Ruling out polynomial-time approximation schemes for hard constraint satisfaction problems

Peter Jonsson[1], Andrei Krokhin[2] and Fredrik Kuivinen[1]

[1] Department of Computer and Information Science
Linköpings Universitet, SE-581 83, Linköping, Sweden
{petej|freku}@ida.liu.se
[2] Department of Computer Science
Durham University, Durham, DH1 3LE, UK
andrei.krokhin@durham.ac.uk

Abstract. The *maximum constraint satisfaction problem* (Max CSP) is the following computational problem: an instance 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 so is **NP**-hard in general. It is natural to study how restrictions on the allowed constraint types (or constraint language) affect the complexity and approximability of Max CSP. All constraint languages, for which the CSP problem (i.e., the problem of deciding whether all constraints in an instance can be simultaneously satisfied) is currently known to be **NP**-hard, have a certain algebraic property, and it has been conjectured that CSP problems are tractable for all other constraint languages. We prove that any constraint language with this algebraic property makes Max CSP hard at gap location 1, thus ruling out the existence of a polynomial-time approximation scheme for such problems. We then apply this result to Max CSP restricted to a single constraint type. We show that, unless $\mathbf{P} = \mathbf{NP}$, such problems either are trivial or else do not admit polynomial-time approximation schemes. All our hardness results hold even if the number of occurrences of each variable is bounded by a constant.

Keywords. maximum constraint satisfaction, complexity, approximability

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 collections of 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, Euclidean Tsp, and Independent Set restricted to planar graphs [1]. It is also well-known that a large number of optimisation problems do not admit a PTAS unless some unexpected collapse of complexity classes occurs. For instance, problems like Max k-SAT and Independent Set do not admit a PTAS unless $\mathbf{P} = \mathbf{NP}$ [1]. 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.

Constraint satisfaction problems (CSP) [19] and its optimisation variants have played an important role in research on approximability. Many combinatorial problems are subsumed by the CSP framework, and examples include problems in graph theory [10], combinatorial optimisation [13], and computational learning [7]. 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 are Max k-SAT and Max Cut.

Let D be a finite set. A subset $R \subseteq D^n$ is called a *relation* and n is the *arity* of R. Let $R_D^{(k)}$ denote 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 constraint language Γ, denoted* CSP(Γ)*, is defined to be the decision problem with instance (V, C), where*

- *V is a finite set of variables, and*
- *C is a (multi)set 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. A constraint $(R_i, (v_1, v_2, \ldots, v_{n_i})) \in C$ is satisfied *by an assignment s if the image of the constraint scope is a member of the constraint relation, i.e., if $(s(v_1), s(v_2), \ldots, s(v_{n_i})) \in R_i$.*

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 *multisets* of constraints instead of just sets of constraints as we do not have any weights in our definition of Max CSP. We chose to use multisets instead of weights because bounded occurrence restrictions are easier to explain in the multiset 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.

Example 1. 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 [1]), 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 Max CSP were restricted to the Boolean case, i.e. when $D = \{0, 1\}$. For instance, Khanna et al. [13] characterise the approximability of Max CSP(Γ) for all Γ over the Boolean domain. It has been noted that the study of non-Boolean CSP gives 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 [3] and also given a classification of constraint languages that contain all unary relations [2]. Corresponding results for Max CSP have been obtained by Jonsson et al. [11] and Deineko et al. [8].

It has been conjectured [4] that, for all constraint languages Γ, CSP(Γ) is either in **P** or is **NP**-complete (i.e., it cannot be **NP**-intermediate), and the conjecture also specifies the dividing line between the two cases, by means of a certain algebraic condition. Moreover, it was shown in [4] that, for all constraint languages Γ satisfying this condition, the problem CSP(Γ) is indeed **NP**-complete. In this paper we prove that, for such languages Γ, it is **NP**-hard to tell 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). In particular, this implies that, for such Γ, the problem Max CSP(Γ) cannot admit a PTAS.

We then apply this result to study the case when the constraint language Γ consist of a single relation R. We show that, for such Γ, Max CSP(Γ) is either trivial or else does not admit a PTAS. Finally, we use this last result to strengthen several earlier hardness results obtained in the study of Max CSP via the algebraic property of supermodularity.

We obtain our results by techniques which are quite different from the ones used in [8, 11]. In [11] it was proved that constraint languages over a three element domain which are cores and not *supermodular* (see Section 5 for a definition) give rise to Max CSP-problems which do not admit a PTAS (it is in fact proved that they are **APX**-hard, which implies that they do not admit a PTAS, unless **P** = **NP**). The technique used in [11] is mainly that of *strict implementations.* With strict implementations, certain new relations can be constructed from old ones in a way that preserve the hardness of the corresponding Max CSP-problem. That is, if Γ is a constraint language which strictly implements a relation R, then Max CSP($\Gamma \cup \{R\}$) is no harder than Max CSP(Γ). This technique can be used to reduce the huge number of constraint languages to a set of constraint languages which is easier to reason about. Hardness results for this smaller set of constraint languages are then obtained from known results.

Let us fix a finite domain D and let U_D be the set of all unary constraints on D, that is $U_D = \{R \mid R \subseteq D\}$. In [8] it was proved that Max CSP($\Gamma \cup U_D$) is **APX**-hard (and, therefore do not admit a PTAS) if $\Gamma \cup U_D$ is not supermodular on any chain (a *chain* is a lattice which is a total order) on D, and tractable (in **PO**) otherwise. The proof of this result uses a characterisation of relations which are supermodular on some chain together with strict implementations.

In contrast to the two results described above, we obtain the results in the present paper by quite different means. Our main result is proved by using perfect implementations and the associated theory of universal algebra. Universal algebra have previously been successfully used to classify the complexity of CSPs (we give an overview of this connection in Section 3, see also [4]). By using the notion of *hardness at gap location 1* and working with bounded occurrence Max CSP-problems (this was also done in [8]) we manage to prove not only **NP**-hardness results, but also the impossibility of a PTAS (unless **P** = **NP**). Proofs of our results are omitted due to space constraints.

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.

Definition 3 (Performance ratio). *A solution $s \in \mathsf{sol}(\mathcal{I})$ to an instance $\mathcal{I}$ of a optimisation 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 optimisation 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.*

Definition 4 (PTAS). *An optimisation problem Π admits a* PTAS *if, for any rational constant $c > 1$, there is an algorithm that, given an instance $\mathcal{I}$ of Π, returns a c-approximate solution in time polynomial in $|\mathcal{I}|$.*

Definition 5 (Hard to approximate). *We say that a problem Π is* hard to approximate *if there exists a constant c such that it is* **NP***-hard to approximate Π with performance ratio c.*

Obviously, any problem that is hard to approximate cannot admit a PTAS. The following notion has been defined in a more general setting in [17].

Definition 6 (Hard gap at location α). Max CSP(Γ) *has a* hard gap at location $\alpha \leq 1$ *if there exists a constant* $\varepsilon < \alpha$ *such that it is* **NP***-hard to decide if, for a given instance* $\mathcal{I} = (V, C)$ *of* Max CSP(Γ), $\textsc{opt}(\mathcal{I}) \geq \alpha|C|$ *or* $\textsc{opt}(\mathcal{I}) \leq \varepsilon|C|$.

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., [1, 20] for surveys on inapproximability results and the related PCP theory.

Petrank [17] gave an informal conjecture which states that for "natural" optimisation problems hardness at gap location 1 can be used to show hardness at all other possible gap locations. Sometimes this can be done by a padding argument. Given an instance $\mathcal{I} = (V, C)$ of Max CSP(Γ), we can add fresh variables V' and constraints C' to $\mathcal{I}$ such that at most a constant fraction ε of the constraints in C' can be satisfied simultaneously. If we choose C' and V' appropriately then we will obtain a proof of hardness at gap location α (for some α which depends on ε) for Max CSP(Γ). Hence, in this sense hardness at gap location 1 is a stronger result than hardness at any other gap location. Petrank [17] give further arguments for why hardness at gap location 1 is a natural and interesting hardness notion.

Throughout the paper, Max CSP(Γ)-k denotes the problem Max CSP(Γ) restricted to instances where the number of occurrences of each variable is bounded by k. Note that 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. The bounded occurrence property is closely related to bounding the degree in graphs. Re-considering Example 1, the problem Max CSP($\{N_2\}$)-3 would correspond to Max Cut restricted to (multi)graphs with maximum degree 3. In 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.

3 Hardness at Gap Location 1 for Max CSP

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 [5]. The article [4] contains a presentation 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:

$$f(\boldsymbol{t_1}, \boldsymbol{t_2}, \ldots, \boldsymbol{t_k}) = (f(\boldsymbol{t_1}[1], \boldsymbol{t_2}[1], \ldots, \boldsymbol{t_k}[1]), \ldots, f(\boldsymbol{t_1}[n], \boldsymbol{t_2}[n], \ldots, \boldsymbol{t_k}[n])),$$

where $\boldsymbol{t_j}[i]$ is the i-th component in tuple $\boldsymbol{t_j}$. An operation $f : D^k \to D$ is said to be *idempotent* if $f(d, d, \ldots, d) = d$ for all $d \in D$, and it is called a *projection* if there is $1 \leq i \leq k$ such that $f(\boldsymbol{x}) = x_i$, for all $\boldsymbol{x} = (x_1, x_2, \ldots, x_k) \in D^k$.

Let R_i be a relation in the constraint language Γ. If f is an operation such that for all $\boldsymbol{t_1}, \boldsymbol{t_2}, \ldots, \boldsymbol{t_k} \in R_i$ we have $f(\boldsymbol{t_1}, \boldsymbol{t_2}, \ldots, \boldsymbol{t_k}) \in R_i$, then R_i is said to be *invariant* under f. If all 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 $\mathrm{Pol}(\Gamma)$. For a set of operations F, the set of all relations which are invariant under each operation in F is denoted $\mathrm{Inv}(F)$.

Example 2. 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.

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) using relations from $\Gamma \cup \{EQ_D\}$. 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 $I(x, y) \iff \exists z : A(x, z) \wedge B(z, y)$.

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 1 ([18]).

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

By using this connection between polymorphisms and relations definable by pp-formulae we obtain the following lemma. This lemma allows us to use some of the algebraic theory, which is commonly used when studying CSP, to get hardness results for Max CSP.

Lemma 1. *Let Γ be a constraint language and let R be a relation which is definable by a pp-formula using relations from Γ. If* 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'.*

The notions of a core and a retraction play an important role in the study of graphs, and they can easily be generalised to constraint languages. A *retraction* of a constraint language Γ is a unary polymorphism $\pi \in \mathrm{Pol}(\Gamma)$ such that

$\pi(x) = x$ for all x in the image of π. 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 minimal image D', then a core of Γ is the set $\Gamma\big|_{D'}$. As in the case of graphs, all cores of Γ are isomorphic, so one can speak about *the* core of Γ.

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. Hence, the problem is equivalent to a problem over this smaller set.

Lemma 2. *If Γ' is the core of Γ, then, for any k,* Max CSP(Γ')-k *is hard at gap location 1 if and only if* Max CSP(Γ)-k *is hard at gap location 1.*

The three definitions below closely follows the presentation in [4].

Definition 7 (Finite algebra). *A* finite algebra *is a pair $\mathcal{A} = (A; F)$ where A is a finite non-empty set and $F = \{f_i^A \mid i \in I\}$ 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 8 (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 $\mathcal{A}$.*

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 9 (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 F consists of 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 $\mathcal{A}$. The following theorem concerns the hardness of CSP for certain constraint languages.

Theorem 2 ([4]). *Let Γ be a constraint language and let Γ' be its core. If the algebra $\mathcal{A}^c_{\Gamma'}$ has a non-trivial factor whose term operations are only projections, then* CSP(Γ) *is* **NP***-hard.*

It has been conjectured [4] that, for any other core languages Γ, the problem CSP(Γ) is tractable, and this conjecture has been verified in many important cases (see, e.g., [2, 3]).

Our first result, Theorem 3, shows that the problems from the above theorem are not only **NP**-hard, but also the corresponding optimisation problems are hard at gap location 1, which rules out the existence of PTAS for such problems. By Lemma 2, it is sufficient to prove this for core constraint languages.

Theorem 3. *Let Γ be a core constraint language. If $\mathcal{A}^c_{\Gamma}$ has a non-trivial factor whose term operations are only projections, then* Max CSP(Γ)-B *is hard at gap location 1.*

There are four basic ingredients in the proof of Theorem 3. The first two are Lemma 1 and the rather standard use of expander graphs to bound the number of variable occurrences (see, e.g., Section 8.4.1 of [1]). We also use the following alternative technical characterisation (obtained in the proof of Proposition 7.9 of [4]) of constraint languages satisfying the conditions of the theorem.

The *not all equal* relation contains the tuples $(0,0,1)$, $(0,1,0)$, $(1,0,0)$, $(1,1,0)$, $(1,0,1)$, and $(0,1,1)$, we denote this relation by NAE. We denote set of all singleton unary relations by C_D, that is, for a finite domain D we have $C_D = \{\{(x)\} \mid x \in D\}$.

Lemma 3. *Let Γ be a core constraint language. The following are equivalent:*

- *The algebra $\mathcal{A}^c_{\Gamma}$ has a non-trivial factor whose term operations are only projections.*
- *There exists 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 relation $\varphi^{-1}(NAE)$ which is the full preimage (under φ) of NAE.*

The final ingredient in the proof of Theorem 3 is that the problem Max Not-All-Equal 3Sat is hard at gap location 1, which was proved in [17].

Note that if CSP(Γ) is tractable, then Max CSP(Γ) cannot be hard at gap location 1. Hence, if the above conjecture from [4] holds, our result describes all problems Max CSP(Γ) that are hard at gap location 1.

It is not hard to see that hardness at gap location 1 rules out the existence of PTAS even when Max CSP(Γ)-B is restricted to satisfiable instances (i.e., those where all constraints can be simultaneously satisfied).

Corollary 1. *Under the assumptions of Theorem 3, there exists a constant c (depending on Γ) such that* Max CSP(Γ)-B *restricted to satisfiable instances cannot be approximated within c in polynomial time (unless* $\mathbf{P} = \mathbf{NP}$*).*

The following conjecture has been made by Feder et al. [9].

Conjecture 1. For any fixed Γ such that $\mathrm{CSP}(\Gamma)$ is **NP**-complete, there is an integer k such that $\mathrm{CSP}(\Gamma)$-k is **NP**-complete.

Under the assumption that the dichotomy conjecture (that all problems $\mathrm{CSP}(\Gamma)$ not covered by Theorem 2 are tractable) holds, an affirmative answer follows immediately from Theorem 3. So, for all constraint languages Γ such that $\mathrm{CSP}(\Gamma)$ is currently known to be **NP**-complete, it is also the case that $\mathrm{CSP}(\Gamma)$-B is **NP**-complete.

4 Approximability of Single Relation Max CSP

A relation R is said to be *d-valid* if $(d, \ldots, d) \in R$ for $d \in D$, and simply *valid* if it is is d-valid for some $d \in D$. It was proved in [12] that every problem Max CSP($\{R\}$) with R neither empty nor valid is **NP**-hard. We strengthen this result by proving that the problems are not only **NP**-hard but also cannot have a PTAS. Note that for some Max CSP problems such approximation hardness results are known, e.g., for Max Cut and Max DiCut (see Example 1). Our result extends those hardness results to all possible relations.

Theorem 4. *Let $R \in R_D$ be non-empty. If $(d, \ldots, d) \in R$ for some $d \in D$ then* Max CSP($\{R\}$) *is trivial. Otherwise,* Max CSP($\{R\}$)-B *is hard to approximate.*

For a constraint language Γ, let $\mathrm{Aut}(\Gamma)$ denote the permutation group consisting of injective unary polymorphisms of Γ. Recall that a permutation group $\mathcal{G}$ on a set D is called *transitive* if, for every $d, d' \in D$, there exists $g \in \mathcal{G}$ such that $g(d) = d'$. A digraph $G = (V, E)$ is said to be *vertex-transitive* if the permutation group $\mathrm{Aut}(\{E\})$ is transitive.

After proving a couple of lemmas which reduce the set of relations one needs to consider to prove Theorem 4, the relations which are left are edge relations of vertex-transitive digraphs. In [16] the following characterisation of the complexity of the CSP problem for such relations was given. By deriving an algebraic characterisation of this result we can use Theorem 3 together with certain ideas and techniques (domain restriction, strict implementation) from [12] to prove Theorem 4.

Theorem 5 ([16]). *Let $G = (V, E)$ be a vertex-transitive digraph which is a core. If G is a directed cycle, then* $\mathrm{CSP}(\{E\})$ *is tractable. Otherwise, if G is not a directed cycle, then* $\mathrm{CSP}(\{E\})$ *is* **NP***-complete.*

Theorem 4 can be used to classify approximability of Max CSP(Γ) for constraint languages Γ with sufficiently many symmetries. The following result can be derived from Theorem 4.

Corollary 2. *Let Γ be a constraint language such that* $\mathrm{Aut}(\Gamma)$ *is transitive. If Γ contains a non-empty relation R which is not d-valid for all $d \in D$, then* Max CSP(Γ) *is hard to approximate. Otherwise,* Max CSP(Γ) *is trivial.*

Note that the constraint languages considered in Corollary 2 can be seen as a generalisation of vertex-transitive graphs.

5 Max CSP and Supermodularity

In this section, we present two results whose proofs make use of Theorem 4. These results strengthens earlier published results [14, 15] in various ways (e.g., they apply to a larger class of constraint languages or they give approximation hardness instead of **NP**-hardness).

Recall that a poset $\mathcal{P} = (D, \sqsubseteq)$ is a *lattice* 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 product* of $\mathcal{L}$, denoted by $\mathcal{L}^n$, is the lattice with domain D^n and operations acting componentwise.

Definition 10 (Supermodular function). *Let $\mathcal{L}$ be a lattice on D. A function $f : D^n \to \mathbb{R}$ 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 $\boldsymbol{a}, \boldsymbol{b} \in D^n$.

The *characteristic function* of a n-ary relation R over the domain D is the function $f : D^n \to \{0, 1\}$ such that $f(\boldsymbol{x}) = 1$ iff $\boldsymbol{x} \in R$. Call a relation supermodular if its characteristic function is such. The set of all supermodular relations 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 $\Gamma \subseteq \mathrm{Spmod}_{\mathcal{L}}$.

Supermodularity on lattices plays an important role in the study of Max CSP, as all known tractable cases of Max CSP(Γ) can be explained via this property [6, 8, 11, 15].

The next definition follows [6].

Definition 11 (Generalised 2-monotone). *Given a poset $\mathcal{P} = (D, \sqsubseteq)$, a relation R is said to be* generalised 2-monotone *on $\mathcal{P}$ if*

$$x \in R \iff ((x_{i_1} \sqsubseteq a_{i_1}) \wedge \ldots \wedge (x_{i_s} \sqsubseteq a_{i_s})) \vee ((x_{j_1} \sqsupseteq b_{j_1}) \wedge \ldots \wedge (x_{j_s} \sqsupseteq b_{j_s}))$$

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 relations 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 proposition is a combination of results proved in [6] and [14].

Proposition 1.

- *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 2. *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.*

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. A *chain* is a lattice with total order. The following proposition is a combination of results proved in [6] and [15].

Theorem 6. *Let Γ contain all at most binary 2-monotone predicates on a lattice $\mathcal{L}$ which is either a chain or a diamond. If $\Gamma \not\subseteq \mathrm{Spmod}_{\mathcal{L}}$, then* Max CSP$(\Gamma)$ *is* **NP**-*hard.*

We can strengthen this result in three ways: our result applies to arbitrary lattices, we obtain hardness of approximation instead of **NP**-hardness, and we get the result for bounded occurrence instances.

Theorem 7. *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.*

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. G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and approximation: Combinatorial Optimization Problems and their Approximability Properties.* Springer, 1999.
2. A. Bulatov. Tractable conservative constraint satisfaction problems. In *Proceedings of the 18th Annual IEEE Symposium on Logic in Computer Science (LICS '03)*, pages 321–330, 2003.
3. A. Bulatov. A dichotomy theorem for constraint satisfaction problems on a 3-element set. *J. ACM*, 53(1):66–120, 2006.
4. A. Bulatov, P. Jeavons, and A. Krokhin. Classifying the complexity of constraints using finite algebras. *SIAM J. Comput.*, 34(3):720–742, 2005.
5. S. Burris and H. Sankappanavar. *A Course in Universal Algebra.* Springer Verlag, Berlin, 1981.

6. D. Cohen, M. Cooper, P. Jeavons, and A. Krokhin. Supermodular functions and the complexity of Max CSP. *Discrete Appl. Math.*, 149(1-3):53–72, 2005.
7. V. Dalmau and P. Jeavons. Learnability of quantified formulas. *Theor. Comput. Sci.*, (306):485–511, 2003.
8. V. Deineko, P. Jonsson, M. Klasson, and A. Krokhin. Supermodularity on chains and complexity of maximum constraint satisfaction. In *European Conference on Combinatorics, Graph Theory and Applications (EuroComb '05)*, volume AE of *DMTCS Proceedings*, pages 51–56. Discrete Mathematics and Theoretical Computer Science, 2005. Full version available as "The approximability of Max CSP with fixed-value constraints", arXiv.org:cs.CC/0602075.
9. T. Feder, P. Hell, and J. Huang. List homomorphisms of graphs with bounded degrees. *Discrete Math.*, 307:386–392, 2007.
10. P. Hell and J. Nešetřil. *Graphs and Homomorphisms*. Oxford University Press, 2004.
11. P. Jonsson, M. Klasson, and A. Krokhin. The approximability of three-valued Max CSP. *SIAM J. Comput.*, 35(6):1329–1349, 2006.
12. P. Jonsson and A. Krokhin. Maximum H-colourable subdigraphs and constraint optimization with arbitrary weights. *J. Comput. System Sci.*, 73(5):691–702, 2007.
13. S. Khanna, M. Sudan, L. Trevisan, and D. P. Williamson. The approximability of constraint satisfaction problems. *SIAM J. Comput.*, 30(6):1863–1920, 2000.
14. A. Krokhin and B. Larose. Maximum constraint satisfaction on diamonds. Technical Report CS-RR-408, University of Warwick, UK, 2004.
15. A. Krokhin and B. Larose. Maximum constraint satisfaction on diamonds. In *Principles and Practice of Constraint Programming (CP '05)*, pages 388–402. Springer, 2005.
16. G. MacGillivray. On the complexity of colouring by vertex-transitive and arc-transistive digraphs. *SIAM J. Discret. Math.*, 4(3):397–408, 1991.
17. E. Petrank. The hardness of approximation: Gap location. *Computational Complexity*, 4:133–157, 1994.
18. R. Pöschel and L. Kalužnin. *Funktionen- und Relationenalgebren*. DVW, Berlin, 1979.
19. F. Rossi, P. van Beek, and T. Walsh, editors. *Handbook of Constraint Programming*. Elsevier, 2006.
20. L. Trevisan. Inapproximability of combinatorial optimization problems, 2004. arXiv.org:cs.CC/0409043.