup \{f\})$-$k \leq_{AP}$ Max CSP(Γ)-k'*.*

Lemma 12 will be used as follows in our proofs of approximation hardness: if Γ' is a fixed finite collection of predicates each of which can be strictly implemented by Γ, then we can assume that $\Gamma' \subseteq \Gamma$. For example, if Γ contains a binary predicate f, then we can assume, at any time when it is convenient, that Γ also contains $f'(x, y) = f(y, x)$, since this equality is a strict 1-implementation of f'.

For proving hardness at gap location 1, we have the following lemma.

Lemma 13 *If a finite constraint language Γ perfectly implements a relation R and* Max CSP$(\Gamma \cup \{R\})$-k *has a hard gap at location 1, then* Max CSP(Γ)-k' *has a hard gap at location 1 for some integer k'.*

Proof. Let N be the minimum number of relations that are needed in a perfect implementation of R using relations from Γ.

Given an instance $\mathcal{I} = (V, C)$ of Max CSP$(\Gamma \cup \{R\})$-k, we construct an instance $\mathcal{I}' = (V', C')$ of Max CSP(Γ)-k' (where k' will be specified below) as follows: we use the set V'' to store auxiliary variables during the reduction so we initially let V'' be the empty set. For a constraint $c = (Q, \boldsymbol{s}) \in C$, there are two cases to consider:

(1) If $Q \neq R$, then add N copies of c to C'.
(2) If $Q = R$, then add the implementation of R to C' where any auxiliary variables in the implementation are replaced with fresh variables which are added to V''.

Finally, let $V' = V \cup V''$. It is clear that there exists an integer k', independent of $\mathcal{I}$, such that $\mathcal{I}'$ is an instance of Max CSP(Γ')-k'.

If all constraints are simultaneously satisfiable in $\mathcal{I}$, then all constraints in $\mathcal{I}'$ are also simultaneously satisfiable. On the other hand, if $\textsc{opt}(\mathcal{I}) \leq \varepsilon|C|$ then

$$\begin{aligned} \textsc{opt}(\mathcal{I}') &\leq \varepsilon N|C| + (1-\varepsilon)(N-1)|C| \\ &= (\varepsilon + (1-\varepsilon)(1-1/N))\,|C'|. \end{aligned}$$

The inequality holds because each constraint in $\mathcal{I}$ introduces a group of N constraints in $\mathcal{I}'$ and, as $\textsc{opt}(\mathcal{I}) \leq \varepsilon|C|$, at most $\varepsilon|C|$ such groups are completely satisfied. In all other groups (there are $(1-\varepsilon)|C|$ such groups) at least one constraint is not satisfied. We conclude that Max CSP(Γ)-k' has a hard gap at location 1. $\square$

An important concept is that of a *core*. To define cores formally we need retractions. A *retraction* of a constraint language $\Gamma \subseteq R_D$ is a function π :

$D \to D$ such that if D' is the image of π then $\pi(x) = x$ for all $x \in D'$, furthermore for every $R \in \Gamma$ we have $(\pi(t_1), \ldots, \pi(t_n)) \in R$ for all $(t_1, \ldots, t_n) \in R$. We will say that Γ is a *core* if the only retraction of Γ is the identity function. Given a relation $R \in R_D^{(k)}$ and a subset X of D we define the *restriction of R onto X* as follows: $R\big|_X = \{\boldsymbol{x} \in X^k \mid \boldsymbol{x} \in R\}$. For a set of relations Γ we define $\Gamma\big|_X = \{R\big|_X \mid R \in \Gamma\}$. If π is a retraction of Γ with image D', chosen such that $|D'|$ is minimal, then a core of Γ is the set $\Gamma\big|_{D'}$. For constraint language Γ, Γ' we say that Γ *retracts to* Γ' if there is a retraction π of Γ such that $\pi(\Gamma) = \Gamma'$.

The intuition here is that if Γ is not a core, then it has a non-injective retraction π, which implies that, for every assignment s, there is another assignment πs that satisfies all constraints satisfied by s and uses only a restricted set of values. Consequently the problem is equivalent to a problem over this smaller set. As in the case of graphs, all cores of Γ are isomorphic, so one can speak about *the* core of Γ. [31]

The following simple lemma connects cores with non-approximability.

Lemma 14 *If Γ' is the core of Γ, then, for any k,* Max CSP(Γ')-k *has a hard gap at location 1 if and only if* Max CSP(Γ)-k *has a hard gap at location 1.*

Proof. Let π be the retraction of Γ such that $\Gamma' = \{\pi(R) \mid R \in \Gamma\}$, where $\pi(R) = \{\pi(\boldsymbol{t}) \mid \boldsymbol{t} \in R\}$. Given an instance $\mathcal{I} = (V, C)$ of Max CSP(Γ)-k, we construct an instance $\mathcal{I}' = (V, C')$ of Max CSP(Γ')-k by replacing each constraint $(R, \boldsymbol{s}) \in C$ by $(\pi(R), \boldsymbol{s})$.

From a solution s to $\mathcal{I}'$, we construct a solution s' to $\mathcal{I}'$ such that $s'(x) = \pi(s(x))$. Let $(R, \boldsymbol{s}) \in C$ be a constraint which is satisfied by s. Then, there is a tuple $\boldsymbol{x} \in R$ such that $s(\boldsymbol{s}) = \boldsymbol{x}$ so $\pi(\boldsymbol{x}) \in \pi(R)$ and $s'(\boldsymbol{s}) = \pi(s(\boldsymbol{s})) = \pi(\boldsymbol{x}) \in \pi(R)$. Conversely, if $(\pi(R), \boldsymbol{s})$ is a constraint in $\mathcal{I}'$ which is satisfied by s', then there is a tuple $\boldsymbol{x} \in R$ such that $s'(\boldsymbol{s}) = \pi(s(\boldsymbol{s})) = \pi(\boldsymbol{x}) \in \pi(R)$, and $s(\boldsymbol{s}) = \boldsymbol{x} \in R$. We conclude that $m(\mathcal{I}, s) = m(\mathcal{I}', s')$.

It is not hard to see that we can do this reduction in the other way too, i.e., given an instance $\mathcal{I}' = (V', C')$ of Max CSP(Γ')-k, we construct an instance $\mathcal{I}$ of Max CSP(Γ)-k by replacing each constraint $(\pi(R), \boldsymbol{s}) \in C'$ by $(R, \boldsymbol{s})$. By the same argument as above, this direction of the equivalence follows, and we conclude that the lemma is valid. □

An analogous result holds for the CSP problem, i.e., if Γ' is the core of Γ, then CSP(Γ) is in **P** (**NP**-complete) if and only if CSP(Γ') is in **P** (**NP**-complete); see [34] for a proof. Cores play an important role in §4, too. We have the following lemma:

Lemma 15 (Lemma 2.11 in [36]) *Let Γ' be the core of Γ. For every k, there exists k' such that* Max CSP(Γ')-$k \leq_{AP}$ Max CSP(Γ)-k'.

The lemma is stated in a slightly different form in [36] but the proof establishes the required AP-reduction.

3 Hardness at Gap Location 1 for Max CSP

In this section, we prove our first main result: Theorem 22. The proof makes use of some concepts from universal algebra and we present the relevant definitions and results in §3.1 and §3.2. The proof is contained in §3.3.

3.1 Definitions and Results from Universal Algebra

We will now present the definitions and basic results we need from universal algebra. For a more thorough treatment of universal algebra in general we refer the reader to [15,19]. The articles [13,18] contain presentations of the relationship between universal algebra and constraint satisfaction problems.

An *operation* on a finite set D is an arbitrary function $f : D^k \to D$. Any operation on D can be extended in a standard way to an operation on tuples over D, as follows: let f be a k-ary operation on D. For any collection of k n-tuples, $\boldsymbol{t_1}, \boldsymbol{t_2}, \ldots, \boldsymbol{t_k} \in D^n$, the n-tuple $f(\boldsymbol{t_1}, \boldsymbol{t_2}, \ldots, \boldsymbol{t_k})$ is defined as follows:

$$\begin{aligned} f(\boldsymbol{t_1}, \boldsymbol{t_2}, \ldots, \boldsymbol{t_k}) = (&f(\boldsymbol{t_1}[1], \boldsymbol{t_2}[1], \ldots, \boldsymbol{t_k}[1]), f(\boldsymbol{t_1}[2], \boldsymbol{t_2}[2], \ldots, \boldsymbol{t_k}[2]), \ldots, \\ &f(\boldsymbol{t_1}[n], \boldsymbol{t_2}[n], \ldots, \boldsymbol{t_k}[n])), \end{aligned}$$

where $\boldsymbol{t_j}[i]$ is the i-th component in tuple $\boldsymbol{t_j}$. If $f(d, d, \ldots, d) = d$ for all $d \in D$, then f is said to be *idempotent*. An operation $f : D^k \to D$ which satisfies $f(x_1, x_2, \ldots, x_k) = x_i$, for some i, is called a *projection*.

Let R be a relation in the constraint language Γ. If f is an operation such that for all $\boldsymbol{t_1}, \boldsymbol{t_2}, \ldots, \boldsymbol{t_k} \in R$ we have $f(\boldsymbol{t_1}, \boldsymbol{t_2}, \ldots, \boldsymbol{t_k}) \in R$, then R is said to be *invariant* (or, in other words, closed) under f. If all constraint relations in Γ are invariant under f, then Γ is said to be invariant under f. An operation f such that Γ is invariant under f is called a *polymorphism* of Γ. The set of all polymorphisms of Γ is denoted $\mathrm{Pol}(\Gamma)$. Given a set of operations F, the set of all relations that is invariant under all the operations in F is denoted $\mathrm{Inv}(F)$.

Example 16 *Let $D = \{0, 1, 2\}$ and let R be the directed cycle on D, i.e., $R = \{(0, 1), (1, 2), (2, 0)\}$. One polymorphism of R is the operation $f : \{0, 1, 2\}^3 \to \{0, 1, 2\}$ defined as $f(x, y, z) = x - y + z \pmod 3$. This can be verified by* *considering all possible combinations of three tuples from R and evaluating f component-wise. Let K be the complete graph on D. It is well known and not hard to check that if we view K as a binary relation, then all idempotent polymorphisms of K are projections.*

We continue by defining a closure operator $\langle\cdot\rangle$ on sets of relations: for any set $\Gamma \subseteq R_D$, the set $\langle\Gamma\rangle$ consists of all relations that can be expressed using relations from $\Gamma \cup \{EQ_D\}$ (where EQ_D denotes the equality relation on D), conjunction, and existential quantification. Those are the relations definable by *primitive positive formulae* (pp-formulae). As an example of a pp-formula consider the relations $A = \{(0,0),(0,1),(1,0)\}$ and $B = \{(1,0),(0,1),(1,1)\}$ over the Boolean domain $\{0,1\}$. With those two relations we can construct $I = \{(0,0),(0,1),(1,1)\}$ with the pp-formula

$$I(x,y) \iff \exists z : A(x,z) \wedge B(z,y).$$

Note that pp-formulae and perfect implementations from Definition 11 are the same concept. Intuitively, constraints using relations from $\langle\Gamma\rangle$ are exactly those which can be simulated by constraints using relations from Γ in the CSP problem. Hence, for any finite subset Γ' of $\langle\Gamma\rangle$, CSP(Γ') is not harder than CSP(Γ). That is, if CSP(Γ') is **NP**-complete for some finite subset Γ' of $\langle\Gamma\rangle$, then CSP(Γ) is **NP**-complete. If CSP(Γ) is in **P**, then CSP(Γ') is in **P** for every finite subset Γ' of $\langle\Gamma\rangle$. We refer the reader to [35] for a further discussion on this topic.

The sets of relations of the form $\langle\Gamma\rangle$ are referred to as *relational clones*, or *co-clones*. An alternative characterisation of relational clones is given in the following theorem.

Theorem 17 ([51])

- *For every set* $\Gamma \subseteq R_D$, $\langle\Gamma\rangle = \mathrm{Inv}(\mathrm{Pol}(\Gamma))$.
- *If* $\Gamma' \subseteq \langle\Gamma\rangle$, *then* $\mathrm{Pol}(\Gamma) \subseteq \mathrm{Pol}(\Gamma')$.

We will now define finite algebras and some related notions which we need later on. The three definitions below closely follow the presentation in [13].

Definition 18 (Finite algebra) *A* finite algebra *is a pair $\mathcal{A} = (A; F)$ where A is a finite non-empty set and F is a set of finitary operations on A.*

We will only make use of finite algebras so we will write *algebra* instead of *finite algebra*. An algebra is said to be *non-trivial* if it has more than one element.

Definition 19 (Homomorphism of algebras) *Given two algebras $\mathcal{A} = (A; F_A)$ and $\mathcal{B} = (B; F_B)$ such that $F_A = \{f_i^A \mid i \in I\}$, $F_B = \{f_i^B \mid i \in I\}$ and both f_i^A*

and f_i^B *are* n_i*-ary for all* $i \in I$*, then* $\varphi : A \to B$ *is said to be an* homomorphism *from* $\mathcal{A}$ *to* $\mathcal{B}$ *if*

$$\varphi(f_i^A(a_1, a_2, \ldots, a_{n_i})) = f_i^B(\varphi(a_1), \varphi(a_2), \ldots, \varphi(a_{n_i}))$$

for all $i \in I$ *and* $a_1, a_2, \ldots, a_{n_i} \in A$*. If* φ *is surjective, then* $\mathcal{B}$ *is a* homomorphic image *of* $\mathcal{A}$*.*

Given a homomorphism φ mapping $\mathcal{A} = (A; F_A)$ to $\mathcal{B} = (B; F_B)$, we can construct an equivalence relation θ on A as $\theta = \{(x, y) \mid \varphi(x) = \varphi(y)\}$. The relation θ is said to be a *congruence relation* of $\mathcal{A}$. We can now construct the *quotient algebra* $\mathcal{A}/\theta = (A/\theta; F_A/\theta)$. Here, $A/\theta = \{x/\theta \mid x \in A\}$ and x/θ is the equivalence class containing x. Furthermore, $F_A/\theta = \{f/\theta \mid f \in F_A\}$ and f/θ is defined such that $f/\theta(x_1/\theta, x_2/\theta, \ldots, x_n/\theta) = f(x_1, x_2, \ldots, x_n)/\theta$.

For an operation $f : D^n \to D$ and a subset $X \subseteq D$ we define $f\big|_X$ as the function $g : X^n \to D$ such that $g(\boldsymbol{x}) = f(\boldsymbol{x})$ for all $\boldsymbol{x} \in X^n$. For a set of operations F on D we define $F\big|_X = \{f\big|_X \mid f \in F\}$.

Definition 20 (Subalgebra) *Let* $\mathcal{A} = (A; F_A)$ *be an algebra and* $B \subseteq A$*. If for each* $f \in F_A$ *and any* $b_1, b_2, \ldots, b_n \in B$*, we have* $f(b_1, b_2, \ldots, b_n) \in B$*, then* $\mathcal{B} = (B; F_A\big|_B)$ *is a* subalgebra *of* $\mathcal{A}$*.*

The operations in $\mathrm{Pol}(\mathrm{Inv}(F_A))$ are the *term operations* of $\mathcal{A}$. If all term operations are surjective, then the algebra is said to be *surjective.* Note that $\mathrm{Inv}(F_A)$ is a core if and only if $\mathcal{A}$ is surjective [13,34]. If F consist of all the idempotent term operations of $\mathcal{A}$, then the algebra $(A; F)$ is called the *full idempotent reduct of* $\mathcal{A}$, and we will denote this algebra by $\mathcal{A}^c$. Given a set of relations Γ over the domain D we say that the algebra $\mathcal{A}_\Gamma = (D; \mathrm{Pol}(\Gamma))$ is *associated* with Γ. An algebra $\mathcal{B}$ is said to be a *factor* of the algebra $\mathcal{A}$ if $\mathcal{B}$ is a homomorphic image of a subalgebra of $\mathcal{A}$. A *non-trivial factor* is an algebra which is not trivial, i.e., it has at least two elements.

3.2 Constraint Satisfaction and Algebra

We continue by describing some connections between constraint satisfaction problems and universal algebra. The following theorem concerns the hardness of CSP for certain constraint languages.

Theorem 21 ([13]) *Let* Γ *be a core constraint language. If* $\mathcal{A}_\Gamma^c$ *has a non-trivial factor whose term operations are only projections, then* $\mathrm{CSP}(\Gamma)$ *is* **NP***-hard.*

The algebraic CSP conjecture [13] states that, for all other core languages Γ, the problem CSP(Γ) is tractable. This conjecture has been verified in many important cases (see, e.g., [8,10]).

The first main result of this article is the following theorem which states that MAX CSP(Γ)-B has a hard gap at location 1 whenever the condition which makes CSP(Γ) hard in Theorem 21 is satisfied.

Theorem 22 *Let Γ be a core constraint language. If $\mathcal{A}_\Gamma^c$ has a non-trivial factor whose term operations are only projections, then* MAX CSP(Γ)-B *has a hard gap at location 1.*

The proof of this result can be found in §3.3. Note that if the above conjecture is true then Theorem 22 describes all constraint languages Γ for which MAX CSP(Γ) has a hard gap at location 1 because, obviously, Γ cannot have this property when CSP(Γ) is tractable.

There is another characterisation of the algebras in Theorem 21 which corresponds to tractable constraint languages. To state the characterisation we need the following definition.

Definition 23 (Weak Near-Unanimity Function) *An operation $f : D^n \to D$, where $n \geq 2$, is a* weak near-unanimity function *if f is idempotent and*

$$f(x, y, y, \ldots, y) = f(y, x, y, y, \ldots, y) = \ldots = f(y, \ldots, y, x)$$

for all $x, y \in D$.

Hereafter we will use the acronym *wnuf* for weak near-unanimity functions. We say that an algebra $\mathcal{A}$ *admits a wnuf* if there is a wnuf among the term operations of $\mathcal{A}$. We also say that a constraint language Γ admits a wnuf if there is a wnuf among the polymorphisms of Γ. By combining a theorem by Maróti and McKenzie [48, Theorem 1.1] with a result by Bulatov and Jeavons [12, Proposition 4.14], we get the following:

Theorem 24 *Let $\mathcal{A}$ be an idempotent algebra. The following are equivalent:*

- *There is a non-trivial factor $\mathcal{B}$ of $\mathcal{A}$ such that $\mathcal{B}$ only has projections as term operations.*
- *The algebra $\mathcal{A}$ does not admit any wnuf.*

3.3 Proof of Theorem 22

Let $3SAT_0$ denote the relation $\{0,1\}^3 \setminus \{(0,0,0)\}$. We also introduce three slight variations of $3SAT_0$, let $3SAT_1 = \{0,1\}^3 \setminus \{(1,0,0)\}$, $3SAT_2 = \{0,1\}^3 \setminus$

$\{(1,1,0)\}$, and $3SAT_3 = \{0,1\}^3 \setminus \{(1,1,1)\}$. To simplify the notation we let $\Gamma_{3SAT} = \{3SAT_0,\ 3SAT_1,\ 3SAT_2,\ 3SAT_3\}$. It is not hard to see that the problem MAX CSP(Γ_{3SAT}) is precisely MAX 3SAT. It is well-known that this problem, even when restricted to instances in which each variable occurs at most a constant number of times, has a hard gap at location 1, see e.g., [56, Theorem 7]. We state this as a lemma.

Lemma 25 ([56]) MAX CSP(Γ_{3SAT})-B *has a hard gap at location 1.*

To prove Theorem 22 we will utilise *expander graphs.*

Definition 26 (Expander graph) *A d-regular graph G is an* expander graph *if, for any $S \subseteq V[G]$, the number of edges between S and $V[G] \setminus S$ is at least* $\min(|S|, |V[G] \setminus S|)$.

Expander graphs are frequently used for proving properties of MAX CSP, cf. [22,49]. Typically, they are used for bounding the number of variable occurrences. A concrete construction of expander graphs has been provided by Lubotzky et al. [47].

Theorem 27 *A polynomial-time algorithm T and a fixed integer N exist such that, for any $k > N$, $T(k)$ produces a 14-regular expander graph with $k(1+o(1))$ vertices.*

There are four basic ingredients in the proof of Theorem 22. The first three are Lemma 13, Lemma 25, and the use of expander graphs to bound the number of variable occurrences. We also use an alternative characterisation (Lemma 28) of constraint languages satisfying the conditions of the theorem. This is a slight modification of a part of the proof of Proposition 7.9 in [13]. The implication below is in fact an equivalence and we refer the reader to [13] for the details. Given a function $f : D \to D$, and a relation $R \in R_D$, the *full preimage* of R under f, denoted by $f^{-1}(R)$, is the relation $\{\boldsymbol{x} \mid f(\boldsymbol{x}) \in R\}$ (as usual, $f(\boldsymbol{x})$ denotes that f should be applied componentwise to $\boldsymbol{x}$). For any $a \in D$, we denote the unary constant relation containing only a by c_a, i.e., $c_a = \{(a)\}$. Let C_D denote the set of all constant relations over D, that is, $C_D = \{c_a \mid a \in D\}$.

Lemma 28 *Let Γ be a core constraint language. If the algebra $\mathcal{A}_\Gamma^{\mathrm{c}}$ has a non-trivial factor whose term operations are only projections, then there is a subset B of D and a surjective mapping $\varphi : B \to \{0,1\}$ such that the relational clone $\langle \Gamma \cup C_D \rangle$ contains the relations $\varphi^{-1}(3SAT_0)$, $\varphi^{-1}(3SAT_1)$, $\varphi^{-1}(3SAT_2)$, and $\varphi^{-1}(3SAT_3)\}$.*

Proof. Let $\mathcal{A}'$ be the subalgebra of $\mathcal{A}_\Gamma^{\mathrm{c}}$ such that there is a homomorphism φ from $\mathcal{A}'$ to a non-trivial algebra $\mathcal{B}$ whose term operations are only projections. We can assume, without loss of generality, that the set $\{0,1\}$ is contained in the universe of $\mathcal{B}$. It is easy to see that any relation is invariant under any projections. Since $\mathcal{B}$ only has projections as term operations, the four relations $3SAT_0$, $3SAT_1$, $3SAT_2$ and $3SAT_3$ are invariant under the term operations of $\mathcal{B}$. It is not hard to check (see [13]) that the full preimages of those relations under φ are invariant under the term operations of $\mathcal{A}'$ and therefore they are also invariant under the term operations of $\mathcal{A}_\Gamma^c$. By the observation that $\mathcal{A}_\Gamma^c = \mathcal{A}_{\Gamma \cup C_D}$ and Theorem 17, this implies $\{\varphi^{-1}(3SAT_0), \varphi^{-1}(3SAT_1), \varphi^{-1}(3SAT_2), \varphi^{-1}(3SAT_3)\} \subseteq \langle \Gamma \cup C_D \rangle$. $\square$

We are now ready to present the proof of Theorem 22. Let S be a permutation group on the set X. An *orbit* of S is a subset Ω of X such that $\Omega = \{g(x) \mid g \in S\}$ for some $x \in X$.

Proof. By Lemma 13, in order to prove the theorem, it suffices to find a finite set $\Gamma' \subseteq \langle \Gamma \rangle$ such that Max CSP(Γ')-B has a hard gap at location 1.

Since Γ is a core, its unary polymorphisms form a permutation group S on D. We can without loss of generality assume that $D = \{1, \ldots, p\}$. It is known (see Proposition 1.3 of [55]) and not hard to check (using Theorem 17) that Γ can perfectly implement the following relation: $R_S = \{(g(1), \ldots, g(p)) \mid g \in S\}$. Then it can also perfectly implement the relations EQ_i for $1 \leq i \leq p$ where EQ_i is the restriction of the equality relation on D to the orbit in S which contains i. We have

$$\begin{aligned} EQ_i(x,y) \iff \exists z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_p : & R_S(z_1, \ldots, z_{i-1}, x, z_{i+1}, \ldots, z_p) \wedge \\ & R_S(z_1, \ldots, z_{i-1}, y, z_{i+1}, \ldots, z_p). \end{aligned}$$

By Lemma 28, there exists a subset (in fact, a subalgebra) B of D and a surjective mapping $\varphi : B \to \{0,1\}$ such that the relational clone $\langle \Gamma \cup C_D \rangle$ contains $\varphi^{-1}(\Gamma_{3SAT}) = \{\varphi^{-1}(R) \mid R \in \Gamma_{3SAT}\}$. For $0 \leq i \leq 3$, let R_i be the preimage of $3SAT_i$ under φ. Since $R_i \in \langle \Gamma \cup C_D \rangle$, we can show that there exists a $(p+3)$-ary relation R_i' in $\langle \Gamma \rangle$ such that

$$R_i = \{(x,y,z) \mid (1, 2, \ldots, p, x, y, z) \in R_i'\}.$$

Indeed, since $R_i \in \langle \Gamma \cup C_D \rangle$, R_i can be defined by a pp-formula $R_i(x,y,z) \iff \exists \mathbf{t} : \psi(\mathbf{t}, x, y, z)$ (here $\mathbf{t}$ denotes a tuple of variables) where ψ is a conjunction of atomic formulas involving predicates from $\Gamma \cup C_D$ and variables from $\mathbf{t}$ and $\{x, y, z\}$. Note that, in ψ, no predicate from C_D is applied to one of $\{x, y, z\}$ because these variables can take more than one value in R_i. We can without loss of generality assume that every predicate from C_D appears in ψ exactly once. Indeed, if such a predicate appears more than once, then we can identify all variables to which it is applied, and if it does not appear at all then we can add a new variable to $\mathbf{t}$ and apply this predicate to it. Now assume without loss of generality that the predicate c_i, $1 \leq i \leq p$, is applied to the variable t_i in ψ, and $\psi = \psi_1 \wedge \psi_2$ where $\psi_1 = \bigwedge_{i=1}^{p} c_i(t_i)$ and ψ_2 contains only predicates from $\Gamma \setminus C_D$. Let $\mathbf{t}'$ be the list of variables obtained from $\mathbf{t}$ by removing $t_1, \ldots, t_p$. It now is easy to check that that the $(p+3)$-ary relation R'_i defined by the pp-formula $\exists \mathbf{t}' : \psi_2(\mathbf{t}, x, y, z)$ has the required property.

Choose R'_i to be the (inclusion-wise) minimal relation in $\langle \Gamma \rangle$ such that

$$R_i = \{(x, y, z) \mid (1, 2, \ldots, p, x, y, z) \in R'_i\}$$

and let $\Gamma' = \{R'_i \mid 0 \leq i \leq 3\} \cup \{EQ_1, \ldots, EQ_p\}$. Note that we have $\Gamma' \subseteq \langle \Gamma \rangle$.

We will need a more concrete description of R'_i, so we now show that

$$R'_i = \{(g(1), g(2), \ldots, g(p), g(x), g(y), g(z)) \mid g \in S, (x, y, z) \in R_i\}.$$

The set on the right-hand side of the above equality must be contained in R'_i because R'_i is invariant under all operations in S. On the other hand, if a tuple $\mathbf{b} = (b_1, \ldots, b_p, d, e, f)$ belongs to R'_i, then there is a permutation $g \in S$ such that $(b_1, \ldots, b_p) = (g(1), \ldots, g(p))$ (otherwise, the intersection of this relation with $R_S \times D^3 \in \langle \Gamma \rangle$ would give a smaller relation with the required property). Now note that the tuple $(1, \ldots, p, g^{-1}(d), g^{-1}(e), g^{-1}(f))$ also belongs to R'_i implying, by the choice of R'_i, that $(g^{-1}(d), g^{-1}(e), g^{-1}(f)) \in R_i$. Therefore, the relation R'_i is indeed as described above.

By Lemma 25, there is an integer l such that MAX CSP(Γ_{3SAT})-l has a hard gap at location 1. By Lemma 14, MAX CSP($\varphi^{-1}(\Gamma_{3SAT})$)-$l$ has the same property (because Γ_{3SAT} is the core of $\varphi^{-1}(\Gamma_{3SAT})$). To complete the proof, we will now AP-reduce MAX CSP($\varphi^{-1}(\Gamma_{3SAT})$)-$l$ to MAX CSP(Γ')-l' where $l' = \max\{14p + 1, l\}$ (recall that $p = |D|$ is a constant). Take an arbitrary instance $\mathcal{I} = (V, C)$ of MAX CSP($\varphi^{-1}(\Gamma_{3SAT})$)-$l$, and build an instance $\mathcal{I}' = (V', C')$ of MAX CSP(Γ') as follows: introduce new variables $u_1, \ldots, u_p$, and replace each constraint $R_i(x, y, z)$ in $\mathcal{I}$ by $R'_i(u_1, \ldots, u_p, x, y, z)$. Note that every variable, except the u_i's, in $\mathcal{I}'$ appears at most l times. We will now use expander graphs to construct an instance $\mathcal{I}''$ of MAX CSP(Γ') with a constant bound on the number of occurrences for each variables.

Let q be the number of constraints in $\mathcal{I}$ and let $q' = \max\{N, q\}$, where N is the constant in Theorem 27. Let $G = (W, E)$ be an expander graph (constructed in polynomial time by the algorithm $T(q')$ in Theorem 27) such that $W = \{w_1, w_2, \ldots, w_m\}$ and $m \geq q$. The expander graph $T(q')$ has $q'(1 + o(1))$ vertices. Hence, there is a constant α such that $T(q')$ has at most αq vertices. For each $1 \leq j \leq p$, we introduce m fresh variables $w_1^j, w_2^j, \ldots, w_m^j$ into $\mathcal{I}''$. For each edge $\{w_i, w_k\} \in E$ and $1 \leq j \leq p$, introduce p copies of the constraint $EQ_j(w_i^j, w_k^j)$ into C''. Let $C_1, C_2, \ldots, C_q$ be an enumeration of the constraints in C'. Replace u_j by w_i^j in C_i for all $1 \leq i \leq q$. Finally, let C^* be the union of the (modified) constraints in C' and the equality constraints in C''. It is clear that each variable occurs in $\mathcal{I}''$ at most $l' = \max\{14p+1, l\}$ times (as G is 14-regular).

Clearly, a solution s to $\mathcal{I}$ satisfying all constraints can be extended to a solution to $\mathcal{I}''$, also satisfying all constraints, by setting $s(w_i^j) = j$ for all $1 \leq i \leq m$ and all $1 \leq j \leq p$.

On the other hand, if $m(\mathcal{I}, s) \leq \varepsilon|C|$, then let s' be an optimal solution to $\mathcal{I}''$. We will prove that there is a constant $\varepsilon' < 1$ (which depends on ε but not on $\mathcal{I}$) such that $m(\mathcal{I}'', s') \leq \varepsilon'|C^*|$.

We first prove that, for each $1 \leq j \leq p$, we can assume that all variables in $W^j = \{w_1^j, w_2^j, \ldots, w_m^j\}$ have been assigned the same value by s' and that all constraints in C'' are satisfied by s'. We show that given a solution s' to $\mathcal{I}''$, we can construct another solution s_2 such that $m(\mathcal{I}'', s_2) \geq m(\mathcal{I}'', s')$ and s_2 satisfies all constraints in C''.

Let a^j be the value that at least m/p of the variables in W^j have been assigned by s'. We construct the solution s_2 as follows: $s_2(w_i^j) = a^j$ for all i and j, and $s_2(x) = s'(x)$ for all other variables.

If there is some j such that $X = \{x \in W^j \mid s'(x) \neq a^j\}$ is non-empty, then, since G is an expander graph, there are at least $p \cdot \min(|X|, |W^j \setminus X|)$ constraints in C'' which are not satisfied by s'. Note that by the choice of X, we have $|W^j \setminus X| \geq m/p$ which implies $p \cdot \min(|X|, |W^j \setminus X|) \geq |X|$. By changing the value of the variables in X, we will make at most $|X|$ non-equality constraints in C^* unsatisfied because each of the variables in W^j occurs in at most one non-equality constraint in C^*. In other words, when the value of the variables in X are changed we gain at least $|X|$ in the measure as some of the equality constraints in C'' will become satisfied, furthermore we lose at most $|X|$ by making at most $|X|$ constraints in C^* unsatisfied. We conclude that $m(\mathcal{I}', s_2) \geq m(\mathcal{I}', s')$. Thus, we may assume that all equality constraints in C'' are satisfied by s'.

Since the expander graph G is 14-regular and has at most αq vertices, it has at most $\frac{14}{2}\alpha q$ edges. Hence, the number of equality constraints in C'' is at most $7\alpha qp$, and $|C''|/|C'| \leq 7\alpha p$. We can now bound $m(\mathcal{I}'', s_2)$ as follows:

$$m(\mathcal{I}'', s_2) \leq \text{OPT}(\mathcal{I}') + |C''| \leq \frac{\varepsilon|C'| + |C''|}{|C'| + |C''|}(|C'| + |C''|) \leq \frac{\varepsilon + 7\alpha p}{1 + 7\alpha p}(|C'| + |C''|).$$

Since $|C^*| = |C'| + |C''|$, it remains to set $\varepsilon' = \frac{\varepsilon + 7\alpha p}{1 + 7\alpha p}$. $\square$

We finish this section by using Theorem 22 to answer, at least partially, two open questions. The first one concerns the complexity of CSP(Γ)-B. In particular, the following conjecture has been made by Feder et al. [28].

Conjecture: For any fixed Γ such that CSP(Γ) is **NP**-complete there is an integer k such that CSP(Γ)-k is **NP**-complete.

Under the assumption that the algebraic CSP conjecture (that all problems CSP(Γ) not covered by Theorem 21 are tractable) holds, an affirmative answer follows immediately from Theorem 22. So for all constraint languages Γ such that CSP(Γ) is currently known to be **NP**-complete it is also the case that CSP(Γ)-B is **NP**-complete.

The second result concerns the approximability of equations over non-abelian groups. Petrank [50] has noted that hardness at gap location 1 implies the following: suppose that we restrict ourselves to instances of Max CSP(Γ) such that there exist solutions that satisfy all constraints, i.e. we concentrate on *satisfiable* instances. Then, there exists a constant c (depending on Γ) such that no polynomial-time algorithm can approximate this problem within c. We get the following result for satisfiable instances:

Corollary 29 *Let Γ be a core constraint language and let $\mathcal{A}$ be the algebra associated with Γ. Assume there is a factor $\mathcal{B}$ of $\mathcal{A}^c$ such that $\mathcal{B}$ only have projections as term operations. Then, there exists a constant c such that* Max CSP(Γ)-B *restricted to satisfiable instances cannot be approximated within c in polynomial time.*

We will now use this observation for studying a problem concerning groups. Let $\mathcal{G} = (G, \cdot)$ denote a finite group with identity element 1_G. An *equation* over a set of variables V is an expression of the form $w_1 \cdot \ldots \cdot w_k = 1_G$, where w_i (for $1 \leq i \leq k$) is either a variable, an inverted variable, or a group constant. Engebretsen et al. [27] have studied the following problem:

Definition 30 (Eq$_{\mathcal{G}}$) *The computational problem* Eq$_{\mathcal{G}}$ *(where $\mathcal{G}$ is a finite group) is defined to be the optimisation problem with*

Instance: *A set of variables V and a collection of equations E over V.*
Solution: *An assignment $s : V \to G$ to the variables.*
Measure: *Number of equations in E which are satisfied by s.*

The problem Eq$^1_{\mathcal{G}}[3]$ is the same as Eq$_{\mathcal{G}}$ except for the additional restrictions that each equation contains exactly three variables and no equation contains the same variable more than once. Their main result was the following inapproximability result:

Theorem 31 (Theorem 1 in [27]) *For any finite group $\mathcal{G}$ and constant $\varepsilon >$*

0, *it is* **NP**-*hard to approximate* $\textsc{Eq}^1_{\mathcal{G}}[3]$ *within* $|G| - \varepsilon$.

Engebretsen et al. left the approximability of $\textsc{Eq}^1_{\mathcal{G}}[3]$ for satisfiable instances as an open question. We will give a partial answer to the approximability of satisfiable instances of $\textsc{Eq}_{\mathcal{G}}$.

It is not hard to see that for any integer k, the equations with at most k variables over a finite group can be viewed as a constraint language. For a group $\mathcal{G}$, we denote the constraint language which corresponds to equations with at most three variables by $\Gamma_{\mathcal{G}}$. Hence, for any finite group $\mathcal{G}$, the problem $\textsc{Max CSP}(\Gamma_{\mathcal{G}})$ is no harder than $\textsc{Eq}_{\mathcal{G}}$.

Goldmann and Russell [30] have shown that $\mathrm{CSP}(\Gamma_{\mathcal{G}})$ is **NP**-hard for every finite non-abelian group $\mathcal{G}$. This result was extended to more general algebras by Larose and Zádori [45]. They also showed that for any non-abelian group $\mathcal{G}$, the algebra $\mathcal{A} = (G; \mathrm{Pol}(\Gamma_{\mathcal{G}}))$ has a non-trivial factor $\mathcal{B}$ such that $\mathcal{B}$ only has projections as term operations. We now combine Larose and Zádori's result with Theorem 22:

Corollary 32 *For any finite non-abelian group* $\mathcal{G}$, $\textsc{Eq}_{\mathcal{G}}$ *has a hard gap at location 1.*

Thus, there is a constant c such that no polynomial-time algorithm can approximate satisfiable instances of $\textsc{Eq}_{\mathcal{G}}$ better than c. There also exists a constant k (depending on the group $\mathcal{G}$) such that the result holds for instances with variable occurrence bounded by k.

4 Approximability of Single Relation Max CSP

In this section, we will prove the following theorem:

Theorem 33 *Let* $R \in R_D^{(n)}$ *be non-empty. If* $(d, \ldots, d) \in R$ *for some* $d \in D$, *then* $\textsc{Max CSP}(\{R\})$ *is solvable in linear time. Otherwise,* $\textsc{Max CSP}(\{R\})$-B *is hard to approximate.*

Proof. The tractability part of the theorem is trivial. It was shown in [36] that any non-empty non-valid relation of arity $n \geq 2$ strictly implements a binary non-empty non-valid relation. Hence, by Lemma 12, it is sufficient to to prove the the hardness part for binary relations. It is often convenient to view binary relations as digraphs. The proof for vertex-transitive digraphs is presented in §4.1, and for the remaining digraphs in §4.2. □

Recall that a *digraph* is a pair (V, E) such that V is a finite set and $E \subseteq V \times V$. A *graph* is a digraph (V, E) such that for every pair $(x, y) \in E$ we also have $(y, x) \in E$. Let $R \in R_D$ be a binary relation. As R is binary it can be viewed as a digraph G with vertex set $V[G] = D$ and edge set $E[G] = R$. We will mix freely between those two notations. For example, we will sometimes write $(x, y) \in G$ with the intended meaning $(x, y) \in E[G]$.

Let G be a digraph, $R = E[G]$, and let $\mathrm{Aut}(G)$ denote the automorphism group of G. If $\mathrm{Aut}(G)$ is transitive (i.e., contains a single orbit), then we say that G is *vertex-transitive*. If D can be partitioned into two sets, A and B, such that for any $x, y \in A$ (or $x, y \in B$) we have $(x, y) \notin R$, then R (and G) is *bipartite*. The *directed cycle of length* n is the digraph G with vertex set $V[G] = \{0, 1, \ldots, n-1\}$ and edge set $E[G] = \{(x, x+1) \mid x \in V[G]\}$, where the addition is modulo n. Analogously, the *undirected cycle of length* n is the graph H with vertex set $V[H] = \{0, 1, \ldots, n-1\}$ and edge set $E[H] = \{(x, x+1) \mid x \in V[H]\} \cup \{(x+1, x) \mid x \in V[H]\}$ (also in this case the additions are modulo n). The undirected path with two vertices will be denoted by P_2.

4.1 Vertex-transitive Digraphs

We will now tackle non-bipartite vertex-transitive digraphs and prove that they give rise to Max CSP problems which are hard at gap location 1. To do this, we make use of the algebraic framework which we used and developed in §3. We will also use a theorem by Barto, Kozik, and Niven [7] on the complexity of $\mathrm{CSP}(G)$ for digraphs G without sources and sinks. A vertex v in a digraph is a *source* if there is no incoming edge to v. Similarly, a vertex v is a *sink* if there is no outgoing edge from v.

Theorem 34 ([7]) *If G is a core digraph without sources and sinks which does not retract to a disjoint union of directed cycles, then G admits no wnuf.*

From this result we derive the following corollary.

Corollary 35 *Let H be a vertex-transitive core digraph which is non-empty, non-valid, and not a directed cycle. Then,* Max CSP$(\{H\})$*-B has a hard gap at location 1.*

Proof. Let v and u be two vertices in H. As H is vertex-transitive the in- and out-degrees of u and v must coincide, and hence the in- and out-degrees of v must be the same. Hence, H does not have any sources or sinks. Furthermore, as H is vertex-transitive and a core it follows that it is connected. The result now follows from Theorem 34, Theorem 24, and Theorem 22. □

The next lemmas help to deal with the remaining vertex-transitive graphs, i.e. those that retract to a directed cycle.

Lemma 36 *If G is the undirected path with two vertices P_2, or an undirected cycle C_k, $k > 2$, then* MAX CSP($\{G\}$)-B *is hard to approximate.*

Proof. If $G = P_2$, then the result follows from Example 3. If $G = C_k$ and k is even, then the core of C_k is isomorphic to P_2 and the result follows from Lemmas 15, 9 combined with Example 3.

From now on, assume that $G = C_k$, k is odd, and $k \geq 3$. We will show that we can strictly implement N_k, i.e., the inequality relation. We use the following strict implementation

$$N_k(z_1, z_{k-1}) + (k-3) = \max_{z_2, z_3, \ldots, z_{k-2}} C_k(z_1, z_2) + C_k(z_2, z_3) + \ldots + C_k(z_{k-3}, z_{k-2}) + C_k(z_{k-2}, z_{k-1}).$$

It is not hard to see that if $z_1 \neq z_{k-1}$, then all $k-2$ constraints on the right hand side can be satisfied. If $z_1 = z_{k-1}$, then $k-3$ constraints are satisfied by the assignment $z_i = z_1 + i - 1$, for all i such that $1 < i < k-1$ (the addition and subtraction are modulo k). Furthermore, no assignment can satisfy all constraints. To see this, note that such an assignment would define a path $z_1, z_2, \ldots, z_{k-1}$ in C_k with $k-2$ edges and $z_1 = z_{k-1}$. This is impossible since $k-2$ is odd and $k-2 < k$.

The lemma now follows from Lemmas 12 and 9 together with Example 3. □

Lemma 37 *If G is a digraph such that $(x, y) \in E[G] \Rightarrow (y, x) \notin E[G]$, then* MAX CSP($\{H\}$)-$B \leq_{AP}$ MAX CSP($\{G\}$)-B, *where H is the undirected graph obtained from G by replacing every edge in G by two edges in opposing directions in H.*

Proof. $H(x, y) + (1-1) = G(x, y) + G(y, x)$ is a strict implementation of H and the result follows from Lemma 12. □

Lemma 38 *If G is a non-empty non-valid vertex-transitive digraph, then* MAX CSP($\{G\}$)-B *is hard to approximate.*

Proof. By Lemmas 15 and 9, it is enough to consider cores. For directed cycles, the result follows from Lemmas 36 and 37, and, for all other digraphs, from Corollary 35. □

We now deal with digraphs that are not vertex-transitive.

Lemma 39 *If G is a bipartite digraph which is neither empty nor valid, then* MAX CSP($\{G\}$)-B *is hard to approximate.*

Proof. If there are two edges $(x, y), (y, x) \in E[G]$, then the core of G is isomorphic to P_2 and the result follows from Lemmas 9 and 15 together with Example 3. If no such pair of edges exist, then Lemmas 9 and 37 reduce this case to the previous case where there are two edges $(x, y), (y, x) \in E[G]$. $\square$

We will use a technique known as *domain restriction* [24] in the sequel. For a subset $D' \subseteq D$, let $\Gamma\big|_{D'} = \{R\big|_{D'} \mid R \in \Gamma \text{ and } R\big|_{D'} \text{ is non-empty}\}$. The following lemma was proved in [24, Lemma 3.5] (the lemma is stated in a slightly different form there, but the proof together with [6, Lemma 8.2] and Lemma 5 implies the existence of the required AP-reduction).

Lemma 40 *If $D' \subseteq D$ and $D' \in \Gamma$, then* MAX CSP($\Gamma\big|_{D'}$)-$B \leq_{AP}$ MAX CSP(Γ)-B.

Typically, we will let D' be an orbit in the automorphism group of a graph. We are now ready to present the three lemmas that are the building blocks of the main lemma in this section, Lemma 44. Let G be a digraph. For a set $A \subseteq V[G]$, we define $A^+ = \{j \mid (i, j) \in E[G], i \in A\}$, and $A^- = \{i \mid (i, j) \in E[G], j \in A\}$.

Lemma 41 *If a constraint language Γ contains two unary predicates S, T such that $S \cap T = \varnothing$, then Γ strictly implements $S \cup T$.*

Proof. Let $U = S \cup T$. Then $U(x) + (1 - 1) = S(x) + T(x)$ is a strict implementation of $U(x)$. $\square$

Lemma 42 *Let H be a core digraph and Ω an orbit in* $\mathrm{Aut}(H)$. *Then, H strictly implements Ω^+ and Ω^-.*

Proof. Assume that $H \in R_D$ where $D = \{1, 2, \ldots, p\}$ and (without loss of generality) assume that $1 \in \Omega$. We construct a strict implementation of Ω^+; the other case can be proved in a similar way. Consider the function

$$g(z_1, \ldots, z_p) = \sum_{H(i,j)=1} H(z_i, z_j).$$

Since H is a core, it follows that $g(a_1, \dots, a_p) = |E[H]|$ if and only if the function mapping i to a_i, $i = 1, \dots, p$, is an automorphism of H. This also implies that a necessary condition for $g(a_1, \dots, a_p) = |E[H]|$ is that a_1 is assigned some element in the orbit containing 1, i.e. the orbit Ω. We claim that Ω^+ can be strictly implemented as follows:

$$\Omega^+(x) + (\alpha - 1) = \max_{\boldsymbol{z}} (H(z_1, x) + g(\boldsymbol{z}))$$

where $\boldsymbol{z} = (z_1, z_2, \dots, z_p)$ and $\alpha = |E[H]| + 1$.

Assume first that $x \in \Omega^+$ and choose $y \in \Omega$ such that $H(y, x) = 1$. Then, there exists an automorphism σ such that $\sigma(1) = y$ and $H(z_1, x) + g(\boldsymbol{z}) = 1 + |E[H]|$ by assigning variable z_i, $1 \le i \le p$, the value $\sigma(i)$.

If $x \notin \Omega^+$, then there is no $y \in \Omega$ such that $H(y, x) = 1$. If the constraint $H(z_1, x)$ is to be satisfied, then z_1 must be chosen such that $z_1 \notin \Omega$. We have already observed that such an assignment cannot be extended to an automorphism of H and, consequently, $H(z_1, x) + g(\boldsymbol{z}) < 1 + |E[H]|$ whenever $z_1 \notin \Omega$. However, the assignment $z_i = i$, $1 \le i \le p$, makes $H(z_1, x) + g(\boldsymbol{z}) = |E[H]|$ since the identity function is an automorphism of H. □

Lemma 43 *If H is a core digraph and Ω an orbit in* $\mathrm{Aut}(H)$*, then, for every k, there is a k' such that* MAX CSP($\{H|_\Omega\}$)-k $\le_{AP}$ MAX CSP($\{H\}$)-k'.

Proof. Let $V[H] = \{1, 2, \dots, p\}$ and arbitrarily choose one element $d \in \Omega$. Let $\mathcal{I} = (V, C)$ be an arbitrary instance of MAX CSP($\{H|_\Omega\}$)-k and let $V = \{v_1, \dots, v_n\}$. Let $k' = k|E[H]| + k$. We construct an instance $\mathcal{I}' = (V' \cup V, C' \cup C)$ of MAX CSP($\{H\}$)-k' as follows: for each variable $v_i \in V$:

(1) Add fresh variables $w_i^1, \dots, w_i^{d-1}, w_i^{d+1}, \dots, w_i^p$ to V' and let w_i^d denote the variable v_i.
(2) For each $(a, b) \in E[H]$, add k copies of the constraint $H(w_i^a, w_i^b)$ to C'.

It is clear that $\mathcal{I}'$ is an instance of MAX CSP($\{H\}$)-k'. (If some vertex $i \in V[H]$ occur in every edge in H, then w_i^d occur at most $k|E[H]| + k$ times in $\mathcal{I}'$. This is the worst case given by the construction above.)

Let s' be a solution to $\mathcal{I}'$. For an arbitrary variable $v_i \in V$, if there is some constraint in C' which is not satisfied by s', then we can get another solution s'' by modifying s' so that every constraint in C' is satisfied (if $H(w_i^a, w_i^b)$ is a constraint which is not satisfied by s' then set $s''(w_i^a) = a$ and $s''(w_i^b) = b$). We will denote this polynomial-time algorithm by P', so $s'' = P'(s')$. The corresponding solution to $\mathcal{I}$ will be denoted by $P(s')$, so $P(s')(v_i) = P'(s')(w_i^d)$.

The algorithm P may make some of the constraints involving v_i unsatisfied (at most k constraints will be made unsatisfied as v_i occurs in at most k constraints in $\mathcal{I}$). However, the number of copies, k, of the constraints in C' implies that $m(\mathcal{I}', s') \leq m(\mathcal{I}', P'(s'))$. In particular, this means that any optimal solution to $\mathcal{I}'$ can be used to construct another optimal solution which satisfies all constraints in C'.

Hence, for each $v_i \in V$, all constraints from step 2 are satisfied by $s'' = P'(s')$. As H is a core, s'' restricted to $w_i^1, \ldots, w_i^p$ (for any $v_i \in V$) induces an automorphism of H. Denote the automorphism by $f : V[H] \to V[H]$ and note that f can be defined as $f(x) = s''(w_i^x)$. Furthermore, $s''(w_i^d) \in \Omega$ for all $w_i^d \in V$ since $d \in \Omega$.

To simplify the notation we let $l = |E[H]|$. By a straightforward probabilistic argument we have $\textsc{opt}(\mathcal{I}) \geq \frac{l}{p^2}|C|$. Using this fact and the argument above we can bound the optimum of $\mathcal{I}'$ as follows:

$$\begin{aligned}\textsc{opt}(\mathcal{I}') &\leq \textsc{opt}(\mathcal{I}) + kl|V| \\ &\leq \textsc{opt}(\mathcal{I}) + k^2 l|C| \\ &\leq \textsc{opt}(\mathcal{I}) + k^2p^2\textsc{opt}(\mathcal{I}) \\ &= (1 + k^2p^2)\textsc{opt}(\mathcal{I}).\end{aligned}$$

From Lemma 5 we know that there exists a polynomial-time approximation algorithm A for $\textsc{Max}$ CSP$(H\big|_{\Omega})$. Let us assume that A is a c-approximation algorithm, i.e., it produces solutions which are c-approximate in polynomial time. We construct the algorithm G in the AP-reduction as follows:

$$G(\mathcal{I}, s') = \begin{cases} P(s') \text{ if } m(\mathcal{I}, P(s')) \geq m(\mathcal{I}, A(\mathcal{I})), \\ A(\mathcal{I}) \text{ otherwise.} \end{cases}$$

We see that $\textsc{opt}(\mathcal{I})/m(\mathcal{I}, G(\mathcal{I}, s')) \leq c$. Let s' be an r-approximate solution to $\mathcal{I}'$. As $m(\mathcal{I}', s') \leq m(\mathcal{I}', P'(s'))$, we get that $P'(s')$ is an r-approximate solution to $\mathcal{I}'$, too. Furthermore, since $P'(s')$ satisfies all constraints introduced in step 2, we have $\textsc{opt}(\mathcal{I}') - m(\mathcal{I}', P'(s')) = \textsc{opt}(\mathcal{I}) - m(\mathcal{I}, P(s'))$. Let $\beta =$

$1 + k^2p^2$ and note that

$$\begin{aligned}
\frac{\text{OPT}(\mathcal{I})}{m(\mathcal{I}, G(\mathcal{I}, s'))} &= \frac{m(\mathcal{I}, P(s'))}{m(\mathcal{I}, G(\mathcal{I}, s'))} + \frac{\text{OPT}(\mathcal{I}') - m(\mathcal{I}', P'(s'))}{m(\mathcal{I}, G(\mathcal{I}, s'))} \\
&\leq 1 + \frac{\text{OPT}(\mathcal{I}') - m(\mathcal{I}', P'(s'))}{m(\mathcal{I}, G(\mathcal{I}, s'))} \\
&\leq 1 + c \cdot \frac{\text{OPT}(\mathcal{I}') - m(\mathcal{I}', P'(s'))}{\text{OPT}(\mathcal{I})} \\
&\leq 1 + c\beta \cdot \frac{\text{OPT}(\mathcal{I}') - m(\mathcal{I}', P'(s'))}{\text{OPT}(\mathcal{I}')} \\
&\leq 1 + c\beta \cdot \frac{\text{OPT}(\mathcal{I}') - m(\mathcal{I}', P'(s'))}{m(\mathcal{I}', P'(s'))} \leq 1 + c\beta(r-1).
\end{aligned}$$

□

Lemma 44 *Let H be a non-empty non-valid digraph with at least two vertices which is not vertex-transitive. Then* MAX CSP($\{H\}$)-B *is hard to approximate.*

Proof. The proof is by induction on the number of vertices, $|V[H]|$. If $|V[H]| = 2$ then the result follows from Lemma 39. Assume now that $|V[H]| > 2$ and the lemma holds for all digraphs with a smaller number of vertices. Note that if H is not a core then the core of H has fewer vertices or is vertex-transitive. In either case, the result follows. So assume that H is a core.

We claim that either (a) MAX CSP($\{H\}$)-B is hard to approximate, or (b) there exists a proper subset X of V such that $|X| \geq 2$, $H\big|_X$ is non-empty, $H\big|_X$ is non-valid and for every k there exists a k' such that MAX CSP($\{H\big|_X\}$)-k $\leq_{AP}$ MAX CSP($\{H\}$)-k'. Since the core of $H\big|_X$ either is vertex-transitive or has fewer vertices than H, the lemma will follow from this claim.

We now split the proof of the claim into three cases.

Case 1: There exists an orbit $\Omega_1 \subsetneq V[H]$ such that Ω_1^+ contains at least one orbit.

If $H\big|_{\Omega_1}$ is non-empty, then we get the result from Lemma 43 and the induction hypothesis, since $\Omega_1 \subsetneq V[H]$ (we cannot have $|\Omega_1| = 1$ because then H would contain a loop). Assume that $H\big|_{\Omega_1}$ is empty. As $H\big|_{\Omega_1}$ is empty, we get that Ω_1^+ is a proper subset of $V[H]$ with at least two elements. If $H\big|_{\Omega_1^+}$ is non-empty, then we get the result from Lemmas 42, 12 and 40. Hence, we assume that $H\big|_{\Omega_1^+}$ is empty.

Arbitrarily choose an orbit $\Omega_2 \subseteq \Omega_1^+$ and note that $\Omega_1^+ \cap \Omega_2^- = \varnothing$ since $H\big|_{\Omega_1^+}$ is empty. If $\Omega_1^+ \cup \Omega_2^- \subsetneq V[H]$, then we get the result from Lemmas 42, 12, 41 and 40 because $H\big|_{\Omega_1^+ \cup \Omega_2^-}$ is non-empty. Hence, we can assume without loss of generality that $\Omega_1^+ \cup \Omega_2^- = V[H]$, and since $\Omega_1^+ \cap \Omega_2^- = \varnothing$, we have an partition of $V[H]$ into the sets Ω_1^+ and Ω_2^-. Using the same argument as for Ω_1^+, we can assume that $H\big|_{\Omega_2^-}$ is empty. Therefore, Ω_1^+,Ω_2^- is a partition of $V[H]$ and $H\big|_{\Omega_1^+}$,$H\big|_{\Omega_2^-}$ are both empty. This implies that H is bipartite and we get the result from Lemma 39.

Case 2: There exists an orbit $\Omega_1 \subset V[H]$ such that Ω_1^- contains at least one orbit.

This case is analogous to the previous case.

Case 3: For every orbit $\Omega \subseteq V[H]$, neither Ω^+ nor Ω^- contains any orbits.

Pick any two orbits Ω_1 and Ω_2 (not necessarily distinct). Assume that there are $x \in \Omega_1$ and $y \in \Omega_2$ such that $(x, y) \in E[H]$. Let z be an arbitrary vertex in Ω_2. Since Ω_2 is an orbit of H, there is an automorphism $\rho \in \mathrm{Aut}(H)$ such that $\rho(y) = z$, so $(\rho(x), z) \in E[H]$. Furthermore, Ω_1 is an orbit of $\mathrm{Aut}(H)$ so $\rho(x) \in \Omega_1$. Since z was chosen arbitrarily, we conclude that $\Omega_2 \subseteq \Omega_1^+$. However, this contradicts our assumption that neither Ω_1^+ nor Ω_1^- contains any orbit. We conclude that for any pair Ω_1, Ω_2 of orbits and any $x \in \Omega_1$, $y \in \Omega_2$, we have $(x, y) \notin E[G]$. This implies that H is empty and Case 3 cannot occur. $\square$

We will now give a simple example on how Theorem 33 can be used for studying the approximability of constraint languages.

Corollary 45 *Let* Γ *be a constraint language such that* $\mathrm{Aut}(\Gamma)$ *contains a single orbit. If* Γ *contains a non-empty* k*-ary,* $k > 1$*, relation* R *which is not* d*-valid for all* $d \in D$*, then* Max CSP(Γ)-B *is hard to approximate. Otherwise,* Max CSP(Γ) *is tractable.*

Proof. If a relation R with the properties described above exists, then Max CSP(Γ)-B is hard to approximate by Theorem 33 (note that R cannot be d-valid for any d). Otherwise, every k-ary, $k > 1$, relation $S \in \Gamma$ is d-valid for all $d \in D$. If Γ contains a unary relation U such that $U \subsetneq D$, then $\mathrm{Aut}(\Gamma)$ would contain at least two orbits which contradict our assumptions. It follows that Max CSP(Γ) is trivially solvable. $\square$

Note that the constraint languages considered in Corollary 45 may be seen as a generalisation of vertex-transitive graphs.

4.3 Max CSP *and Supermodularity*

In this section, we will prove two results whose proofs make use of Theorem 33. The first result (Proposition 51) concerns the hardness of approximating Max CSP(Γ) for Γ which contains all at most binary relations which are 2-monotone (see §4.3.1 for a definition) on some partially ordered set which is not a lattice order. The other result, Theorem 53, states that Max CSP(Γ) is hard to approximate if Γ contains all at most binary supermodular predicates on some lattice and in addition contains at least one predicate which is not supermodular on the lattice.

These results strengthens earlier published results [42,43] in various ways (e.g., they apply to a larger class of constraint languages or they give approximation hardness instead of **NP**-hardness). In §4.3.1 we give a few preliminaries which are needed in this section while the new results are contained in §4.3.2.

4.3.1 Preliminaries

Recall that a partial order $\sqsubseteq$ on a domain D is a *lattice order* if, for every $x, y \in D$, there exist a greatest lower bound $x \sqcap y$ and a least upper bound $x \sqcup y$. The algebra $\mathcal{L} = (D; \sqcap, \sqcup)$ is a *lattice*, and $x \sqcup y = y \iff x \sqcap y = x \iff x \sqsubseteq y$. We will write $x \sqsubset y$ if $x \neq y$ and $x \sqsubseteq y$. All lattices we consider will be finite, and we will simply refer to these algebras as *lattices* instead of using the more appropriate term *finite lattices*. The *direct power* of $\mathcal{L}$, denoted by $\mathcal{L}^n$, is the lattice with domain D^n and operations acting componentwise.

Definition 46 (Supermodular function) *Let $\mathcal{L}$ be a lattice. A function $f : \mathcal{L}^n \to \mathbb{R}$ is called* supermodular *on $\mathcal{L}$ if it satisfies,*

$$f(\boldsymbol{a}) + f(\boldsymbol{b}) \leq f(\boldsymbol{a} \sqcap \boldsymbol{b}) + f(\boldsymbol{a} \sqcup \boldsymbol{b}) \tag{1}$$

for all $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{L}^n$.

The set of all supermodular predicates on a lattice $\mathcal{L}$ will be denoted by $\mathrm{Spmod}_{\mathcal{L}}$ and a constraint language Γ is said to be supermodular on a lattice $\mathcal{L}$ if $\Gamma \subseteq \mathrm{Spmod}_{\mathcal{L}}$. We will sometimes use an alternative way of characterising supermodularity:

Theorem 47 ([25]) *An n-ary function f is supermodular on a lattice $\mathcal{L}$ if and only if it satisfies inequality* (1) *for all* $(a_1, a_2, \ldots, a_n)$, $(b_1, b_2, \ldots, b_n) \in \mathcal{L}^n$ *such that*

(1) $a_i = b_i$ *with one exception, or*
(2) $a_i = b_i$ *with two exceptions, and, for each* i*, the elements* a_i *and* b_i *are comparable in* $\mathcal{L}$*.*

The following definition first occurred in [17].

Definition 48 (Generalised 2-monotone) *Given a poset* $\mathcal{P} = (D, \sqsubseteq)$*, a predicate* f *is said to be* generalised 2-monotone on $\mathcal{P}$ *if*

$$f(\boldsymbol{x}) = 1 \iff ((x_{i_1} \sqsubseteq a_{i_1}) \wedge \ldots \wedge (x_{i_s} \sqsubseteq a_{i_s})) \vee ((x_{j_1} \sqsupseteq b_{j_1}) \wedge \ldots \wedge (x_{j_s} \sqsupseteq b_{j_s}))$$

where $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ *and* $a_{i_1}, \ldots, a_{i_s}, b_{j_1}, \ldots, b_{j_s} \in D$*, and either of the two disjuncts may be empty.*

It is not hard to verify that generalised 2-monotone predicates on some lattice are supermodular on the same lattice. For brevity, we will use the word *2-monotone* instead of generalised 2-monotone.

The following theorem follows from [24, Remark 4.7]. The proof in [24] uses the corresponding unbounded occurrence case as an essential stepping stone; see [21] for a proof of this latter result.

Theorem 49 (Max CSP **on a Boolean domain)** *Let* $D = \{0, 1\}$ *and* $\Gamma \subseteq R_D$ *be a core. If* Γ *is not supermodular on any lattice on* D*, then* Max CSP(Γ)-B *is hard to approximate. Otherwise,* Max CSP(Γ) *is tractable.*

4.3.2 Results

The following proposition is a combination of results proved in [17] and [42].

Proposition 50

- *If* Γ *consists of 2-monotone relations on a lattice, then* Max CSP(Γ) *can be solved in polynomial time.*
- *Let* $\mathcal{P} = (D, \sqsubseteq)$ *be a poset which is not a lattice. If* Γ *contains all at most binary 2-monotone relations on* $\mathcal{P}$*, then* Max CSP(Γ) *is* **NP***-hard.*

We strengthen the second part of the above result as follows:

Proposition 51 *Let* $\sqsubseteq$ *be a partial order, which is not a lattice order, on* D*. If* Γ *contains all at most binary 2-monotone relations on* $\sqsubseteq$*, then* Max CSP(Γ)-B *is hard to approximate.*

Proof. Since $\sqsubseteq$ is a non-lattice partial order, there exist two elements $a, b \in D$ such that either $a \sqcap b$ or $a \sqcup b$ do not exist. We will give a proof for the first case and the other case can be handled analogously.

Let $g(x, y) = 1 \iff (x \sqsubseteq a) \wedge (y \sqsubseteq b)$. The predicate g is 2-monotone on $\mathcal{P}$ so $g \in \Gamma$. We have two cases to consider: (a) a and b have no common lower bound, and (b) a and b have at least two maximal common lower bounds. In the first case g is not valid. To see this, note that if there is an element $c \in D$ such that $g(c, c) = 1$, then $c \sqsubseteq a$ and $c \sqsubseteq b$, and this means that c is a common lower bound for a and b, a contradiction. Hence, g is not valid, and the proposition follows from Theorem 33.

In case (b) we will use the domain restriction technique from Lemma 40 together with Theorem 33. In case (b), there exist two distinct elements $c, d \in D$, such that $c, d \sqsubseteq a$ and $c, d \sqsubseteq b$. Furthermore, we can assume that there is no element $z \in D$ distinct from a, b, c such that $c \sqsubseteq z \sqsubseteq a, b$, and, similarly, we can assume there is no element $z' \in D$ distinct from a, b, d such that $d \sqsubseteq z' \sqsubseteq a, b$.

Let $f(x) = 1 \iff (x \sqsupseteq c) \wedge (x \sqsupseteq d)$. This predicate is 2-monotone on $\mathcal{P}$. Note that there is no element $z \in D$ such that $f(z) = 1$ and $g(z, z) = 1$, but we have $f(a) = f(b) = g(a, b) = 1$. By restricting the domain to $D' = \{x \in D \mid f(x) = 1\}$ with Lemma 40, the result follows from Theorem 33. □

A *diamond* is a lattice $\mathcal{L}$ on a domain D such that $|D|-2$ elements are pairwise incomparable. That is, a diamond on $|D|$ elements consist of a top element, a bottom element and $|D| - 2$ elements which are pairwise incomparable. The following result was proved in [43].

Theorem 52 *Let* Γ *contain all at most binary 2-monotone predicates on some diamond* $\mathcal{L}$. *If* $\Gamma \not\subseteq \mathrm{Spmod}_{\mathcal{L}}$, *then* Max CSP($\Gamma$) *is* **NP**-*hard.*

By modifying the original proof of Theorem 52, we can strengthen the result in three ways: our result applies to arbitrary lattices, we prove inapproximability results instead of **NP**-hardness, and we prove the result for bounded occurrence instances.

Theorem 53 *Let* Γ *contain all at most binary 2-monotone predicates on an arbitrary lattice* $\mathcal{L}$. *If* $\Gamma \not\subseteq \mathrm{Spmod}_{\mathcal{L}}$, *then* Max CSP($\Gamma$)-$B$ *is hard to approximate.*

Proof. Let $f \in \Gamma$ be a predicate such that $f \notin \mathrm{Spmod}_{\mathcal{L}}$. We will first prove that f can be assumed to be at most binary. By Theorem 47, there is a unary or binary predicate $f' \notin \mathrm{Spmod}_{\mathcal{L}}$ which can be obtained from f by substituting all but at most two variables by constants. We present the initial part of the proof with the assumption that f' is binary and the case when f' is unary can be dealt with in the same way. Denote the constants by $a_3, a_4, \ldots, a_n$ and assume that $f'(x, y) = f(x, y, a_3, a_4, \ldots, a_n)$.

Let $k \geq 5$ be an integer and assume that Max CSP$(\Gamma \cup \{f'\})$-k is hard to approximate. We will prove that Max CSP(Γ)-k is hard to approximate by exhibiting an AP-reduction from Max CSP$(\Gamma \cup \{f'\})$-k to Max CSP(Γ)-k. Given an instance $\mathcal{I} = (V, C)$ of Max CSP$(\Gamma \cup \{f'\})$-k, where $C = \{C_1, C_2, \ldots, C_q\}$, we construct an instance $\mathcal{I}' = (V', C')$ of Max CSP(Γ)-k as follows:

(1) for any constraint $(f', \boldsymbol{v}) = C_j \in C$, introduce the constraint $(f, \boldsymbol{v'})$ into C, where $\boldsymbol{v'} = (v_1, v_2, y_3^j, \ldots, y_n^j)$, and add the fresh variables $y_3^j, y_4^j, \ldots, y_n^j$ to V'. Add two copies of the constraints $y_i^j \sqsubseteq a_i$ and $a_i \sqsubseteq y_i^j$ for each $i \in \{3, 4, \ldots, n\}$ to C'.
(2) for other constraints, i.e., $(g, \boldsymbol{v}) \in C$ where $g \neq f'$, add $(g, \boldsymbol{v})$ to C'.

It is clear that $\mathcal{I}'$ is an instance of Max CSP(Γ)-k. If we are given a solution s' to $\mathcal{I}'$, we can construct a new solution s'' to $\mathcal{I}'$ by letting $s''(y_i^j) = a_i$ for all i, j and $s''(x) = s'(x)$, otherwise. Denote this transformation by P, so $s'' = P(s')$. It is not hard to see that $m(\mathcal{I}', P(s')) \geq m(\mathcal{I}', s')$.

From Lemma 5 we know that there is a constant c and polynomial-time c-approximation algorithm A for Max CSP$(\Gamma \cup \{f'\})$. We construct the algorithm G in the AP-reduction as follows:

$$G(\mathcal{I}, s') = \begin{cases} P(s')\big|_V & \text{if } m(\mathcal{I}, P(s')\big|_V) \geq m(\mathcal{I}, A(\mathcal{I})), \\ A(\mathcal{I}) & \text{otherwise.} \end{cases}$$

We see that $\text{OPT}(\mathcal{I})/m(\mathcal{I}, G(\mathcal{I}, s')) \leq c$.

By Lemma 5, there is a constant c' such that for any instance $\mathcal{I}$ of Max CSP(Γ), we have $\text{OPT}(\mathcal{I}) \geq c'|C|$. Furthermore, due to the construction of $\mathcal{I}'$ and the fact that $m(\mathcal{I}', P(s')) \geq m(\mathcal{I}', s')$, we have

$$\begin{aligned} \text{OPT}(\mathcal{I}') &\leq \text{OPT}(\mathcal{I}) + 4(n-2)|C| \\ &\leq \text{OPT}(\mathcal{I}) + \frac{4(n-2)}{c'} \cdot \text{OPT}(\mathcal{I}) \\ &\leq \text{OPT}(\mathcal{I}) \cdot \left(1 + \frac{4(n-2)}{c'}\right). \end{aligned}$$

Let s' be an r-approximate solution to $\mathcal{I}'$. As $m(\mathcal{I}', s') \leq m(\mathcal{I}', P(s'))$, we get that $P(s')$ also is an r-approximate solution to $\mathcal{I}'$. Furthermore, since $P(s')$ satisfies all constraints introduced in step 1, we have $\text{OPT}(\mathcal{I}') - m(\mathcal{I}', P(s')) =$

$\text{OPT}(\mathcal{I}) - m(\mathcal{I}, P(s')\big|_V)$. Let $\beta = 1 + 4(n-2)/c'$ and note that

$$\begin{aligned}
\frac{\text{OPT}(\mathcal{I})}{m(\mathcal{I}, G(\mathcal{I}, s'))} &= \frac{m(\mathcal{I}, P(s')\big|_V)}{m(\mathcal{I}, G(\mathcal{I}, s'))} + \frac{\text{OPT}(\mathcal{I}') - m(\mathcal{I}', P(s'))}{m(\mathcal{I}, G(\mathcal{I}, s'))} \\
&\leq 1 + \frac{\text{OPT}(\mathcal{I}') - m(\mathcal{I}', P(s'))}{m(\mathcal{I}, G(\mathcal{I}, s'))} \\
&\leq 1 + c \cdot \frac{\text{OPT}(\mathcal{I}') - m(\mathcal{I}', P(s'))}{\text{OPT}(\mathcal{I})} \\
&\leq 1 + c\beta \cdot \frac{\text{OPT}(\mathcal{I}') - m(\mathcal{I}', P(s'))}{\text{OPT}(\mathcal{I}')} \\
&\leq 1 + c\beta \cdot \frac{\text{OPT}(\mathcal{I}') - m(\mathcal{I}', P(s'))}{m(\mathcal{I}', P(s'))} \leq 1 + c\beta(r-1).
\end{aligned}$$

We conclude that MAX CSP(Γ)-k is hard to approximate if MAX CSP($\Gamma \cup \{f'\}$)-k is hard to approximate.

We will now prove that MAX CSP(Γ)-B is hard to approximate under the assumption that f is at most binary. We say that the pair $(\boldsymbol{a}, \boldsymbol{b})$ *witnesses the non-supermodularity of* f if $f(\boldsymbol{a}) + f(\boldsymbol{b}) \not\leq f(\boldsymbol{a} \sqcap \boldsymbol{b}) + f(\boldsymbol{a} \sqcup \boldsymbol{b})$.

Case 1: f is unary. As f is not supermodular on $\mathcal{L}$, there exists elements $a, b \in \mathcal{L}$ such that (a, b) witnesses the non-supermodularity of f.

Note that a and b cannot be comparable because we would have $\{a \sqcup b, a \sqcap b\} = \{a, b\}$, and so $f(a \sqcup b) + f(a \sqcap b) = f(a) + f(b)$ contradicting the choice of (a, b). We can now assume, without loss of generality, that $f(a) = 1$. Let $z_* = a \sqcap b$ and $z^* = a \sqcup b$. Note that the two predicates $u(x) = 1 \iff x \sqsubseteq z^*$ and $u'(x) = 1 \iff z_* \sqsubseteq x$ are 2-monotone and, hence, contained in Γ. By using Lemma 40, it is therefore enough to prove approximation hardness for MAX CSP($\Gamma\big|_{D'}$)-B, where $D' = \{x \in D \mid z_* \sqsubseteq x \sqsubseteq z^*\}$.

Subcase 1a: $f(a) = 1$ and $f(b) = 1$. At least one of $f(z^*) = 0$ and $f(z_*) = 0$ must hold.

Assume that $f(z_*) = 0$, the other case can be handled in a similar way. Let $g(x, y) = 1 \iff [(x \sqsubseteq a) \wedge (y \sqsubseteq b)]$ and note that g is 2-monotone so $g \in \Gamma$.

Let d be an arbitrary element in D' such that $g(d, d) = 1$. From the definition of g we know that $d \sqsubseteq a, b$ so $d \sqsubseteq z_*$ which implies that $d = z_*$. Furthermore, we have $g(a, b) = 1, f(a) = f(b) = 1$, and $f(z_*) = 0$. Let $D'' = \{x \in D' \mid f(x) = 1\}$. By applying Theorem 33 to $g|_{D''}$, we see that MAX CSP($\Gamma\big|_{D''}$)-B is hard to approximate. Now Lemma 40 implies the result for MAX CSP($\Gamma\big|_{D'}$)-B, and hence for MAX CSP(Γ)-B.

Subcase 1b: $f(a) = 1$ and $f(b) = 0$. In this case, $f(z^*) = 0$ and $f(z_*) = 0$ holds.

If there exists $d \in D'$ such that $b \sqsubset d \sqsubset z^*$ and $f(d) = 1$, then we get $f(a) = 1$, $f(d) = 1$, $a \sqcup d = z^*$ and $f(z^*) = 0$, so this case can be handled by Subcase 1a. Assume that such an element d does not exist.

Let $u(x) = 1 \iff b \sqsubseteq x$. The predicate u is 2-monotone so $u \in \Gamma$. Let $h(x) = f|_{D'}(x) + u|_{D'}(x)$. By the observation above, this is a strict implementation. By Lemmas 12 and 9, it is sufficient to prove the result for $\Gamma' = \Gamma|_{D'} \cup \{h\}$. This can be done exactly as in the previous subcase, with $D'' = \{x \in D' \mid h(x) = 1\}$.

Case 2: f is binary. We now assume that Case 1 does not apply. By Theorem 47, there exist a_1, a_2, b_1, b_2 such that

$$f(a_1, a_2) + f(b_1, b_2) \not\leq f(a_1 \sqcup b_1, a_2 \sqcup b_2) + f(a_1 \sqcap b_1, a_2 \sqcap b_2) \qquad (2)$$

where a_1, b_1 are comparable and a_2, b_2 are comparable. Note that we cannot have $a_1 \sqsubseteq b_1$ and $a_2 \sqsubseteq b_2$, because then the right hand side of (2) is equal to $f(b_1, b_2) + f(a_1, a_2)$ which is a contradiction. Hence, we can without loss of generality assume that $a_1 \sqsubseteq b_1$ and $b_2 \sqsubseteq a_2$.

As in Case 1, we will use Lemma 40 to restrict our domain. In this case, we will consider the subdomain $D' = \{x \in D \mid z_* \sqsubseteq x \sqsubseteq z^*\}$ where $z_* = a_1 \sqcap b_2$ and $z^* = a_2 \sqcup b_1$. As the two predicates $u_{z^*}(x)$ and $u_{z_*}(x)$, defined by $u_{z^*}(x) = 1 \iff x \sqsubseteq z^*$ and $u_{z_*}(x) = 1 \iff z_* \sqsubseteq x$, are 2-monotone predicates and members of Γ, Lemma 40 tells us that it is sufficient to prove hardness for MAX CSP(Γ')-B where $\Gamma' = \Gamma\Big|_{D'}$.

We define the functions $t_i : \{0, 1\} \to \{a_i, b_i\}$, $i = 1, 2$ as follows:

- $t_1(0) = a_1$ and $t_1(1) = b_1$;
- $t_2(0) = b_2$ and $t_2(1) = a_2$.

Hence, $t_i(0)$ is the least element of a_i and b_i and $t_i(1)$ is the greatest element of a_i and b_i.

Our strategy will be to reduce a certain Boolean MAX CSP problem to MAX CSP(Γ')-B. Define three Boolean predicates as follows: $g(x, y) = f(t_1(x), t_2(y))$, $c_0(x) = 1 \iff x = 0$, and $c_1(x) = 1 \iff x = 1$. One can verify that MAX CSP$(\{c_0, c_1, g\})$-B is hard to approximate for each possible choice of g, by using Theorem 49; consult Table 1 for the different possibilities of g.

The following 2-monotone predicates (on D') will be used in the reduction:

$$h_i(x, y) = 1 \iff [(x \sqsubseteq z_*) \wedge (y \sqsubseteq t_i(0))] \vee [(z^* \sqsubseteq x) \wedge (t_i(1) \sqsubseteq y)], i = 1, 2.$$

Table 1
Possibilities for g.

x	y	$t_1(x)$	$t_2(y)$	$g(x,y)$				
0	0	a_1	b_2	0	0	0	0	1
0	1	a_1	a_2	1	1	0	1	1
1	0	b_1	b_2	1	0	1	1	1
1	1	b_1	a_2	1	0	0	0	0

The predicates h_1, h_2 are 2-monotone so they belong to Γ'. We will also use the following predicates:

- $L_d(x) = 1 \iff x \sqsubseteq d$,
- $G_d(x) = 1 \iff d \sqsubseteq x$, and
- $N_{d,d'}(x) = 1 \iff (x \sqsubseteq d) \vee (d' \sqsubseteq x)$

for arbitrary $d, d' \in D'$. These predicates are 2-monotone.

Let w be an integer such that MAX CSP($\{g, c_0, c_1\}$)-w is hard to approximate; such an integer exists according to Theorem 49. Let $\mathcal{I} = (V, C)$, where $V = \{x_1, x_2, \ldots, x_n\}$ and $C = \{C_1, \ldots, C_m\}$, be an instance of MAX CSP($\{g, c_0, c_1\}$)-w. We will construct an instance $\mathcal{I}'$ of MAX CSP(Γ')-w', where $w' = 8w + 5$, as follows:

1. For every $C_i \in C$ such that $C_i = g(x_j, x_k)$, introduce
   (a) two fresh variables y_j^i and y_k^i,
   (b) the constraint $f(y_j^i, y_k^i)$,
   (c) $2w + 1$ copies of the constraints $L_{b_1}(y_j^i), G_{a_1}(y_j^i), N_{a_1,b_1}(y_j^i)$,
   (d) $2w + 1$ copies of the constraints $L_{a_2}(y_k^i), G_{b_2}(y_k^i), N_{b_2,a_2}(y_k^i)$, and
   (e) $2w + 1$ copies of the constraints $h_1(x_j, y_j^i), h_2(x_k, y_k^i)$.
2. for every $C_i \in C$ such that $C_i = c_0(x_j)$, introduce the constraint $L_{z_*}(x_j)$, and
3. for every $C_i \in C$ such that $C_i = c_1(x_j)$, introduce the constraint $G_{z^*}(x_j)$.

The intuition behind this construction is as follows: due to the bounded occurrence property and the quite large number of copies of the constraints in steps 1c, 1d and 1e, all of those constraints will be satisfied in "good" solutions. The elements 0 and 1 in the Boolean problem corresponds to z_* and z^*, respectively. This may be seen in the constraints introduced in steps 2 and 3. The constraints introduced in step 1c essentially force the variables y_j^i to be either a_1 or b_1, and the constraints in step 1d work in a similar way. The constraints in step 1e work as bijective mappings from the domains $\{a_1, b_1\}$ and $\{a_2, b_2\}$ to $\{z_*, z^*\}$. For example, $h_1(x_j, y_j^i)$ will set x_j to z_* if y_j^i is a_1, otherwise if y_j^i is b_1, then x_j will be set to z^*. Finally, the constraint introduced in step 1b corresponds to $g(x_j, x_k)$ in the original problem.

It is clear that $\mathcal{I}'$ is an instance of Max CSP(Γ')-w'. Note that due to the bounded occurrence property of $\mathcal{I}'$, a solution which does not satisfy all constraints introduced in steps 1c, 1d and 1e can be used to construct a new solution which satisfies those constraints and has a measure which is greater than or equal to the measure of the original solution. We will denote this transformation of solutions by P.

Given a solution s' to $\mathcal{I}'$, we can construct a solution $s = G(s')$ to $\mathcal{I}$ by, for every $x \in V$, letting $s(x) = 0$ if $P(s')(x) = z_*$ and $s(x) = 1$, otherwise.

Let M be the number of constraints in C of type g. We have that, for an arbitrary solution s' to $\mathcal{I}'$, $m(\mathcal{I}', P(s')) = m(\mathcal{I}, G(s')) + 8(2w+1) \cdot M \geq m(\mathcal{I}', s')$. Furthermore, $\textsc{opt}(\mathcal{I}') = \textsc{opt}(I) + 8(2w+1)M$.

Now, assume that $\textsc{opt}(\mathcal{I}')/m(\mathcal{I}', s') \leq \varepsilon'$. Then $\textsc{opt}(\mathcal{I}')/m(\mathcal{I}', P(s')) \leq \varepsilon'$ and

$$\begin{aligned}
&\frac{\textsc{opt}(I) + 8(2w+1)M}{m(I, G(s')) + 8(2w+1)M} \leq \varepsilon' && \Rightarrow \\
&\textsc{opt}(I) \leq \varepsilon' m(I, G(s')) + (\varepsilon' - 1)8(2w+1)M && \Rightarrow \\
&\frac{\textsc{opt}(\mathcal{I})}{m(\mathcal{I}, G(s'))} \leq \varepsilon' + \frac{8(2w+1)M(\varepsilon'-1)}{m(\mathcal{I}, G(s'))}.
\end{aligned}$$

Furthermore, by standard arguments, we can assume that $m(\mathcal{I}, G(s')) \geq |C|/c$, for some constant c. We get,

$$\frac{\textsc{opt}(\mathcal{I})}{m(\mathcal{I}, G(s'))} \leq \varepsilon' + 8(2w+1)c(\varepsilon' - 1).$$

Hence, a polynomial time approximation algorithm for Max CSP(Γ')-w' with performance ratio ε' can be used to obtain ε''-approximate solutions, where ε'' is given by $\varepsilon' + 8(2w+1)c(\varepsilon'-1)$, for Max CSP$(\{c_0, c_1, g\})$-$w$ in polynomial time. Note that ε'' tends to 1 as ε' approaches 1. This implies that Max CSP(Γ')-w' is hard to approximate because Max CSP$(\{c_0, c_1, g\})$-w is hard to approximate. □

5 Conclusions and Future Work

This article has two main results: the first one is that Max CSP(Γ) has a hard gap at location 1 whenever Γ satisfies a certain condition which makes CSP(Γ) **NP**-hard. This condition captures all constraint languages which are currently known to make CSP(Γ) **NP**-hard. This condition has also been conjectured to be the dividing line between tractable (in **P**) CSPs and **NP**-hard CSPs.

The second result is that single relation MAX CSP is either trivial or hard to approximate.

It is possible to strengthen these results in a number of ways. The following possibilities applies to both of our results.

We have paid no attention to the constant which we prove inapproximability for. That is, given a constraint language Γ, what is the smallest constant c such that MAX CSP(Γ) is not approximable within $c-\varepsilon$ for any $\varepsilon > 0$ in polynomial time? For some relations a lot of work has been done in this direction, cf. [6,32,40,56] for more details. As mentioned in the introduction Raghavendra's result [52] give almost optimal approximability results for all constraint languages, assuming the UGC. The methods used to obtain good constants are based on sophisticated PCP constructions, semidefinite programming and the UGC. We note that these techniques are very different from the ones we have used in this paper. At present it seems difficult to use the algebraic techniques to obtain good constants.

We have a constant number of variable occurrences in our hardness results, but the constant is unspecified. For some problems, for example MAX 2SAT, it is known that allowing only three variable occurrences still makes the problem hard to approximate (even **APX**-hard) [6]. This is also true for some other MAX CSP problems such as MAX CUT [1]. However, there are CSP problems which are **NP**-hard but which becomes easy if the number of variable occurrences are restricted to three. In particular, it is known that for each $k \geq 3$ there is an integer $f(k)$ such that if $s \leq f(k)$ then k-SAT-s (the satisfiability problem with clauses of length k and at most s occurrences of each variable) is trivial (every instance is satisfiable) and otherwise, if $s > f(k)$, then the problem is **NP**-complete. Some bounds are also known for f but the exact behaviour remains unknown [41]. As every instance is satisfiable the corresponding maximisation problem MAX k-SAT-s is also trivial for $s \leq f(k)$. This leads to the following problem: find the smallest integer $k(\Gamma)$ such that MAX CSP(Γ)-$k(\Gamma)$ is hard to approximate, for constraint languages Γ which satisfies the condition in Lemma 21 (so CSP(Γ) is **NP**-complete). One can also ask the same question for a single non-empty non-valid relation R: find the smallest integer $k(R)$ so that MAX CSP($\{R\}$)-$k(R)$ is hard to approximate.

One of the main open problems is to classify MAX CSP(Γ) for all constraint languages Γ, with respect to tractability of finding an optimal solution. The current results in this direction [17,24,36,43] seems to indicate that the concept of *supermodularity* is of central importance for the complexity of MAX CSP. However, the problem is open on both ends — we do not know if supermodularity implies tractability and neither do we know if non-supermodularity implies non-tractability. Here "non-tractability" should be interpreted as "not in **PO**" under some suitable complexity-theoretic assumption, the questions of **NP**-hardness and approximation hardness are, of course, also open.

Acknowledgements.

The authors would like to thank Gustav Nordh for comments which have improved the presentation of this paper. Peter Jonsson is partially supported by the *Center for Industrial Information Technology* (CENIIT) under grant 04.01, and by the *Swedish Research Council* (VR) under grant 621-2003-3421. Andrei Krokhin is supported by the UK EPSRC grant EP/C543831/1. Fredrik Kuivinen is supported by the *National Graduate School in Computer Science* (CUGS), Sweden.

References

[1] P. Alimonti, V. Kann, Some APX-completeness results for cubic graphs., Theor. Comput. Sci. 237 (1-2) (2000) 123–134.

[2] S. Arora, Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems, J. ACM 45 (5) (1998) 753–782.

[3] S. Arora, C. Lund, Hardness of approximations, in: D. Hochbaum (ed.), Approximation Algorithms for NP-hard Problems, chap. 10, PWS Publishing, Boston, MA, USA, 1997, pp. 399–446.

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

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

[6] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, M. Protasi, Complexity and approximation: Combinatorial Optimization Problems and their Approximability Properties, Springer, 1999.

[7] L. Barto, M. Kozik, T. Niven, The CSP dichotomy holds for digraphs with no sources and no sinks (a positive answer to a conjecture of Bang-Jensen and Hell), SIAM J. Comput. 38 (5) (2009) 1782–1802.

[8] A. Bulatov, Tractable conservative constraint satisfaction problems, in: Proceedings of the 18th Annual IEEE Symposium on Logic in Computer Science (LICS '03), IEEE Computer Society, Washington, DC, USA, 2003.

[9] A. Bulatov, H-coloring dichotomy revisited, Theor. Comput. Sci. 349 (1) (2005) 31–39.

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

[11] A. Bulatov, V. Dalmau, A simple algorithm for Mal'tsev constraints, SIAM J. Comput. 36 (1) (2006) 16–27.

[12] A. Bulatov, P. Jeavons, Algebraic structures in combinatorial problems, Tech. Rep. MATH-AL-4-2001, Technische Universität Dresden (2001).

[13] A. Bulatov, P. Jeavons, A. Krokhin, Classifying the complexity of constraints using finite algebras, SIAM J. Comput. 34 (3) (2005) 720–742.

[14] A. Bulatov, A. Krokhin, P. Jeavons, The complexity of maximal constraint languages, in: Proceedings of the thirty-third Annual ACM Symposium on Theory of Computing (STOC '01), ACM Press, New York, NY, USA, 2001.

[15] S. Burris, H. Sankappanavar, A Course in Universal Algebra, Springer Verlag, Berlin, 1981.
URL `http://citeseer.ist.psu.edu/sankappanavar81course.html`

[16] M. Charikar, K. Makarychev, Y. Makarychev, Near-optimal algorithms for unique games, in: Proceedings of the thirty-eighth Annual ACM Symposium on Theory of Computing (STOC '06), ACM Press, New York, NY, USA, 2006.

[17] D. Cohen, M. Cooper, P. Jeavons, A. Krokhin, Supermodular functions and the complexity of Max CSP, Discrete Appl. Math. 149 (1-3) (2005) 53–72.

[18] D. Cohen, P. Jeavons, The complexity of constraint languages, in: F. Rossi, P. van Beek, T. Walsh (eds.), Handbook of Constraint Programming, chap. 8, Elsevier, 2006, pp. 245–280.

[19] P. Cohn, Universal Algebra, No. 6 in Mathematics and its Applications, Reidel, 1981, originally published by Harper and Row, 1965.

[20] N. Creignou, A dichotomy theorem for maximum generalized satisfiability problems, J. Comput. Syst. Sci. 51 (3) (1995) 511–522.

[21] N. Creignou, S. Khanna, M. Sudan, Complexity Classifications of Boolean Constraint Satisfaction Problems, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001.

[22] P. Crescenzi, A short guide to approximation preserving reductions, in: Proceedings of the 12th Annual IEEE Conference on Computational Complexity (CCC '97), IEEE Computer Society, Washington, DC, USA, 1997.

[23] V. Dalmau, P. Jeavons, Learnability of quantified formulas, Theor. Comput. Sci. 306 (1-3) (2003) 485–511.

[24] V. Deineko, P. Jonsson, M. Klasson, A. Krokhin, The approximability of MAX CSP with fixed-value constraints, J. ACM 55 (4) (2008) 1–37.

[25] B. L. Dietrich, A. J. Hoffman, On greedy algorithms, partially ordered sets, and submodular functions, IBM J. Res. Dev. 47 (1) (2003) 25–30.

[26] I. Dinur, The PCP theorem by gap amplification, J. ACM 54 (3) (2007) 12.

[27] L. Engebretsen, J. Holmerin, A. Russell, Inapproximability results for equations over finite groups, Theor. Comput. Sci. 312 (1) (2004) 17–45.

[28] T. Feder, P. Hell, J. Huang, List homomorphisms of graphs with bounded degrees, Discrete Math. 307 (2007) 386–392.

[29] T. Feder, 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) (1998) 57–104.

[30] M. Goldmann, A. Russell, The complexity of solving equations over finite groups., Inf. Comput. 178 (1) (2002) 253–262.

[31] P. Hell, J. Nešetřil, Graphs and Homomorphisms, Oxford University Press, 2004.

[32] J. Håstad, Some optimal inapproximability results, J. ACM 48 (4) (2001) 798–859.

[33] O. H. Ibarra, C. E. Kim, Fast approximation for the knapsack and sum of subset problems, J. ACM 22 (4) (1975) 463–468.

[34] P. Jeavons, On the algebraic structure of combinatorial problems, Theor. Comput. Sci. 200 (1-2) (1998) 185–204.

[35] P. Jeavons, D. Cohen, M. Gyssens, Closure properties of constraints, J. ACM 44 (1997) 527–548.

[36] P. Jonsson, M. Klasson, A. Krokhin, The approximability of three-valued Max CSP, SIAM J. Comput. 35 (6) (2006) 1329–1349.

[37] P. Jonsson, A. Krokhin, Maximum H-colourable subdigraphs and constraint optimization with arbitrary weights, J. Comput. System Sci. 73 (5) (2007) 691–702.

[38] S. Khanna, M. Sudan, L. Trevisan, D. P. Williamson, The approximability of constraint satisfaction problems., SIAM J. Comput. 30 (6) (2000) 1863–1920.

[39] S. Khot, On the power of unique 2-prover 1-round games, in: Proceedings of the thiry-fourth Annual ACM Symposium on Theory of Computing (STOC '02), ACM Press, New York, NY, USA, 2002.

[40] S. Khot, G. Kindler, E. Mossel, R. O'Donnell, Optimal inapproximability results for Max-Cut and other 2-variable CSPs?, SIAM J. Comput. 37 (1) (2007) 319–357.

[41] J. Kratochvíl, P. Savický, and Z. Tuza, One more occurrence of variables makes satisfiability jump from trivial to NP-complete, SIAM J. Comput. 22 (1) (1993) 203–210.

[42] A. Krokhin, B. Larose, Maximum constraint satisfaction on diamonds, Tech. Rep. CS-RR-408, University of Warwick, UK (2004).

[43] A. Krokhin, B. Larose, Maximum constraint satisfaction on diamonds, in: Principles and Practice of Constraint Programming (CP '05), Springer, 2005.

[44] R. E. Ladner, On the structure of polynomial time reducibility, J. ACM 22 (1) (1975) 155–171.

[45] B. Larose, L. Zádori, Taylor terms, constraint satisfaction and the complexity of polynomial equations over finite algebras, Internat. J. Algebra Comput. 16 (3) (2006) 563–581.

[46] R. Lipton, R. Tarjan, Applications of a planar separator theorem, SIAM J. Comput. 9 (1980) 615–627.

[47] A. Lubotzky, R. Phillips, P. Sarnak, Ramanujan graphs, Combinatorica 8 (3) (1988) 261–277.

[48] M. Maróti, R. McKenzie, Existence theorems for weakly symmetric operations, Algebra Universalis 59 (3) (2008) 463–489.

[49] C. H. Papadimitriou, M. Yannakakis, Optimization, approximation, and complexity classes, J. Comput. System Sci. 43 (1991) 425–440.

[50] E. Petrank, The hardness of approximation: gap location, Computational Complexity 4 (1994) 133–157.

[51] R. Pöschel, L. Kalužnin, Funktionen- und Relationenalgebren, DVW, Berlin, 1979.

[52] P. Raghavendra, Optimal algorithms and inapproximability results for every CSP?, in: Proceedings of the fortieth Annual ACM Symposium on Theory of Computing (STOC '08), ACM Press, New York, NY, 2008.

[53] F. Rossi, P. van Beek, T. Walsh (eds.), Handbook of Constraint Programming, Elsevier, 2006.

[54] T. J. Schaefer, The complexity of satisfiability problems, in: Proceedings of the tenth Annual ACM Symposium on Theory of Computing (STOC '78), ACM Press, New York, NY, USA, 1978.

[55] Á. Szendrei, Clones in Universal Algebra, vol. 99 of Séminaire de Mathématiques Supérieures, University of Montreal, 1986.

[56] L. Trevisan, Inapproximability of combinatorial optimization problems, arXiv.org:cs.CC/0409043 (2004).