Approximability of Integer Programming with Generalised Constraints*

Peter Jonsson† Fredrik Kuivinen‡ Gustav Nordh§

September 12, 2007

Abstract

We study a family of problems, called MAXIMUM SOLUTION, where the objective is to maximise a linear goal function over the feasible integer assignments to a set of variables subject to a set of constraints. When the domain is Boolean (i.e. restricted to $\{0, 1\}$), the maximum solution problem is identical to the well-studied MAX ONES problem, and the approximability is completely understood for all restrictions on the underlying constraints [Khanna et al., SIAM J. Comput., 30 (2001), pp. 1863-1920]. We continue this line of research by considering domains containing more than two elements. We present two main results: a complete classification for the approximability of all maximal constraint languages over domains of cardinality at most 4, and a complete classification of the approximability of the problem when the set of allowed constraints contains all permutation constraints. Under the assumption that a conjecture due to Szczepara holds, we give a complete classification for all maximal constraint languages. These classes of languages are well-studied in universal algebra and computer science; they have, for instance, been considered in connection with machine learning and constraint satisfaction. Our results are proved by using algebraic results from clone theory and the results indicates that this approach is very powerful for classifying the approximability of certain optimisation problems.

Keywords. combinatorial optimisation, constraint satisfaction, approximability, universal algebra

AMS: 08A70, 90C27, 68Q17, 68Q25

---

*Preliminary versions of parts of this article appeared in *Proceedings of the 12th International Conference on Principles and Practice of Constraint Programming*, Nantes, France, 2006

†Department of Computer and Information Science, Linköpings Universitet, SE-581 83 Linköping, Sweden, email: petej@ida.liu.se, phone: +46 13 282415, fax: +46 13 284499

‡Department of Computer and Information Science, Linköpings Universitet, SE-581 83 Linköping, Sweden, email: freku@ida.liu.se, phone: +46 13 286607, fax: +46 13 284499

§Department of Computer and Information Science, Linköpings Universitet, SE-581 83 Linköping, Sweden, email: gusno@ida.liu.se, phone: +46 13 282693, fax: +46 13 284499

1 Introduction

Our starting-point is the general combinatorial optimisation problem Max Ones(Γ) where Γ (known as the *constraint language*) is a finite set of finitary relations over $\{0, 1\}$. An instance of this problem consists of constraints from Γ applied to a number of Boolean variables, and the goal is to find an assignment that satisfies all constraints while maximizing the number of variables set to 1. It is easy to see that by choosing the constraint language appropriately, Max Ones(Γ) captures a number of well-known problems, for instance, Max Independent Set (problem GT23 in [2]), and certain variants of Max 0/1 Programming (problem MP2 in [2]). Many other problems are equivalent to Max Ones under different reductions: for instance, Max Set Packing (also known as Max Hypergraph Matching) and Max Ones are equivalent under Ptas-reductions [3].

The approximability (and thus the computational complexity) is known for all choices of Γ [37]. For any Boolean constraint language Γ, Max Ones(Γ) is either in **PO** or is **APX**-complete or **poly-APX**-complete or finding a solution of non-zero value is **NP**-hard or finding any solution is **NP**-hard. The exact borderlines between the different cases are given in [37]. Actually, two different problems are studied in [37]: the *weighted* problem (where each variable is assigned a non-negative weight and the objective is to find a solution of maximum total weight), and the *unweighted* problem (where each variable is assigned the weight 1). They prove that the approximability for the weighted and unweighted versions of the problem coincides.

We will study a generalization of Max Ones where variable domains are different from $\{0, 1\}$: this allows us to capture more problems than with Max Ones. For instance, this enables the study of certain problems in integer linear programming [28], problems in multiple-valued logic [36], and in equation solving over Abelian groups [38]. For larger domains, it seems significantly harder to obtain an exact characterisation of approximability than in the Boolean case. Such a characterisation would, for instance, show whether the dichotomy conjecture for constraint satisfaction problems is true or not – a famous open question which is believed to be difficult [25]. Hence, we exhibit restricted (but still fairly general) families of constraint languages where the approximability can be determined.

Let us now formally define the problem that we will study: Let $D \subset \mathbb{N}$ (*the domain*) be a finite set. The set of all n-tuples of elements from D is denoted by D^n. Any subset of D^n is called an n-ary relation on D. The set of all finitary relations over D is denoted by R_D. A *constraint language* over a finite set, D, is a finite set $\Gamma \subseteq R_D$. Constraint languages are the way in which we specify restrictions on our problems. The constraint satisfaction problem over the constraint language Γ, denoted Csp(Γ), is defined to be the decision problem with instance (V, D, C), where

- V is a set of variables,
- D is a finite set of values (sometimes called a domain), and

- C is a set of constraints $\{C_1, \ldots, C_q\}$, in which each constraint C_i is a pair (s_i, R_i) where s_i is a list of variables of length m_i, called the constraint scope, and R_i is an m_i-ary relation over the set D, belonging to Γ, called the constraint relation.

The question is whether there exists a solution to (V, D, C) or not, that is, a function from V to D such that, for each constraint in C, the image of the constraint scope is a member of the constraint relation. To exemplify this definition, let NAE be the following ternary relation on $\{0,1\}$: $NAE = \{0,1\}^3 \setminus \{(0,0,0),(1,1,1)\}$. It is easy to see that the well-known **NP**-complete problem NOT-ALL-EQUAL SAT can be expressed as CSP($\{NAE\}$).

The optimisation problem that we are going to study, WEIGHTED MAXIMUM SOLUTION(Γ) (which we abbreviate W-MAX SOL(Γ)) is defined as follows:

Instance: Tuple (V, D, C, w), where D is a finite subset of $\mathbb{N}$, (V, D, C) is a CSP(Γ) instance, and $w : V \to \mathbb{N}$ is a weight function.

Solution: An assignment $f : V \to D$ to the variables such that all constraints are satisfied.

Measure: $\sum\limits_{v \in V} w(v) \cdot f(v)$

Example 1.1 *Consider the domain $D = \{0,1\}$ and the binary relation $R = \{(0,0),(1,0),(0,1)\}$. Then,* W-MAX SOL($\{R\}$) *is exactly the weighted* MAXIMUM INDEPENDENT SET *problem.*

Although the W-MAX SOL(Γ) problem is only defined for finite constraint languages, we will, in order to simplify the presentation, sometimes deal with sets of relations which are infinite. For a (possible infinite) set of relations X we will say that W-MAX SOL(X) is tractable if W-MAX SOL(Y) is tractable for every finite subset Y of X. Here "tractable" may be containment in one of **PO**, **APX**, or **poly-APX**. Similarly, we say that W-MAX SOL(X) is hard if there is a finite subset Y of X such that W-MAX SOL(Y) is hard. Here "hard" will be one of **APX**-hard, **poly-APX**-hard or that it is **NP**-hard to find feasible solutions.

Note that our choice of measure function in the definition of W-MAX SOL(Γ) is just one of several reasonable choices. Another reasonable alternative, used in [38], would be to let the domain D be any finite set and introduce an additional function $g : D \to \mathbb{N}$ mapping elements from the domain to natural numbers. The measure could then be defined as $\sum_{v \in V} w(v) \cdot g(f(v))$. This would result in a parameterised problem W-MAX SOL(Γ, g) where the goal is to classify the complexity of W-MAX SOL(Γ, g) for all combinations of constraint languages Γ and functions g. Note that our definition of W-MAX SOL(Γ) is equivalent to the definition of W-MAX SOL(Γ, g) if in addition g is required to be injective. One of our motivations for the choice of measure function in the definition of W-MAX SOL(Γ) is to stay closer to the definition of integer programming.

Only considering finite constraint language is in many cases not very restrictive. Consider for instance integer programming over the bounded domain $\{0, \ldots, d-1\}$. Each row in the constraint matrix can be viewed as an inequality

$$a_1x_1 + a_2x_2 + \ldots + a_kx_k \geq b.$$

Obviously, such an inequality is equivalent to the following three inequalities

$$\begin{array}{c} a_1x_1 + a_2x_2 + \ldots + a_{\lfloor k/2 \rfloor}x_{\lfloor k/2 \rfloor} - z \geq 0 \\ -a_1x_1 - a_2x_2 - \ldots - a_{\lfloor k/2 \rfloor}x_{\lfloor k/2 \rfloor} + z \geq 0 \\ z + a_{\lfloor k/2 \rfloor+1} + \ldots + a_kx_k \geq b \end{array}$$

where z denotes a fresh variable that is given the weight 0 in the objective function. By repeating this process, one ends up with a set of inequalities where each inequality contains at most three variables, and the optimal solution to this instance have the same measure as the original instance. There are at most $2^d + 2^{d^2} + 2^{d^3}$ different inequalities of length ≤ 3 since the domain contains d elements, that is, we have reduced the problem to one with a finite constraint language. Finally, this reduction is polynomial-time: each inequality of length k in the original instance give rise to at most $3^{\lceil \log_2 k \rceil} = O(k^2)$ inequalities and at most $O(k^2)$ new variables.

While the approximability of W-MAX SOL is well-understood for the Boolean domain, this is not the case for larger domains. For larger domains we are aware of three results, the first one is a tight (in)approximability results for W-MAX SOL(Γ) when Γ is the set of relations that can be expressed as linear equations over $\mathbb{Z}_p$ [38]. The second result is due to Hochbaum and Naor [28] and they study integer programming with monotone constraints, i.e., every constraint is of the form $ax - by \leq c$, where x and y are variables and $a, b \in \mathbb{N}$ and $c \in \mathbb{Z}$. In our setting their result is a polynomial time algorithm for certain constraint languages. The third result is a study of the approximability of certain logically defined constraint languages [36]. The main goal of this article is to gain a better understanding of non-Boolean W-MAX SOL — for doing so, we will adapt the algebraic approach for CSPs [13, 32] for studying the approximability of W-MAX SOL.

When the algebraic approach is applicable to a certain problem, there is an equivalence relation on the constraint languages such that two constraint languages which are equivalent under this relation have the same complexity. More specifically, two constraint languages are in the same equivalence class if they generate the same *relational clone*. The relational clone generated by Γ, captures the expressive power of Γ and is denoted by $\langle \Gamma \rangle$. Hence, instead of studying every possible finite set of relations it is enough to study the relational clones. Thus, given two constraint languages Γ_1 and Γ_2 such that $\langle \Gamma_1 \rangle = \langle \Gamma_2 \rangle$ then, W-MAX SOL(Γ_1) and W-MAX SOL(Γ_2) are equivalent under polynomial-time reductions.

The clone-theoretic approach for studying the complexity of CSPs has been very successful: it has, for instance, made it possible to design new efficient algorithms and to clarify the borderline between tractability and intractability in many important cases. In particular the complexity of the CSP problem over three element domains is now completely understood [10]. In addition to the CSP problem it is possible to use those tools from universal algebra to prove complexity results in many other CSP-like problems, for example the complexity of the quantified constraint satisfaction problem, where variables can not only be existentially quantified but also universally quantified, has successfully been attacked with the clone-theoretic approach [7, 17]. Furthermore, the #CSP problem [11] (where the number of solutions to a CSP is counted) have also benefitted from this approach. However, it seems that this technique cannot be used for some other CSP-like problems: notable exceptions are MAX CSP [34] and the problem of enumerating all solutions to a CSP instance [46]. For some problems it is the case that the relational clones are a useable tool in the boolean domain but not in larger domains. The enumeration problem is one such case [46].

We begin by proving that the algebraic approach is applicable to W-MAX SOL and this result can be found in Theorem 3.3*. In fact, we show that given two constraint languages Γ_1 and Γ_2 such that $\langle\Gamma_1\rangle = \langle\Gamma_2\rangle$, then W-MAX SOL($\Gamma_1$) *S*-reduces to W-MAX SOL(Γ_2), and vice-versa. An *S-reduction* is a certain strong approximation-preserving reduction: if $\langle\Gamma_1\rangle = \langle\Gamma_2\rangle$, then Γ_1 and Γ_2 are very similar with respect to approximability. For instance, if W-MAX SOL(Γ_1) is **NP**-hard to approximate within some constant c, then W-MAX SOL(Γ_2) is **NP**-hard to approximate within c, too. The proof is accompanied by an example of how the approach can be used "in practice" for proving approximability results. We note that the clone-theoretic approach was not used in the original classification of MAX ONES and, consequently, the techniques we use differs substantially from those used in [37]. The results that we prove with the aid of Theorem 3.3 are the following:

Result 1. Our first result concerns the complexity of W-MAX SOL for *maximal* constraint languages. A constraint language Γ is maximal if, for any $R \notin \langle\Gamma\rangle$, $\Gamma \cup \{R\}$ has the ability to express (in a sense to be formally defined later on) every relation in R_D. Such languages have attracted much attention lately: for instance, the complexity of the corresponding CSP problems has been completely classified. In [14] the complexity was classified for domains $|D| \leq 3$ and necessary conditions for tractability was proved for the general case. More recently, in [8], it was proved that those necessary conditions also are sufficient for tractability. Maximal constraint languages have also been studied in the context of machine learning [24] and quantified CSPs [18], and they attract a great deal of attention in universal algebra, cf. the survey by Quackenbush [44].

Our results shows that if Γ is maximal and $|D| \leq 4$, then W-MAX SOL(Γ) is either tractable, **APX**-complete, **poly-APX**-complete, that finding any solution with non-zero measure is **NP**-hard, or that CSP(Γ) is not tractable. Moreover, we prove that under a conjecture by Szczepara [47] our classification of maximal constraint languages extends to arbitrary finite domains. (In the

*The proof is easy to adapt to other problems such as W-MIN SOL (the minimization version of W-MAX SOL) and AW-MAX SOL (where both positive and negative weights are allowed).

conference version [35] of this article we claimed that we had characterised the complexity of MAX SOL for all maximal constraint languages. Unfortunately, there was a flaw in one of the proofs. We have managed to repair some of it by proving the weaker results as stated above, but the general case, when $|D| > 4$ and Szczepara's conjecture is not assumed to hold, remains open.) We also note that the different cases can be efficiently recognised, i.e. the approximability of a maximal constraint language Γ can be decided in polynomial time (in the size of Γ).

When proving this result, we identified a new large tractable class of W-MAX SOL(Γ): *generalised max-closed* constraints. This class (which may be of independent interest) significantly extend some of the tractable classes of MAX ONES that were identified by Khanna et al. [37]. It is also related to *monotone* constraints which have been studied in mathematical programming and computer science [27, 28, 51]. In fact, generalised max-closed constraints generalise monotone constraints over finite domains. A certain kind of generalised max-closed constraints are relevant in constraint programming languages such as CHIP [50] as is pointed out in [33]. It may thus be possible to extend such languages with optimisation capabilities by using the techniques presented in this article.

Result 2. We completely characterise the approximability of W-MAX SOL(Γ) when Γ contains all permutation constraints. Such languages are known as *homogeneous* languages and Dalmau [23] has determined the complexity of CSP(Γ) for all such languages while the complexity of the corresponding quantified CSPs has been studied by Börner et al. [6]. Szendrei [48] provides a compact presentation of algebraic results on homogeneous algebras and languages.

We show that W-MAX SOL(Γ) is either tractable, **APX**-complete, **poly-APX**-complete, or that CSP(Γ) is not tractable. The four different cases can, just as in **Result 1**, be efficiently recognised. The proof is based on the characterisation of homogenous algebras by Marczewski [40] and Marchenkov [39]. For each domain D, there exists a set of relations Q_D such that every tractable homogenous constraint language on D is a subset of Q_D. The relations in Q_D are invariant under a certain operation $t : D^3 \to D$ (known as the *discriminator* on D) and the algebra $(D; t)$ is an example of a *quasi-primal* algebra in the sense of Pixley [41]. We note that the tractable homogenous constraint languages have been considered earlier in connection with *soft constraints* [21], i.e. constraints which allows different levels of 'desirability' to be associated with different value assignments [5]. In the terminology of [21], these languages are invariant under a $\langle \mathsf{Mjrty}_1, \mathsf{Mjrty}_2, \mathsf{Mnrty}_3 \rangle$ multimorphism. We also note that the tractable homogeneous languages extend the *width-2 affine* class of MAX ONES that were identified by Khanna et al. [37].

We remark that we do not deal explicitly with the unweighted version of the problem (denoted MAX SOL(Γ)), where all variables have weight 1. The reason for this is that the approximability classifications for MAX SOL(Γ) can be deduced from the classifications for W-MAX SOL(Γ) (for all constraint languages Γ considered in this paper). In fact, as we explain in Section 8, MAX SOL(Γ)

have the same approximability as W-MAX SOL(Γ) when Γ one of the constraint languages considered in this article.

The article is structured as follows: We begin by presenting some basics on approximability in §2. The algebraic approach for studying W-MAX SOL is presented in §3, §4 identifies certain hard constraint languages, and §5 contains some tractability results. We continue with §6 that contain **Result 1** and §7 that contain **Result 2**. Finally, §8 contains some final remarks.

2 Approximability, Reductions, and Completeness

A *combinatorial optimisation problem* is defined over a set of *instances* (admissible input data); each instance I has a finite set $\mathsf{sol}(I)$ of *feasible solutions* associated with it. Given an instance I and a feasible solution s of I, $m(I, s)$ denotes the positive integer *measure* of s. The objective is, given an instance I, to find a feasible solution of *optimum* value with respect to the measure m. 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 **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., [2]).

We say that a solution $s \in \mathsf{sol}(I)$ to an instance I of an **NPO** problem Π is *r-approximate* if it is satisfying

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

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

We define **PO** to be 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. Similarly, Π is in the class **poly-APX** if there is a polynomial-time approximation algorithm for Π whose performance ratio is bounded by a polynomial in the size of the input. Completeness in **APX** and **poly-APX** is defined using appropriate reductions, called AP-reductions and A-reductions respectively [22, 37]. AP-reductions are more sensitive than A-reductions and every AP-reduction is also an A-reduction [37]. In this paper we will not need the added flexibility of A-reductions for proving our **poly-APX**-completeness results. Hence, we only need the definition of AP-reductions.

Definition 2.1 *An* **NPO** *problem* Π_1 *is said to be* AP-reducible *to an* **NPO** *problem* Π_2 *if two polynomial-time computable functions* F *and* G *and a constant* α *exist such that*

(a) *for any instance I of Π_1, $F(I)$ is an instance of Π_2;*

(b) *for any instance I of Π_1, and any feasible solution s' of $F(I)$, $G(I, s')$ is a feasible solution of I;*

(c) *for any instance I of Π_1, and any $r \geq 1$, if s' is an r-approximate solution of $F(I)$ then $G(I, s')$ is an $(1 + (r-1)\alpha + o(1))$-approximate solution of I where the o-notation is with respect to $|I|$.*

An **NPO** *problem Π is* **APX**-hard *(***poly-APX**-hard*) if every problem in* **APX** *(***poly-APX***) is AP-reducible (A-reducible) to it. If, in addition, Π is in* **APX** *(***poly-APX***), then Π is called* **APX**-complete *(***poly-APX**-complete*).*

It is a well-known fact (see, e.g., §8.2.1 in [2]) that AP-reductions compose. In some proofs we will use another kind of reduction, S-reductions. They are defined as follows:

Definition 2.2 *An* **NPO** *problem Π_1 is said to be S-*reducible *to an* **NPO** *problem Π_2 if two polynomial-time computable functions F and G exist such that*

(a) *given any instance I of Π_1, algorithm F produces an instance $I' = F(I)$ of Π_2, such that the measure of an optimal solution for I', $\text{OPT}(I')$, is exactly $\text{OPT}(I)$.*

(b) *given $I' = F(I)$, and any solution s' to I', algorithm G produces a solution s to I such that $m_1(I, G(s')) = m_2(I', s')$, where m_1 is the measure for Π_1 and m_2 is the measure for Π_2.*

Obviously, the existence of an S-reduction from Π_1 to Π_2 imply the existence of an AP-reduction from Π_1 to Π_2. The reason why we need S-reductions is that AP-reductions do not (generally) preserve membership in **PO** [37]. We also note that S-reduction preserve approximation thresholds exactly for problems in **APX**: let Π_1, Π_2 be problems in **APX**, assume that it is **NP**-hard to approximate Π_1 within c, and that there exists an S-reduction from Π_1 to Π_2. Then, it is **NP**-hard to approximate Π_2 within c, too.

In some of our hardness proofs, it will be convenient for us to use a type of approximation-preserving reduction called L-reduction [2].

Definition 2.3 *An* **NPO** *problem Π_1 is said to be L-*reducible *to an* **NPO** *problem Π_2 if two polynomial-time computable functions F and G and positive constants β and γ exist such that*

(a) *given any instance I of Π_1, algorithm F produces an instance $I' = F(I)$ of Π_2, such that the measure of an optimal solution for I', $\text{OPT}(I')$, is at most $\beta \cdot \text{OPT}(I)$;*

(b) *given $I' = F(I)$, and any solution s' to I', algorithm G produces a solution s to I such that $|m_1(I, s) - \text{OPT}(I)| \leq \gamma \cdot |m_2(I', s') - \text{OPT}(I')|$, where m_1 is the measure for Π_1 and m_2 is the measure for Π_2.*

It is well-known (see, e.g., Lemma 8.2 in [2]) that, if Π_1 is L-reducible to Π_2 and $\Pi_1 \in$ **APX** then there is an AP-reduction from Π_1 to Π_2.

3 Algebraic Approach

We sometimes need to define relations in terms of other relations, using certain logical formulas. In these definitions we use the standard correspondence between constraints and relations: a relation consists of all tuples of values satisfying the corresponding constraint. Although, we sometimes use the same symbol for a constraint and its corresponding relation, the meaning will always be clear from the context. More specifically, for a relation R with arity a we will sometimes write $R(x_1, \ldots, x_a)$ with the meaning $(x_1, \ldots, x_a) \in R$ and the constraint $((x_1, \ldots, x_a), R)$ will sometimes be written as $R(x_1, \ldots, x_a)$.

An *operation* on a finite set D (the domain) is an arbitrary function $f : D^k \rightarrow D$. 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 and let R be an n-ary relation over D. For any collection of k tuples, $\boldsymbol{t_1}, \boldsymbol{t_2}, \ldots, \boldsymbol{t_k} \in R$, the n-tuple $f(\boldsymbol{t_1}, \boldsymbol{t_2}, \ldots, \boldsymbol{t_k})$ is defined as follows: $f(\boldsymbol{t_1}, \boldsymbol{t_2}, \ldots, \boldsymbol{t_k}) = (f(\boldsymbol{t_1}[1], \boldsymbol{t_2}[1], \ldots, \boldsymbol{t_k}[1]), f(\boldsymbol{t_1}[2], \boldsymbol{t_2}[2], \ldots, \boldsymbol{t_k}[2]), \ldots, f(\boldsymbol{t_1}[n], \boldsymbol{t_2}[n], \ldots, \boldsymbol{t_k}[n]))$, where $\boldsymbol{t_j}[i]$ is the i-th component in tuple $\boldsymbol{t_j}$. A technique that has shown to be useful in determining the computational complexity of $\mathrm{CSP}(\Gamma)$ is that of investigating whether the constraint language Γ is invariant under certain families of operations [32].

Now, let $R_i \in \Gamma$. If f is an operation such that for all $\boldsymbol{t_1}, \boldsymbol{t_2}, \ldots, \boldsymbol{t_k} \in R_i$ $f(\boldsymbol{t_1}, \boldsymbol{t_2}, \ldots, \boldsymbol{t_k}) \in R_i$, then R_i is *invariant* (or, in other words, closed) under f. If all constraint relations in Γ are invariant under f then Γ is 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 $Pol(\Gamma)$. Given a set of operations F, the set of all relations that are invariant under all the operations in F is denoted $Inv(F)$. Whenever there is only one operation under consideration, we write $Inv(f)$ instead of $Inv(\{f\})$.

We will need a number of operations in the sequel: an operation f over D is said to be

- a *constant* operation if f is unary and $f(a) = c$ for all $a \in D$ and some $c \in D$;
- a *majority* operation if f is ternary and $f(a, a, b) = f(a, b, a) = f(b, a, a) = a$ for all $a, b \in D$;
- a *binary commutative idempotent* operation if f is binary, $f(a, a) = a$ for all $a \in D$, and $f(a, b) = f(b, a)$ for all $a, b \in D$;
- an *affine* operation if f is ternary and $f(a, b, c) = a - b + c$ for all $a, b, c \in D$ where $+$ and $-$ are the binary operations of an Abelian group $(D, +, -)$.

Example 3.1 *Let $D = \{0, 1, 2\}$ and let f be the majority operation on D where $f(a, b, c) = a$ if a, b and c are all distinct. Furthermore, let*

$$R = \{(0, 0, 1), (1, 0, 0), (2, 1, 1), (2, 0, 1), (1, 0, 1)\}.$$

It is then easy to verify that for every triple of tuples, $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in R$, *we have* $f(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \in R$. *For example, if* $\boldsymbol{x} = (0,0,1), \boldsymbol{y} = (2,1,1)$ *and* $\boldsymbol{z} = (1,0,1)$ *then*

$$f(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = \Big(f(\boldsymbol{x}[1], \boldsymbol{y}[1], \boldsymbol{z}[1]), f(\boldsymbol{x}[2], \boldsymbol{y}[2], \boldsymbol{z}[2]), f(\boldsymbol{x}[3], \boldsymbol{y}[3], \boldsymbol{z}[3]) \Big) =$$
$$\big(f(0,2,1), f(0,1,0), f(1,1,1) \big) = (0,0,1) \in R.$$

We can conclude that R *is invariant under* f *or, equivalently, that* f *is a polymorphism of* R.

We continue by defining a closure operation $\langle \cdot \rangle$ 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 $\Gamma \cup \{=_D\}$ ($=_D$ is the equality relation on D), conjunction, and existential quantification. Intuitively, constraints using relations from $\langle \Gamma \rangle$ are exactly those which can be simulated by constraints using relations from Γ. The sets of relations of the form $\langle \Gamma \rangle$ are referred to as *relational clones*. An alternative characterisation of relational clones is given in the following theorem.

Theorem 3.2 ([43]) *For every set* $\Gamma \subseteq R_D$, $\langle \Gamma \rangle = Inv(Pol(\Gamma))$.

The following theorem states that when we are studying the approximability of W-Max Sol(Γ), it is sufficient to consider constraint languages that are relational clones.

Theorem 3.3 *Let* Γ *be a constraint language and* $\Gamma' \subseteq \langle \Gamma \rangle$ *finite. Then* W-Max Sol(Γ') *is S-reducible to* W-Max Sol(Γ).

Proof. Consider an instance $I = (V, D, C, w)$ of W-Max Sol(Γ'). We transform I into an instance $F(I) = (V', D, C', w')$ of W-Max Sol(Γ).

For every constraint $C = ((v_1, \ldots, v_m), R)$ in I, R can be represented as

$$\exists_{v_{m+1}}, \ldots, \exists_{v_n} R_1(v_{11}, \ldots, v_{1n_1}) \wedge \cdots \wedge R_k(v_{k1}, \ldots, v_{kn_k})$$

where $R_1, \ldots, R_k \in \Gamma \cup \{=_D\}$, $v_{m+1}, \ldots, v_n$ are fresh variables, and $v_{11}, \ldots, v_{1n_1}$, $v_{21}, \ldots, v_{kn_k} \in \{v_1, \ldots, v_n\}$. Replace the constraint C with the constraints

$$((v_{11} \ldots, v_{1n_1}), R_1), \ldots, ((v_{k1}, \ldots, v_{kn_k}), R_k),$$

add $v_{m+1}, \ldots, v_n$ to V, and extend w so that $v_{m+1}, \ldots v_n$ are given weight 0. If we repeat the same reduction for every constraint in C, then it results in an equivalent instance of W-Max Sol($\Gamma_1 \cup \{=_D\}$).

For each equality constraint $((v_i, v_j), =_D)$, we do the following:

- replace all occurrences of v_j with v_i, update w' so that the weight of v_j is added to the weight of v_i, remove v_j from V, and remove the weight corresponding to v_j from w'; and
- remove $((v_i, v_j), =_D)$ from C.

The resulting instance $F(I) = (V', D, C', w')$ of W-MAX SOL(Γ) has the same optimum as I (i.e., $\text{OPT}(I) = \text{OPT}(F(I))$) and has been obtained in polynomial time.

Now, given a feasible solution S' for $F(I)$, let $G(I, S')$ be the feasible solution for I where:

- The variables in I assigned by S' inherit their value from S'.
- The variables in I which are still unassigned all occur in equality constraints and their values can be found by simply propagating the values of the variables which have already been assigned.

It should be clear that $m(I, G(I, S')) = m(F(I), S')$ for any feasible solution S' for $F(I)$. Hence, the functions F and G, as described above, is an S-reduction from W-MAX SOL(Γ') to W-MAX SOL(Γ).

To exemplify the use of the results in this section, we prove the following tight approximability result:

Lemma 3.4 *Let Γ be a finite constraint language over the domain $\{0, 1\}$. If* MAX ONES(Γ) *is in* **APX** *and not in* **PO**, *then there is a polynomial time approximation algorithm for* MAX ONES(Γ) *with performance ratio* 2, *and it is* **NP**-*hard to approximate* MAX ONES(Γ) *within* $2 - \epsilon$, *for any* $\epsilon > 0$.

Proof (Sketch). It follows from the classification results in [37] that if MAX ONES(Γ) is in **APX** and not in **PO**, then Γ is closed under the affine function $f(x, y, z) = x - y + z \pmod 2$. It also follows from [37, Lemma 6.6] that MAX ONES(Γ) is approximable within 2.

In the Boolean domain, the structure of all relational clones is known. This classification was made by Emil Post in [42] and is often referred to as Post's lattice. A nice introduction to boolean relations and Post's lattice can be found in [15, 16].

By Theorem 3.3, it is enough to study the relational clones. Studying Post's lattice and the results for MAX ONES in [37] one can conclude that there are three relational clones which are interesting in our case (i.e., there are three relational clones where MAX ONES(Γ) is in **APX** but not in **PO**). Those relational clones are called IL_0, IL_2 and IL_3 and can be defined as follows [16]:

$$\begin{aligned} IL_0 &= \{x_1 + \cdots + x_k = 0 \pmod 2 \mid k \in \mathbb{N}\} \\ IL_2 &= \{x_1 + \cdots + x_k = c \pmod 2 \mid k \in \mathbb{N},\ c \in \{0, 1\}\} \\ IL_3 &= \{x_1 + \cdots + x_k = c \pmod 2 \mid k \text{ even},\ c \in \{0, 1\}\} \end{aligned}$$

We get the following inclusions from Post's lattice: $IL_0 \subset IL_2$ and $IL_3 \subset IL_2$.

It is proved in [38] that for a certain finite subset Γ of IL_3, MAX ONES(Γ) is **NP**-hard to approximate within $2 - \epsilon$ for all $\epsilon > 0$. As $IL_3 \subset IL_2$ we get that MAX ONES(Γ) is **NP**-hard to approximate within $2 - \epsilon$ for all $\epsilon > 0$ if $\langle \Gamma \rangle = IL_2$.

What remains to be done is to prove **NP**-hardness for approximating MAX ONES(Γ) within $2 - \epsilon$ if $\langle \Gamma \rangle = IL_0$. We do this with a reduction from MAX-E3-LIN-2 which is the following problem: given a set of equations over $\mathbb{Z}_2$ with exactly three variables per equation, satisfy as many equations as possible. It is proved in [29] that it is **NP**-hard to approximate MAX-E3-LIN-2 within $2 - \epsilon$ for any $\epsilon > 0$.

Let I be an instance of MAX-E3-LIN-2. We will construct an instance I' of MAX ONES(Γ) for a subset Γ of IL_0. Given an equation $x_1 + x_2 + x_3 = 1 \pmod 2$ in I (we can assume that all equations have 1 on the right hand side [29]), we add the equation $x_1 + x_2 + x_3 = z$ (where z is a fresh variable that only occurs in one equation) to I'. Furthermore, we assign the weight zero to x_1, x_2 and x_3 and the weight one to z. It is not hard to see that a solution with measure m to I can easily be transformed into a solution with measure m for I'. It is also the case that a solution of measure m for I' can be seen as a solution with measure m for I.

4 Hardness and Membership Results

In this section, we first prove some general and easy **APX** and **poly-APX** membership results for W-MAX SOL(Γ). We also prove **APX**-completeness and **poly-APX**-completeness for some particular constraint languages. Most of our hardness results in subsequent sections are based on these results.

We begin by making the following easy but interesting observation: we know from the classification of W-MAX SOL(Γ) over the Boolean domain $\{0, 1\}$ that there exist many constraint languages Γ for which W-MAX SOL(Γ) is **poly-APX**-complete. However, if 0 is not in the domain, then there are no constraint languages Γ such that W-MAX SOL(Γ) is **poly-APX**-complete.

Proposition 4.1 *If* CSP(Γ) *is in* **P** *and* $0 \notin D$*, then* W-MAX SOL(Γ) *is in* **APX**.

Proof. It is proved in [19] that if CSP(Γ) is in **P**, then we can also find a solution in polynomial time. It should be clear that this solution is a $\frac{\max(D)}{\min(D)}$-approximate solution. Hence, we have a trivial approximation algorithm with performance ratio $\frac{\max(D)}{\min(D)}$.

Next, we present a general membership result for W-MAX SOL(Γ). The proof is similar to the proof of the corresponding result for the Boolean domain in [37, Lemma 6.2] so we omit the proof.

Lemma 4.2 *Let* $\Gamma^c = \{\Gamma \cup \{\{(d_1)\}, \ldots, \{(d_n)\}\}$*, where* $D = \{d_1, \ldots, d_n\}$ *(i.e.,* Γ^c *is the constraint language corresponding to* Γ *where we can force variables to take any given value in the domain). If* CSP(Γ^c) *is in* **P***, then* W-MAX SOL(Γ) *is in* **poly-APX**.

We continue by proving the **APX**-completeness of some constraint languages.

Lemma 4.3 *Let* $R = \{(a,a),(a,b),(b,a)\}$ *and* $a, b \in D$ *such that* $0 < a < b$. *Then,* W-MAX SOL($\{R\}$) *is* **APX**-*complete.*

Proof. We give an L-reduction (with parameters $\beta = 4b$ and $\gamma = \frac{1}{b-a}$) from the **APX**-complete problem INDEPENDENT SET restricted to degree 3 graphs [1] to MAX SOL($\{R\}$). Given an instance $I = (V, E)$ of INDEPENDENT SET (restricted to graphs of degree at most 3 and containing no isolated vertices), let $F(I) = (V, D, C)$ be the instance of MAX SOL($\{R\}$) where, for each edge $(v_i, v_j) \in E$, we add the constraint $R(x_i, x_j)$ to C. For any feasible solution S' for $F(I)$, let $G(I, S')$ be the solution for I where all vertices corresponding to variables assigned b in S' form the independent set. We have $|V|/4 \leq \text{OPT}(I)$ and $\text{OPT}(F(I)) \leq b|V|$ so $\text{OPT}(F(I)) \leq 4b\text{OPT}(I)$. Thus, $\beta = 4b$ is an appropriate parameter.

Let K be the number of variables being set to b in an arbitrary solution S' for $F(I)$. Then,

$$|\text{OPT}(I) - m(I, G(I, S'))| = \text{OPT}(I) - K \quad \text{and}$$
$$|\text{OPT}(F(I)) - m(F(I), S')| = (b-a)(\text{OPT}(I) - K).$$

Hence,

$$|\text{OPT}(I) - m(I, G(I, S')| = \frac{1}{b-a}|\text{OPT}(F(I)) - m(F(I), S')|$$

and $\gamma = \frac{1}{b-a}$ is an appropriate parameter.

The generic **poly-APX**-complete constraint languages are presented in the following lemma.

Lemma 4.4 *Let* $R = \{(0,0),(0,b),(b,0)\}$ *and* $b \in D$ *such that* $0 < b$. *Then,* W-MAX SOL($\{R\}$) *is* **poly-APX**-*complete.*

Proof. It is proved in [37, Lemma 6.15] that for $Q = \{(0,0),(0,1),(1,0)\}$, it is the case that W-MAX SOL($\{Q\}$) is **poly-APX**-complete. To prove the **poly-APX**-hardness we give an AP-reduction from W-MAX SOL($\{Q\}$) to W-MAX SOL($\{R\}$). Given an instance I of W-MAX SOL($\{Q\}$), let $F(I)$ be the instance of W-MAX SOL($\{R\}$) where all occurrences of Q has been replaced by R. For any feasible solution S' for $F(I)$, let $G(I, S')$ be the solution for I where all variables assigned b in S' are instead assigned 1. It should be clear that this is an AP-reduction, since if S' is an α-approximate solution to $F(I)$, then $G(I, S')$ is an α-approximate solution for I.

To see that W-MAX SOL($\{R\}$) is in **poly-APX**, let $D = \{d_1, \ldots, d_n\}$ and note that $\Gamma^c = \{R, \{(d_1)\}, \ldots, \{(d_n)\}\}$ is invariant under the min function. As the min function is associative, commutative and idempotent, CSP(Γ^c) is solvable in polynomial time [32]. Hence, W-MAX SOL($\{R\}$) is in **poly-APX** due to Lemma 4.2

5 Tractable Constraint Languages

In this section, we present tractability results for two classes of constraint languages: *injective* constraint languages and *generalised max-closed* constraint languages. The tractability of injective constraints follows from Cohen *et al.* [21, Sec. 4.4] but we present a simple proof for increased readability. The tractability result for generalised max-closed constraints is new and its proof constitutes the main part of this section.

These two classes can be seen as substantial and nontrivial generalisations of the tractable classes known for the corresponding (Weighted) Max Ones problem over the Boolean domain. There are only three tractable classes of constraint languages over the Boolean domain, namely width-2 affine, 1-valid, and weakly positive [37]. Width-2 affine constraint languages are examples of injective constraint languages and the classes of 1-valid and weakly positive constraint languages are examples of generalised max-closed constraint languages. The monotone constraints which are, for instance, studied by Hochbaum et al. [27, 28] (in relation with integer programming) and Woeginger [51] (in relation with constraint satisfaction) are also related to generalised max-closed constraints. Hochbaum & Naor [28] show that monotone constraints can be characterised as those constraints that are simultaneously invariant under the max and min operators. Hence, monotone constraints are also generalised max-closed constraints as long as the underlying domain is finite.

5.1 Injective relations

We begin by formally defining *injective relations*.

Definition 5.1 *A relation, $R \in R_D$, is called* injective *if there exists a subset $D' \subseteq D$ and an injective function $\pi : D' \to D$ such that*

$$R = \{(x, \pi(x)) \mid x \in D'\}.$$

It is important to note that the function π is *not* assumed to be total on D. Let I^D denote the set of all injective relations on the domain D and let $\Gamma_I^D = \langle I^D \rangle$.

Example 5.2 *Let $D = \{0, 1\}$ and let $R = \{(x, y) \mid x, y \in D,\ x + y \equiv 1 \pmod 2\}$. The relation R is injective because the function $f : D \to D$ defined as $f(0) = 1$ and $f(1) = 0$ is injective. More generally, let $G = (D', +, -)$ be an arbitrary Abelian group and let $c \in D'$ be an arbitrary group element. It is easy to see that the relation $\{(x, y) \mid x, y \in D',\ x + y = c\}$ is injective.*

R is an example of a relation which is invariant under an affine operation. Such relations have previously been studied in relation with the Max Ones problem in [37, 38]. We will give some additional results for such constraints in §6.3. With the terminology used in [37, 38], R is said to be *width-2 affine.* The relations which can be expressed as the set of solutions to an equation with two variables over an Abelian group are exactly the width-2 affine relations, so the injective relations are a superset of the width-2 affine relations.

To see that W-MAX SOL(Γ) is in **PO** for every finite constraint language $\Gamma \subseteq \langle I^D \rangle$, it is sufficient to prove that W-MAX SOL(I^D) is in **PO** by Theorem 3.3. Given an instance of W-MAX SOL(I^D), consider the graph having the variables as vertices and edges between the vertices/variables occurring together in the same constraint. Each connected component of this graph represents an independent subproblem that can be solved separately. If a value is assigned to a variable/vertex, all variables/vertices in the same component will be forced to take a value by propagating this assignment. Hence, each connected component has at most $|D|$ different solutions (that can be easily enumerated) and an optimal one can be found in polynomial time.

5.2 Generalised Max-Closed Relations

We begin by giving the basic definition:

Definition 5.3 *A constraint language* Γ *over a domain* $D \subset \mathbb{N}$ *is* generalised max-closed *if and only if there exists a binary operation* $f \in Pol(\Gamma)$ *such that for all* $a, b \in D$*,*

1. *if* $a \neq b$ *and* $f(a,b) \leq \min(a,b)$*, then* $f(b,a) > \max(a,b)$*; and*
2. $f(a,a) \geq a$*.*

In the conference version of this article [35] the definition of generalised max-closed constraint languages was slightly more restrictive. The following two examples will clarify the definition above.

Example 5.4 *Assume that the domain* D *is* $\{0,1,2,3\}$*. As an example of a generalised max-closed relation consider*

$$R = \{(0,0),(1,0),(0,2),(1,2)\}.$$

R *is invariant under* max *and is therefore generalised max-closed since* max *satisfies the properties of Definition 5.3. Now, consider the relation* Q *defined as*

$$Q = \{(0,1),(1,0),(2,1),(2,2),(2,3)\}.$$

Q *is not invariant under* max *because*

$$\max((0,1),(1,0)) = (\max(0,1), \max(1,0)) = (1,1) \notin Q.$$

Let the operation $\circ : D^2 \rightarrow D$ *be defined by the following Cayley table:*†

$\circ$	0	1	2	3
0	0	2	2	3
1	2	1	2	2
2	2	2	2	3
3	3	2	3	3

†Note that we write $x \circ y$ instead of $\circ(x,y)$.

Now, it is easy to verify that $Inv(\circ)$ *is a set of generalised max-closed relations and that* $Q \in Inv(\circ)$.

Example 5.5 *Consider the relations* R_1 *and* R_2 *defined as,*

$$R_1 = \{(1,1,1),(1,0,0),(0,0,1),(1,0,1)\}$$

and $R_2 = R_1 \setminus \{(1,1,1)\}$. *The relation* R_1 *is* 1*-valid because the tuple consisting only of ones is in* R_1, *i.e.,* $(1,1,1) \in R_1$. R_2, *on the other hand, is not* 1*-valid but is weakly positive*[‡] *because it is invariant under* max. *Note that both* R_1 *and* R_2 *are generalised max-closed since* R_1 *is invariant under* $f(x,y) = 1$ *and* R_2 *is invariant under* $f(x,y) = \max(x,y)$. *It is in fact the case that every weakly positive relation is invariant under* max, *so the* 1*-valid and weakly positive relations are subsets of the generalised max-closed relations.*

The tractability of generalised max-closed constraint languages crucially depends on the following lemma.

Lemma 5.6 *If* Γ *is generalised max-closed, then all relations*

$$R = \{(d_{11}, d_{12}, \ldots, d_{1m}), \ldots, (d_{t1}, d_{t2}, \ldots, d_{tm})\}$$

in Γ *have the property that the tuple*

$$\boldsymbol{t}_{\mathbf{max}} = (\max\{d_{11}, \ldots, d_{t1}\}, \ldots, \max\{d_{1m}, \ldots, d_{tm}\})$$

is in R, *too.*

Proof. Assume that there is a relation R in Γ such that the tuple

$$\boldsymbol{t}_{\mathbf{max}} = (\max\{d_{11}, \ldots, d_{t1}\}, \ldots, \max\{d_{1m}, \ldots, d_{tm}\})$$

is not in R. Define the distance between two tuples to be the number of coordinates where they disagree (i.e. the Hamming distance). Let $\boldsymbol{a}$ be a tuple in R with minimal distance from $\boldsymbol{t}_{\mathbf{max}}$ and let I denote the set of coordinates where $\boldsymbol{a}$ agrees with $\boldsymbol{t}_{\mathbf{max}}$. By the assumption that $\boldsymbol{t}_{\mathbf{max}}$ is not in R, we know that the distance between $\boldsymbol{a}$ and $\boldsymbol{t}_{\mathbf{max}}$ is at least 1. Hence, without loss of generality, assume that $\boldsymbol{a}[1] \neq \boldsymbol{t}_{\mathbf{max}}[1]$ and that $\boldsymbol{a}[1]$ is maximal for all tuples in R agreeing with $\boldsymbol{t}_{\mathbf{max}}$ on the coordinates in I. Let $\boldsymbol{b}$ be a tuple in R such that $\boldsymbol{b}[1] = \boldsymbol{t}_{\mathbf{max}}[1]$.

Since Γ is generalised max-closed, there exists an operation $f \in Pol(\Gamma)$ such that for all $a, b \in D$ $(a \neq b)$, it holds that $f(a,b) > \max(a,b)$ whenever $f(b,a) \leq \min(a,b)$. Furthermore, for all $a \in D$ it holds that $f(a,a) \geq a$. Now consider the tuple $\boldsymbol{x^n}$ $(n = |D|)$ defined as follows: $\boldsymbol{x^1} = f(\boldsymbol{a}, \boldsymbol{b})$, and

$$\boldsymbol{x^{i+1}} = \begin{cases} f(\boldsymbol{x^i}, \boldsymbol{a}) & \text{if } f(\boldsymbol{x^i}[1], \boldsymbol{a}[1]) > \boldsymbol{a}[1], \\ f(\boldsymbol{a}, \boldsymbol{x^i}) & \text{otherwise.} \end{cases}$$

[‡]A relation is weakly positive if it can be expressed as a CNF formula having at most one negated variable in each clause.

We begin by proving that $\boldsymbol{x^n}$ agrees with $\boldsymbol{a}$ on all coordinates in I. Let $\boldsymbol{z}$ be an arbitrary tuple in R. Note that for each $i \in I$ such that $\boldsymbol{z}[i] \neq \boldsymbol{a}[i]$, it is the case that $f(\boldsymbol{a}[i], \boldsymbol{z}[i]) \leq \min(\boldsymbol{a}[i], \boldsymbol{z}[i])$ implies that $f(\boldsymbol{z}[i], \boldsymbol{a}[i]) > \max(\boldsymbol{a}[i], \boldsymbol{z}[i])$. Hence, as $\boldsymbol{a}[i] = \boldsymbol{t_{\max}}[i]$, we cannot have that $f(\boldsymbol{a}[i], \boldsymbol{z}[i]) \leq \min(\boldsymbol{a}[i], \boldsymbol{z}[i])$. So, for each $\boldsymbol{z} \in R$ and $i \in I$, we must have $f(\boldsymbol{a}[i], \boldsymbol{z}[i]) > \min(\boldsymbol{a}[i], \boldsymbol{z}[i])$ whenever $\boldsymbol{a}[i] \neq \boldsymbol{z}[i]$. By an analogous argument, it follows that for each $\boldsymbol{z} \in R$ and $i \in I$ we must have $f(\boldsymbol{z}[i], \boldsymbol{a}[i]) > \min(\boldsymbol{a}[i], \boldsymbol{z}[i])$ whenever $\boldsymbol{a}[i] \neq \boldsymbol{z}[i]$.

This together with the fact that $f(d, d) \geq d$, for all $d \in D$, and that $\boldsymbol{a}$ agrees with $\boldsymbol{t_{\max}}$ on I implies that $f(\boldsymbol{a}, \boldsymbol{x^n})$ agrees with $\boldsymbol{a}$ on I.

We now show that $\boldsymbol{x^n}[1] > \boldsymbol{a}[1]$. This follows from essentially the same argument as above. First note that $f(\boldsymbol{a}[1], \boldsymbol{b}[1]) = \boldsymbol{x^1}[1] > \boldsymbol{a}[1]$. If $f(\boldsymbol{a}[1], \boldsymbol{b}[1]) \leq \min(\boldsymbol{a}[1], \boldsymbol{b}[1])$, then $f(\boldsymbol{b}[1], \boldsymbol{a}[1]) > \boldsymbol{b}[1]$ which is not possible since $\boldsymbol{b}[1] = \boldsymbol{t_{\max}}[1]$. Hence, we must have $f(\boldsymbol{a}[1], \boldsymbol{b}[1]) = \boldsymbol{x^1}[1] > \min(\boldsymbol{a}[1], \boldsymbol{b}[1])$. Now, by the definition of $\boldsymbol{x}^{i+1}$, it follows that if $\boldsymbol{x^i}[1] > \boldsymbol{a}[1]$, then $\boldsymbol{x}^{i+1}[1] > \min(\boldsymbol{x^i}[1], \boldsymbol{a}[1]) = \boldsymbol{a}[1]$ (just note that at least one of $f(\boldsymbol{x^i}[1], \boldsymbol{a}[1])$ and $f(\boldsymbol{a}[1], \boldsymbol{x^i}[1])$ is strictly larger than $\min(\boldsymbol{x^i}[1], \boldsymbol{a}[1]) = \boldsymbol{a}[1]$). Hence, it follows by induction that $\boldsymbol{x^n}[1] > \boldsymbol{a}[1]$.

Thus, we have a contradiction with the fact that $a[1]$ is maximal for all tuples in R agreeing with $\boldsymbol{t_{\max}}$ on the coordinates in I. Hence, our assumption was wrong and $\boldsymbol{t_{\max}}$ is in R.

The algorithm for solving W-MAX SOL(Γ) when Γ is generalised max-closed is a simple consistency-based algorithm. The algorithm, which is based on pair-wise consistency, closely follows the algorithm for CSPs over max-closed constraint languages from [33].

We first need to introduce some terminology.

Definition 5.7 *Given a constraint $C_i = (s_i, R_i)$ and a (ordered) subset s'_i of the variables in s_i where $(i_1, i_2, \ldots, i_k)$ are the indices in s_i of the elements in s'_i. The* projection *of C_i onto the variables in s'_i is denoted by $\pi_{s'_i} C_i$ and defined as: $\pi_{s'_i} C_i = C'_i = (s'_i, R'_i)$ where R'_i is the relation $\{(\boldsymbol{a}[i_1], \boldsymbol{a}[i_2], \ldots, \boldsymbol{a}[i_k]) \mid \boldsymbol{a} \in R_i\}$.*

Definition 5.8 *For any pair of constraints $C_i = (s_i, R_i)$, $C_j = (s_j, R_j)$, the* join *of C_i and C_j, denoted $C_i \bowtie C_j$, is the constraint on $s_i \cup s_j$ containing all tuples $\boldsymbol{t}$ such that $\pi_{s_i}\{\boldsymbol{t}\} \in R_i$ and $\pi_{s_j}\{\boldsymbol{t}\} \in R_j$.*

Definition 5.9 ([30]) *An instance of a constraint satisfaction problem $I = (V, D, C)$ is* pair-wise consistent *if and only if for any pair of constraints $C_i = (s_i, R_i)$, $C_j = (s_j, R_j)$ in C, it holds that the constraint resulting from projecting C_i onto the variables in $s_i \cap s_j$ equals the constraint resulting from projecting C_j onto the variables in $s_i \cap s_j$, i.e., $\pi_{s_i \cap s_j} C_i = \pi_{s_i \cap s_j} C_j$.*

We are now ready to prove the tractability of generalised max-closed constraint languages.

Theorem 5.10 *If Γ is generalised max-closed, then* W-MAX SOL(Γ) *is in* **PO**.

Proof. Since $Inv(f)$ is a relational clone, constraints built over $Inv(f)$ are invariant under taking joins and projections [31, Lemma 2.8] (i.e., the underlying relations are still invariant under f). It is proved in [30] that any set of constraints can be reduced to an equivalent set of pair-wise consistent constrains in polynomial time. Since the set of pair-wise consistent constraints can be obtained by repeated application of the join and projection operations, the underlying relations in the resulting constraints are still in $Inv(f)$.

Hence, given an instance $I = (V, D, C, w)$ of W-MAX SOL($Inv(f)$), we can assume that the constraints in C are pair-wise consistent. We prove that for pair-wise consistent C, either C has a constraint with a constraint relation that do not contain any tuples (i.e., no assignment satisfies the constraint and there is no solution) or we can find the optimal solution in polynomial time.

Assume that C has no empty constraints. For each variable x_i, let d_i be the maximum value allowed for that variable by some constraint C_j (where x_i is in the constraint scope of C_j). We will prove that $(d_1, \ldots, d_n)$ is an optimal solution to I. Obviously, if $(d_1, \ldots, d_n)$ is a solution to I, then it is the optimal solution. Hence, it is sufficient to prove that $(d_1, \ldots, d_n)$ is a solution to I.

Assume, with the aim of reaching a contradiction, that $(d_1, \ldots, d_n)$ is not a solution to I. Then, there exists a constraint C_j in C not satisfied by $(d_1, \ldots, d_n)$. Since the constraint relation corresponding to C_j is generalised max-closed, there exists a variable x_i in the constraint scope of C_j such that C_j has no solution where d_i is assigned to x_i. Note that it is essential that C_j is generalised max-closed to rule out the possibility that there exist two variables x_i and x_j in the constraint scope of C_j such that C_j has two solutions t, u where $t(x_i) = d_i$ and $u(x_j) = d_j$, but C_j has no solution s where $s(x_i) = d_i$ and $s(x_j) = d_j$. We know that there exists a constraint C_i in C having x_i in its constraint scope and d_i an allowed value for x_i. This contradicts the fact that C is pair-wise consistent. Thus, $(d_1, \ldots, d_n)$ is a solution to I.

6 Maximal Constraint Languages

A *maximal constraint language* Γ is a constraint language such that $\langle\Gamma\rangle \subset R_D$, and if $R \notin \langle\Gamma\rangle$, then $\langle\Gamma \cup \{R\}\rangle = R_D$. That is, the maximal constraint languages are the largest constraint languages that are not able to express all finitary relations over D. This implies, among other things, that there exists an operation f such that $\langle\Gamma\rangle = Inv(f)$ whenever Γ is a maximal constraint language [45]. Relational clones $\langle\Gamma\rangle$ such that Γ is a maximal constraint language are called maximal relational clones. The complexity of the CSP(Γ) problem for all maximal constraint languages on domains $|D| \leq 3$ was determined in [14]. Moreover, it was shown in [14] that the only case that remained to be classified in order to extend the classification to all maximal constraint languages over a finite domain was the case where $\langle\Gamma\rangle = Inv(f)$ for binary commutative idempotent operations f. These constraint languages were finally classified by Bulatov in [8].

Theorem 6.1 ([8, 14]) *Let Γ be a maximal constraint language on an arbi-*

trary finite domain D. Then, CSP(Γ) *is in* **P** *if* $\langle\Gamma\rangle = Inv(f)$ *where* f *is a constant operation, a majority operation, a binary commutative idempotent operation, or an affine operation. Otherwise,* CSP(Γ) *is* **NP***-complete.*

In this section, we classify the approximability of W-MAX SOL(Γ) for all maximal constraint languages Γ over $|D| \leq 4$. Moreover, we prove that the only cases that remain to be classified, in order to extend the classification to all maximal constraint languages over finite domains, are constraint languages Γ such that $\langle\Gamma\rangle$ is invariant under a binary commutative idempotent operation. We also prove that if a certain conjecture regarding minimal clones generated by binary operations, due to Szczepara [47], holds, then our classification can be extended to capture also these last cases.

Theorem 6.2 *Let Γ be maximal constraint language on a finite domain D, with $|D| \leq 4$, and $\langle\Gamma\rangle = Inv(f)$.*

1. *If Γ is generalised max-closed or an injective constraint language, then* W-MAX SOL(Γ) *is in* **PO***;*
2. *else if f is an affine operation, a constant operation different from the constant 0 operation, or a binary commutative idempotent operation satisfying $f(0, b) > 0$ for all $b \in D \setminus \{0\}$ (assuming $0 \in D$); or if $0 \notin D$ and f is a binary commutative idempotent operation or a majority operation, then* W-MAX SOL(Γ) *is* **APX***-complete;*
3. *else if f is a binary commutative idempotent operation or a majority operation, then* W-MAX SOL(Γ) *is* **poly-APX***-complete;*
4. *else if f is the constant 0 operation, then finding a solution with non-zero measure is* **NP***-hard;*
5. *otherwise, finding a feasible solution is* **NP***-hard.*

Moreover, if Conjecture 131 from [47] holds, then the results above hold for arbitrary finite domains D.

The proof of the preceding theorem consists of a careful analysis of the approximability of W-MAX SOL(Γ) for all maximal constraint languages Γ such that $\langle\Gamma\rangle = Inv(f)$, where f is one of the types of operations in Theorem 6.1. These results are presented below.

6.1 Constant Operation

We begin by considering maximal constraint languages that are invariant under constant operations. Given an instance $I = (V, D, C)$ of a CSP problem, we define the *constraint graph* of I to be $G = (V, E)$ where $\{v, v'\} \in E$ if there is at least one constraint $c \in C$ which have both v and v' in its constraint scope.

Lemma 6.3 *Let $d^* = \max(D)$ and $\langle C_d \rangle = Inv(f_d)$ where $f_d : D \to D$ satisfies $f_d(x) = d$ for all $x \in D$. Then,* W-MAX SOL(C_{d^*}) *is in* **PO**, W-MAX SOL(C_d) *is* **APX**-*complete if $d \in D \backslash \{d^*, 0\}$, and it is* **NP**-*hard to find a solution with non-zero measure for* W-MAX SOL(C_0).

Proof. The tractability of W-MAX SOL(C_{d^*}) is trivial, since the optimum solution is obtained by assigning d^* to all variables.

For the **APX**-hardness of W-MAX SOL(C_d) ($d \in D \backslash \{d^*, 0\}$), it is sufficient to note that $\{(d,d),(d,d^*),(d^*,d)\}$ is in $\langle C_d \rangle$, and since $0 < d < d^*$ it follows from Lemma 4.3 that W-MAX SOL(C_d) is **APX**-hard. It is easy to realise that W-MAX SOL(C_d) is in **APX**, since we can obtain a $\frac{d^*}{d}$-approximate solution by assigning the value d to all variables.

The fact that it is **NP**-hard to find a solution with non-zero measure for W-MAX SOL(C_0) over the Boolean domain $\{0,1\}$ is proved in [37, Lemma 6.23]. To prove that it is **NP**-hard to find a solution with non-zero measure for W-MAX SOL(C_0) over a domain D of size ≥ 3, we give a reduction from the well-known **NP**-complete problem POSITIVE-1-IN-3-SAT [26], i.e., CSP($\{R\}$) with $R = \{(1,0,0),(0,1,0),(0,0,1)\}$. It is easy to see that POSITIVE-1-IN-3-SAT restricted to instances where the constraint graph is connected is still **NP**-complete.

Now, let $R' = \{(b,a,a),(a,b,a),(a,a,b),(0,0,0)\}$, where $0 < a < b$ and $a, b, 0 \in D$. For an instance $I = (V, D, C)$ of CSP($\{R\}$) where the constraint graph of I is connected, create an instance I' of W-MAX SOL($\{R'\}$) by replacing all occurrences of R by R' and giving all variables weight 1. Since the constraint graph is connected, I has a solution if and only if I' has a solution with non-zero measure, and since $R' \in C_0$, it follows that it is **NP**-hard to find a solution with non-zero measure for W-MAX SOL(C_0).

6.2 Majority Operation

Maximal constraint languages based on majority operations are fairly easy to analyse due to the results in §4.

Lemma 6.4 *Let m be an arbitrary majority operation on D. Then,* W-MAX SOL($Inv(m)$) *is* **APX**-*complete if $0 \notin D$ and* **poly-APX**-*complete if $0 \in D$.*

Proof. Arbitrarily choose elements $a, b \in D$ such that $a < b$. Then, it is easy to see that $\{(a,a),(a,b),(b,a)\}$ is in $Inv(m)$. Thus, by Proposition 4.1 and Lemmas 4.3 and 4.4, it follows that W-MAX SOL($Inv(m)$) is **APX**-complete or **poly-APX**-complete depending on whether 0 is in D or not.

6.3 Affine Operation

We split the proof of this result into two parts. The first part, §6.3.1, contains the hardness result: for every affine operation $a : D^3 \to D$, W-MAX SOL($Inv(a)$) is **APX**-hard. The proof is based on a reduction from MAX-p-CUT which is a well-known **APX**-complete problem [2]. Membership in **APX** is proved in §6.3.2 by presenting an approximation algorithm with constant performance ratio.

We will denote the affine operation on the group G by a_G, i.e., if $G = (D, +_G, -_G)$ then $a_G(x, y, z) = x -_G y +_G z$.

6.3.1 APX-hardness

We will now prove Theorem 6.11 which states that relations invariant under an affine operation give rise to **APX**-hard W-MAX SOL-problems. We need a number of lemmas before we can prove this result, though. We begin by giving an L-reduction from MAX-p-CUT to W-MAX SOL EQN$(\mathbb{Z}_p, g)$ where p is prime. MAX-p-CUT and W-MAX SOL EQN are defined as follows:

Definition 6.5 ([2]) MAX-p-CUT *is an optimisation problem with*

Instance: *A graph $G = (V, E)$.*

Solution: *A partition of V into p disjoint sets $C_1, C_2, \ldots, C_p$.*

Measure: *The number of edges between the disjoint sets, i.e.,*

$$\sum_{i=1}^{p-1} \sum_{j=i+1}^{p} |\{\{v, v'\} \in E \mid v \in C_i \text{ and } v' \in C_j\}|.$$

Definition 6.6 ([38]) *Let $G = (D, +_G, -_G)$ be a group and $g : D \to \mathbb{N}$ a function.* W-MAX SOL EQN(G, g) *is an optimisation problem with*

Instance: *A triple (V, E, w) where, $V = \{v_1, v_2, \ldots, v_n\}$ is a set of variables, E is a set of equations of the form $w_1 +_G \ldots +_G w_k = 0_G$, where each w_i is either a variable (e.g., "v_4"), an inverted variable (e.g., "$-_G v_7$") or a group constant, and w is a weight function $w : V \to \mathbb{N}$.*

Solution: *An assignment $f : V \to D$ to the variables such that all equations are satisfied.*

Measure: $\sum_{v \in V} w(v) g(f(v))$

We do not require the group G to be Abelian in the definition of W-MAX SOL EQN but this will always be the case in this article. Note that the function g and the group G are not parts of the input so W-MAX SOL EQN(G, g) is a problem parameterised by G and g. We refer the reader to [38] for more information on the problem W-MAX SOL EQN$(\mathbb{Z}_p, g)$,

The following lemma follows from the proof of Proposition 2.3 in [20].

Lemma 6.7 *For any instance $I = (V, E)$ of* MAX-p-CUT*, we have* $\mathrm{OPT}(I) \geq |E|(1 - 1/p)$.

We can now prove the **APX**-hardness of W-MAX SOL EQN.

Lemma 6.8 *For every prime p and every non-constant function $g : \mathbb{Z}_p \to \mathbb{N}$,* W-MAX SOL EQN$(\mathbb{Z}_p, g)$ *is* **APX**-*hard.*

Proof. Given an instance $I = (V, E)$ of MAX-p-CUT, we construct an instance $F(I)$ of W-MAX SOL EQN$(\mathbb{Z}_p, g)$ where, for every vertex $v_i \in V$, we create a variable x_i and give it weight 0, and for every edge $\{v_i, v_j\} \in E$, we create p variables $z_{ij}^{(k)}$ for $k = 0, \ldots, p-1$ and give them weight 1. Let $g_{\min}$ denote an element in $\mathbb{Z}_p$ that minimises g, i.e.,

$$\min_{x \in \mathbb{Z}_p} g(x) = g(g_{\min})$$

and let g_s denote the sum

$$\sum_{k=0}^{p-1} g(k).$$

For every edge $\{v_i, v_j\} \in E$, we introduce the equations

$$k(x_i - x_j) + g_{\min} = z_{ij}^{(k)}$$

for $k = 0, \ldots, p-1$. If $x_i = x_j$, then the p equations for the edge $\{v_i, v_j\}$ will contribute $pg(g_{\min})$ to the measure of the solution. On the other hand, if $x_i \neq x_j$ then the p equations will contribute g_s to the measure.

Given a solution s' to $F(I)$, we can construct a solution s to I in the following way: let $s(v_i) = s'(x_i)$, i.e. for every vertex v_i, place this vertex in partition $s'(x_i)$. The measures of the solutions s and s' are related to each other by the equality

$$m'(F(I), s') = |E| \cdot p \cdot g(g_{\min}) + (g_s - p \cdot g(g_{\min})) \cdot m(I, s). \tag{1}$$

From (1), we get

$$\text{OPT}(F(I)) = |E| \cdot p \cdot g(g_{\min}) + (g_s - p \cdot g(g_{\min})) \cdot \text{OPT}(I) \tag{2}$$

and from Lemma 6.7, we have that $\text{OPT}(I) \geq |E| \cdot (1 - 1/p)$ which implies $\text{OPT}(I) \geq |E|/p$. By combining this with (2), we can conclude that

$$\begin{aligned}\text{OPT}(F(I)) &= \text{OPT}(I) \left(\frac{|E| \cdot p \cdot g(g_{\min})}{\text{OPT}(I)} + g_s - p \cdot g(g_{\min}) \right) \\ &\leq \text{OPT}(I) \Big(p^2 \cdot g(g_{\min}) + g_s - p \cdot g(g_{\min}) \Big).\end{aligned}$$

Hence, $\beta = p(p-1) \cdot g(g_{min}) + g_s$ is an appropriate parameter for the L-reduction.

We will now deduce an appropriate γ-parameter for the L-reduction: from (1) and (2) we get

$$|\text{OPT}(F(I)) - m'(F(I), s')| = (g_s - p \cdot g(g_{\min})) \cdot |\text{OPT}(I) - m(I, s)|$$

so, $\gamma = 1/(g_s - p \cdot g(g_{\min}))$ is sufficient (γ is well-defined because a non-constant g implies $g_s > p \cdot g_{\min}$).

We need two lemmas before we can prove the **APX**-hardness of affine relations. Let $v_1, v_2, \ldots, v_k$ be a collection of variables, $G = (D, +_G, -_G)$ an Abelian group, and E an equation of the form $x_1 +_G +_G x_2 +_G \ldots +_G x_n = c$, where each x_i is a (possibly inverted) variable and $c \in D$. Note that each variable may occur several times in E. The solutions to E may be seen as a k-ary relation R_E on D^k. The following two lemmas are well-known [32].

Lemma 6.9 *The relation R_E is invariant under a_G.*

Lemma 6.10 *If P is a coset of G, then P is invariant under a_G.*

We now have all results needed to prove the main theorem of this section.

Theorem 6.11 W-MAX SOL($Inv(a_G)$) *is* **APX**-*hard for every affine operation a_G.*

Proof. We show that there exists a prime p and a non-constant function $h : \mathbb{Z}_p \to \mathbb{N}$ such that W-MAX SOL EQN($\mathbb{Z}_p, h$) can be S-reduced to W-MAX SOL($Inv(a_G)$). The result will then follow from Lemma 6.8.

Let p be a prime such that $\mathbb{Z}_p$ is isomorphic to a subgroup H of G. We know that such a p always exist by the fundamental theorem of finitely generated Abelian groups. Let α be the isomorphism which maps elements of $\mathbb{Z}_p$ to elements of H and let $h = \alpha$. (Note that $H \subset \mathbb{N}$ since the domain is a subset of $\mathbb{N}$. Consequently, h may be viewed as a function from $\mathbb{Z}_p$ to $\mathbb{N}$.)

Let $I = (V, E, w)$ be an instance of W-MAX SOL EQN($\mathbb{Z}_p, h$) with variables $V = \{v_1, \ldots, v_n\}$ and equations $E = \{e_1, \ldots, e_m\}$. We will construct an instance $I' = (V, D, C, w)$ of W-MAX SOL($Inv(a_G)$).

Let U be the unary relation for which $x \in U \iff x \in H$; this relation is in $Inv(a_G)$ by Lemma 6.10. For every equation $E_i \in E$, there is a corresponding pair (s_i, R_i) where s_i is a list of variables and R_i is a relation in $Inv(a_G)$ such that the set of solutions to E_i are exactly the tuples which satisfies (s_i, R_i) by Lemma 6.9. We can now construct C:

$$C = \{(v_i, U) \mid 1 \leq i \leq n\} \cup \{(s_i, R_i) \mid 1 \leq i \leq m\}.$$

It is easy to see that I and I' are essentially the same in the sense that every feasible solution to I is also a feasible solution to I', and they have the same measure. The converse is also true, every feasible solution to I' is also a feasible solution to I. Hence, we have given a S-reduction from W-MAX SOL EQN($\mathbb{Z}_p, h$) to W-MAX SOL($Inv(a_G)$). As h is not constant (it is in fact injective), it follows from Lemma 6.8 that W-MAX SOL EQN($\mathbb{Z}_p, h$) is **APX**-hard. This S-reduction implies that W-MAX SOL($Inv(a_G)$) is **APX**-hard.

6.3.2 Membership in APX

We will now prove that relations that are invariant under an affine operation give rise to problems which are in **APX**. It has been proved that a relation which is invariant under an affine operation is a coset of a subgroup of some Abelian group [32]. We will give an approximation algorithm for the more general problem when the relations are cosets of subgroups of a finite group.

Our algorithm is based on an algorithm by Bulatov and Dalmau [12] for deciding the satisfiability of *Mal'tsev constraints*. A *Mal'tsev operation* is a ternary operation m such that $m(x,y,y) = m(y,y,x) = x$ for all $x, y \in D$. If a constraint language Γ is invariant under a Mal'tsev operation, then Bulatov and Dalmau have proved that $\text{Csp}(\Gamma)$ is solvable in polynomial time. We note that every affine operation is a Mal'tsev operation since $x -_G y +_G y = x$ and $y -_G y +_G x = x$.

Let G^k denote the direct product of k copies of G. We are now ready to prove containment in **APX**.

Theorem 6.12 *Let $G = (D; \cdot, ^{-1})$ be a finite group and let Γ be a constraint language such that for each $R \in \Gamma$ there is an integer k such that R is a coset of some subgroup of G^k. Then* W-Max Sol(Γ) *is in* **APX**.

Proof. Let $I = (V, D, C, w)$ be an arbitrary instance of W-Max Sol(Γ) where $V = \{v_1, \ldots, v_n\}$. Feasible solutions to our optimisation problem can be viewed as certain elements in $H = G^n$. Each constraint $C_i \in C$ defines a coset $a_i \cdot J_i$ of H with representative $a_i \in H$, for some subgroup J_i of H. The set of solutions to the problem is the intersection of all those cosets. Thus, $S = \bigcap_{i=1}^{|C|} a_i \cdot J_i$ denotes the set of all solutions.

Since Γ is invariant under the affine operation $a_G(x,y,z) = x \cdot y^{-1} \cdot z$ and a_G is a Mal'tsev operation, we can decide if there are any solutions to I in polynomial-time [12]. Clearly, S is empty if and only if there are no solutions. It is well-known that an intersection of a set of cosets is either empty or a coset so if $S \neq \emptyset$, then S is a coset.

We will represent the elements of G^n by vectors $\boldsymbol{x} = (x_1, \ldots, x_n)$ where each x_i is an element of G. For any instance I, we define $\mathcal{R}(I)$ to be the random variable which is uniformly distributed over the set of solutions to I. Let V_i denote the random variable which corresponds to the value which will be assigned to v_i by $\mathcal{R}(I)$. We claim that V_i is uniformly distributed over some subset of G. As S is a coset there is a subgroup S' of G^n and an element $\boldsymbol{s} \in S$ such that $S = \boldsymbol{s} \cdot S'$. Assume, for the sake of contradiction, that V_i is not uniformly distributed. Then, there are group elements $a, b \in G$ such that the sets

$$X_a = \{\boldsymbol{x} \in S' \mid x_i = a\} \text{ and } X_b = \{\boldsymbol{x} \in S' \mid x_i = b\}$$

have different cardinality. Assume that $|X_a| > |X_b|$. Arbitrarily pick $\boldsymbol{y} \in X_a, \boldsymbol{z} \in X_b$ and construct the set $Z = \{\boldsymbol{x} \cdot \boldsymbol{y}^{-1} \cdot \boldsymbol{z} \mid \boldsymbol{x} \in X_a\}$. From the definition of Z and the fact that S' is invariant under a_G, it follows that $Z \subseteq S'$. For each $\boldsymbol{x} \in Z$ we have $x_1 = b$, hence $Z \subseteq X_b$. However, we also have $|Z| = |X_a|$, which contradicts the assumption that $|X_a| > |X_b|$. We conclude that this cannot hold and V_i is uniformly distributed. Hence, for each $1 \leq i \leq n$, V_i is uniformly distributed.

Now, let A denote the set of indices such that for every $i \in A$, $\Pr[V_i = c_i] = 1$ for some $c_i \in G$. That is, A contains the indices of the variables V_i which are constant in every feasible solution. Let B contain the indices for the variables which are not constant in every solution, i.e., $B = [n] \setminus A$.

Let $S^* = \sum_{i \in B} w(v_i) \max(D) + \sum_{i \in A} w(v_i) c_i$ and note that $S^* \geq \text{OPT}$. Furthermore, let

$$E_{\min} = \min_{X \subseteq G, |X|>1} \frac{1}{|X|} \cdot \sum_{x \in X} x$$

and note that $\max(D) > E_{\min} > 0$.

The expected value of the measure of $\mathcal{R}(I)$ can now be estimated as

$$\begin{aligned} \mathrm{E}\left[\sum_{i=1}^{n} w(v_i) V_i\right] &= \sum_{i \in A} w(v_i) \mathrm{E}[V_i] + \sum_{i \in B} w(v_i) \mathrm{E}[V_i] \\ &\geq \sum_{i \in A} w(v_i) c_i + E_{\min} \sum_{i \in B} w(v_i) \geq \frac{E_{\min}}{\max(D)} S^* \geq \frac{E_{\min}}{\max(D)} \text{OPT}. \end{aligned} \tag{3}$$

Since $E_{\min} / \max(D) > 0$, it follows that the measure of $\mathcal{R}(I)$ has, in expectation, a constant performance ratio. We will denote $\frac{E_{\min}}{\max(D)} \cdot \text{OPT}$ by E.

To get a deterministic polynomial-time algorithm, note that for any instance I we can use the algorithm by Bulatov and Dalmau [12] to compute the two sums in (3) in polynomial-time. Hence, we can compute the expected measure of $\mathcal{R}(I)$ in polynomial-time. Our algorithm is presented in Figure 1.

We claim that the following loop invariant holds in the algorithm: before line 4 is executed it is always the case that the expected measure of $\mathcal{R}(I_i)$ is at least E.

We first prove the correctness of the algorithm assuming that the loop invariant holds. From the loop invariant it follows that the expected measure of $\mathcal{R}(I_{|V|+1})$ is at least E. In $I_{|V|+1}$ there is, for each variable $v_i \in V$, a constraint of the form $v_i = x_i$, therefore there is only one solution to $I_{|V|+1}$. This solution will be returned by the algorithm.

We now prove that the loop invariant holds. The first time line 4 is reached the expected performance ratio of $\mathcal{R}(I_1)$ is at least E, per the calculations above. Now assume that the loop invariant holds in iteration $i = k \leq |V|$; we will prove that it also holds in iteration $i = k + 1$. Since the performance ratio of $\mathcal{R}(I_k)$ is at least E, there must be some value $x \in D$ such that when v_i is fixed to x, the performance ratio of $\mathcal{R}(I_{k+1})$ is at least E. This element will be found by the algorithm as it maximises the expected performance ratio $\mathcal{R}(I_{k+1})$. Hence, the loop invariant holds for $i = k + 1$.

6.4 Binary Commutative Idempotent Operation

We now investigate the complexity of W-MAX SOL(Γ) for maximal constraint languages Γ satisfying $\langle \Gamma \rangle = \mathit{Inv}(f)$ where f is a binary commutative idempotent operation.

Input: An instance $I = (V, D, C, w)$ of W-MAX SOL(Γ)
Output: A solution with performance ratio at least $E_{\min} / \max(D)$, or "no solution" if there are no solutions.

1. Return "no solution" if there are no solutions (use Bulatov and Dalmau's algorithm to check this)
2. Let $I_1 = I$.
3. For each i from 1 to $|V|$:
4.   For each $x \in D$:
5.   Let $I_i = I$ and add the constraint $v_i = x$ to I_i
6.   If there is no solution to I_i, then go to 8.
7.   Compute the expected measure of $\mathcal{R}(I_i)$
8.   Remove the constraint $v_i = x$ from I_i
9.   Let $x_i \in D$ be the value which maximises the expected measure of $\mathcal{R}(I_i)$ in the computations in 4–8. Create a new instance, I_{i+1}, which is identical to I_i except for the addition of the constraint $v_i = x_i$.
10. Return the unique solution to $I_{|V|+1}$.

Figure 1: The algorithm in Theorem 6.12

For an Abelian group $(G; +_G, -_G)$ of prime order p there is a unique corresponding field. We will use $\cdot_G$ and 1_G for multiplication and the multiplicative identity element of this field, respectively. Furthermore, let z_G be the unique element in G such that $z_G + z_G = 1_G$. Note that for $G = \mathbb{Z}_p$ we get $1_G = 1$ and $z_G = \frac{p+1}{2}$.

Let $\mathcal{A}$ denote the set of operations $f(x, y) = z_G \cdot_G (x +_G y)$, where G is an Abelian group of prime order $p = |D|$ and $p > 2$. The proof will be partitioned into two main cases due to the following result:

Lemma 6.13 ([14, 49]) *If $Inv(f)$ is a maximal relational clone and f is a binary idempotent operation, then either*

1. *$Inv(f) = Inv(g)$ where $g \in \mathcal{A}$; or*
2. *$B \in Inv(f)$ for some two-element $B \subseteq D$.*

The classification result is given in the next lemma together with a proof outline. Full proofs concerning the case when $Inv(f) = Inv(g)$ and $g \in \mathcal{A}$ can be found in §6.4.1. In §6.4.2 we give a complete characterisation of the complexity for the second case for domains D such that $|D| \leq 4$. Finally, in §6.4.3 we extend the classification to general domains under the assumption of a conjecture due to Szczepara (Conjecture 6.18).

Lemma 6.14 *Let f be a binary commutative idempotent operation on D such that $Inv(f)$ is a maximal relational clone, and let Γ be a constraint language such that $\langle \Gamma \rangle = Inv(f)$.*

- *If $Inv(f) = Inv(g)$ for some $g \in \mathcal{A}$, then* W-Max Sol(Γ) *is* **APX**-*complete;*
- *else if $|D| \leq 4$ and there exist $a, b \in D$ such that $a < b$ and $f(a, b) = a$, then let a_* be the minimal such element (according to $<$), then*
  - W-Max Sol(Γ) *is* **poly-APX**-*complete if $a_* = 0$, and*
  - **APX**-*complete if $a_* > 0$.*
- *Otherwise, if $|D| \leq 4$, then* W-Max Sol(Γ) *is in* **PO**.

Proof. If $Inv(f) = Inv(g)$ and $g \in \mathcal{A}$, then the result follows from §6.4.1.

If there exist $a, b \in D$ such that $a < b$ and $f(a, b) = a$, then we need to consider two cases depending on a_*. If $a_* = 0$, then W-Max Sol(Γ) is **poly-APX**-hard by Lemma 4.4 and a member of **poly-APX** by Lemma 4.2 since Csp is in **P** [14]. If $a_* > 0$, then W-Max Sol(Γ) is **APX**-complete by Lemma 6.25 in §6.4.2.

Finally, if there do not exist any $a, b \in D$ such that $a < b$ and $f(a, b) = a$, then f acts as the max operation on every two-element $B \subseteq D$ such that $B \in Inv(f)$. Lemma 6.26 shows that f is a generalised max operation in this case, and W-Max Sol(Γ) is in **PO** by Theorem 5.10.

6.4.1 $f \in \mathcal{A}$

We will now prove that W-MAX SOL(Γ) is **APX**-complete whenever $f \in \mathcal{A}$ and $\langle \Gamma \rangle = Inv(f)$.

Lemma 6.15 *Let $f(x, y) = z_G \cdot_G (x +_G y)$, where G is an Abelian group of prime order $p = |D| > 2$ and $Inv(f)$ is a maximal relational clone. Then,* W-MAX SOL(Γ) *is* **APX**-*complete if $\langle \Gamma \rangle = Inv(f)$.*

Proof. We will give the proof for $G = \mathbb{Z}_p$ and after that we will argue that the proof easily can be adapted to the general case.

Let $q = \frac{p+1}{2}$ and f be the function $f(x, y) = q(x+y) \pmod p$. We will show that we can express $x - y + z$ through f.

Note that

$$\sum_{i=1}^{p-1} q^i = \frac{1 - q^p}{1 - q} - 1 = 0 \pmod p. \tag{4}$$

(The second equality follows from Fermat's little theorem: $a^{p-1} = 1 \pmod p$ for any prime p and integer a not divisible by p.) By using (4) and Fermat's little theorem again, we get

$$\sum_{i=1}^{p-2} q^i = -1 \pmod p. \tag{5}$$

We can now express $x - y + z$ as follows:

$$\begin{aligned}
&\underbrace{f(f(f(\ldots f(f(f(}_{p-1 \text{ times}} x, z), y), y) \ldots), y), y) &&= \\
&q(q(q(\ldots q(q(q(x + z) + y) + y) + \ldots) + y) + y) &&= \\
&q^{p-1}x + q^{p-1}z + \sum_{i=1}^{p-2} q^i y &&= \\
&x - y + z \pmod p
\end{aligned}$$

where the final equality follows from (4), (5) and Fermat's little theorem.

As any Abelian group G of prime order is isomorphic to $\mathbb{Z}_p$, it is not hard to see that $x -_G y +_G z$ can be expressed through f for any such group. Since $Inv(f)$ is a maximal relational clone, $x -_G y +_G z$ can be expressed through f, and $x -_G y +_G z$ is not a projection, it follows that $Inv(f) = Inv(x -_G y +_G z)$. We now get containment in **APX** from Theorem 6.12 and **APX**-hardness from Theorem 6.11.

6.4.2 $f \notin \mathcal{A}$

In the first part of this section we classify the complexity of W-MAX SOL(Γ) when $\langle \Gamma \rangle = Inv(f)$ for all 2-semilattice operations f. It is noted in [9] that binary operations f such that $Inv(f)$ is a maximal constraint language on $|D| \leq 4$ are either 2-semilattices or otherwise $\text{Csp}(\Gamma)$ is **NP**-complete. Hence, we get a classification of the complexity of W-MAX SOL(Γ) when $\langle\Gamma\rangle = Inv(f)$ is a maximal constraint language over $|D| \leq 4$ and f is a binary operation.

The second result in this section is a complete complexity classification of W-MAX SOL(Γ) for maximal constraint languages Γ, such that $\langle\Gamma\rangle = Inv(f)$ where f is a binary operation, under the condition that Conjecture 131 from [47] holds.

Recall that a 2-semilattice operation f is an operation satisfying the conditions $f(x,x) = x$, $f(x,y) = f(y,x)$, and $f(x, f(x,y)) = f(x,y)$.

Lemma 6.16 *Let f be a 2-semilattice operation on D and $\langle\Gamma\rangle = Inv(f)$. If there exist $a, b \in D$ such that $a < b$ and $f(a,b) = a$ and $a > 0$ is the minimal such a, then* W-MAX SOL(Γ) *is* **APX**-*complete.*

Proof. The **APX**-hardness part is clear. What remains is to show that the problem is in **APX**. We begin by proving that $U = D \setminus \{0\}$ is in $Inv(f)$. Assume that $f(a,b) = 0$ and $a, b > 0$, then $f(a, f(a,b)) = f(a,b) = 0$ and consequently $f(a,0) = 0$ contradicting the assumption that $a > 0$ was the minimal such element. Hence, $f(a,b) = 0$ if and only if $a = b = 0$. In particular U is in $Inv(f)$.

We continue with the actual proof of the lemma. Let $I = (V, D, C, w)$ be an arbitrary instance of W-MAX SOL(Γ). Define $V' \subseteq V$ such that

$$V' = \{v \in V \mid S(v) = 0 \text{ for every solution } S \text{ of } I\}.$$

We see that V' can be computed in polynomial time: a variable v is in V' if and only if the CSP instance $(V, D, C \cup \{((v), U)\})$ is not satisfiable.

Given two assignments $A, B : V \to D$, we define the assignment $f(A, B)$ such that $f(A,B)(v) = f(A(v), B(v))$. We note that if A and B are solutions of I, then $f(A,B)$ is a solution to I, too: indeed, arbitrarily choose one constraint $((x_1, \ldots, x_k), r) \in C$. Then, $(A(x_1), \ldots, A(x_k)) \in r$ and $(B(x_1), \ldots, B(x_k)) \in r$ which implies that $(f(A(x_1), B(x_1)), \ldots, f(A(x_k), B(x_k))) \in r$, too.

Let $S_1, \ldots, S_m$ be an enumeration of all solutions of I and define

$$S^+ = f(S_1, f(S_2, f(S_3 \ldots f(S_{m-1}, S_m) \ldots))).$$

By the choice of V' and the fact that $f(c,d) = 0$ if and only if $c = d = 0$, we see that the solution S^+ has the following property: $S^+(v) = 0$ if and only if $v \in V'$. Let p denote the second least element in D, and note that $\text{OPT}(I) \geq \sum_{v \in V \setminus V'} w(v)p = c$. Thus, by finding a solution with measure $\geq c$, we have approximated I within $(\max D)/p$ and W-MAX SOL(Γ) is in **APX**. To find such a solution, we consider the instance $I' = (V, D, C', w)$, where $C' = C \cup \{((v), u) \mid v \in V \setminus V'\}$. This instance has feasible solutions (since S^+ is a solution) and every solution has measure $\geq c$. Finally, a concrete solution can be found in polynomial time by the result in [19].

Lemma 6.17 *If f is a 2-semilattice operation such that $f \notin \mathcal{A}$, Γ is a maximal constraint language satisfying $\langle\Gamma\rangle = Inv(f)$, and for all two-element $B \in Inv(f)$ the operation f acts as the* max *operation on B, then* W-MAX SOL(Γ) *is in* **PO**.

Proof. What we will prove is that if f acts as max on all two-element $B \in Inv(f)$, then f is a generalised max operation and consequently W-MAX SOL(Γ) is in **PO**.

First note that if $a \neq b$ and $f(a,b) = a$, then by assumption $a > b$ and $f(a,b) > \min\{a,b\}$. Now, if $f(a,b) \neq a$, then $f(a, f(a,b)) = f(a,b)$ and by assumption f is max on $\{a, f(a,b)\}$. As a consequence of this we get $f(a,b) > \min\{a,b\}$. Now, $f(a,b) > \min\{a,b\}$ for all $a \neq b$. Moreover, f is idempotent, so f is a generalised max operation and tractability follows from Theorem 5.10.

We have now completely classified the complexity of W-MAX SOL(Γ) for all constraint languages Γ such that $\langle\Gamma\rangle$ is maximal and $|D| \leq 4$.

6.4.3 Complete Classification under a Conjecture

In this section we will prove that the validity of Conjecture 131 from [47] implies a complete complexity classification of W-MAX SOL(Γ) for all constraint languages Γ such that $\langle\Gamma\rangle$ is maximal. Given a binary operation f on D, the *fixity* of f is denoted $\mathcal{F}(f)$ and is defined by

$$\mathcal{F}(f) = \{(x,y) \in D^2 \mid f(x,y) \in \{x,y\}\}.$$

The *fixity-count* of f is defined to be the cardinality of $\mathcal{F}(f)$ and is denoted $|\mathcal{F}(f)|$.

Conjecture 6.18 ([47], Conjecture 131) *If $Inv(f)$ is a maximal relational clone and $Inv(f') = Inv(f)$, then $|\mathcal{F}(f)| = |\mathcal{F}(f')|$.*

Altough Conjecture 6.18 is not known to hold in the general case, it has been verified for small domains. In particular, it was shown in [47] that for domains D such that $|D| \leq 4$ the conjecture holds. Our proof builds on a construction that facilitates the study of operation f—the details are collected in Lemma 6.19. The underlying idea and the proof of Lemma 6.19 are inspired by Lemma 3 in [14].

Let f be a binary operation on D and define operations $f_1, f_2, \ldots : D^2 \to D$ inductively:

$$f_1(x,y) = f(x,y)$$

$$f_{n+1}(x,y) = f(x, f_n(x,y)).$$

Lemma 6.19 *Assume f to be a binary commutative idempotent operation on D such that $Inv(f)$ is a maximal relational clone and $Inv(f) \neq Inv(g)$ for every $g \in \mathcal{A}$. The following holds:*

1. *$f|_B = f_n|_B$ for every $n \geq 1$ and every two-element $B \subseteq D$ in $Inv(f)$; and*

2. $Inv(f) = Inv(f_n)$, $n \geq 1$.

Proof. 1. Arbitrarily choose a two-element $\{a, b\} = B \subseteq D$ in $Inv(f)$. There are two possible binary commutative idempotent operations on B, namely max and min. We assume without loss of generality that $f|_B = \max$ and prove the result by induction over n. Since $f_1 = f$, the claim holds for $n = 1$. Assume it holds for $n = k$ and consider f_{k+1}. We see that $f_{k+1}(a, b) = f(a, f_k(a, b))$ and, by the induction hypothesis, $f_k(a, b) = \max(a, b)$. Hence, $f_{k+1}(a, b) = \max(a, \max(a, b)) = \max(a, b)$.

2. Obviously, $f_n \in Pol(Inv(f))$ and, thus, $Inv(f) \subseteq Inv(f_n) \subseteq R_D$. Since $Inv(f) \neq Inv(g)$ for every $g \in \mathcal{A}$, we know from Lemma 6.13 that there is some two-element $B \in Inv(f)$. By the proof above, we also know that $f|_B = f_n|_B$ so $f_n|_B$ (and consequently f_n) is not a projection. Thus, $Inv(f_n) \neq R_D$, since $Inv(f') = R_D$ if and only if f' is a projection. By the assumption that $Inv(f)$ is a maximal relational clone and the fact that $Inv(f) \subseteq Inv(f_n) \subsetneq R_D$, we can draw the conclusion that $Inv(f) = Inv(f_n)$.

We will now present some technical machinery that is needed for proving Lemmas 6.25 and 6.26.

Lemma 6.20 ([47], Lemma 28) *Let f be an idempotent binary operation and $n \in \mathbb{N}$. Then, $\mathcal{F}(f) \subseteq \mathcal{F}(f_n)$.*

Proof. Let $(x, y) \in \mathcal{F}(f)$. Then, either $f(x, y) = x$ or $f(x, y) = y$. Now

$$f(x,y) = x \Longrightarrow f_n(x,y) = \underbrace{f(x, f(x, \ldots, f(x,y)\ldots))}_{n\ times} =$$

$$\underbrace{f(x, f(x, \ldots, f(x,x)\ldots))}_{n-1\ times} = x \Longrightarrow f_n(x,y) = x.$$

While

$$f(x,y) = y \Longrightarrow f_n(x,y) = \underbrace{f(x, f(x, \ldots, f(x,y)\ldots))}_{n\ times} =$$

$$\underbrace{f(x, f(x, \ldots, f(x,y)\ldots))}_{n-1\ times} \Longrightarrow f_n(x,y) = y.$$

Assuming that Conjecture 6.18 holds, we get the following corollary as a consequence of Lemma 6.20.

Corollary 6.21 *If $Inv(f)$ is maximal relational clone such that f is commutative, idempotent, and $(x, y) \in \mathcal{F}(f_k)$, (i.e. $f_k(x, y) \in \{x, y\}$), then $\{x, y\} \in Inv(f)$ (i.e. $\{x, y\}$ is a subalgebra of f).*

Proof. By Lemma 6.20, we have $\mathcal{F}(f) \subseteq \mathcal{F}(f_k)$, and if Conjecture 6.18 holds, then $|\mathcal{F}(f)| = |\mathcal{F}(f_k)|$ which implies that $\mathcal{F}(f) = \mathcal{F}(f_k)$. Now, if $f_k(x, y) \in \{x, y\}$, then $f(x, y) \in \{x, y\}$ and by the commutativity of f we have $f(y, x) \in \{x, y\}$. Since f is idempotent, it is clear that $\{x, y\} \in Inv(f)$.

We continue by introducing a digraph associated with the binary operation f. This digraph enables us to make efficient use of Lemma 6.19. Given a binary operation $f : D^2 \to D$, we define $G_f = (V, E)$ such that $V = D \times D$ and $E = \{((a, b), (a, f(a, b))) \mid a, b \in D\}$. We make the following observations about G_f:

(1) an edge $((a, b), (a, c))$ implies that $f(a, b) = c$;

(2) every vertex has out-degree 1; and

(3) there is no edge $((a, b), (c, d))$ with $a \neq c$.

We extract some more information about G_f in the next three lemmas.

Lemma 6.22 *The digraph G_f contains no directed cycle.*

Proof. Assume G_f contains a directed cycle. Fact (3) allows us to assume (without loss of generality) that the cycle is $(0, 1), (0, 2), \dots, (0, k), (0, 1)$ for some $k \geq 2$. Fact (1) tells us that $f(0, 1) = 2$, $f(0, 2) = 3$, $\dots$, $f(0, k - 1) = k$ and $f(0, k) = 1$. Furthermore, one can see that $f_2(0, 1) = 3$, $f_2(0, 2) = 4$, $\dots$, and inductively $f_p(0, i) = i + p \pmod{k}$. This implies that $f_k(0, 1) = 1 + k \pmod{k} = 1$. By Corollary 6.21, $\{0, 1\}$ is a subalgebra of $Inv(f)$ which contradicts the fact that $f(0, 1) = 2$.

Lemma 6.23 *Every path in G_f of length $n \geq |D|$ ends in a reflexive vertex, i.e., $f_n(a, b) = c$ implies that (a, c) is a reflexive vertex.*

Proof. Assume that G_f contains a path P of length $n \geq |D|$. This path can contain at most $|D|$ distinct vertices by fact (3). Fact (2) together with the acyclicity of G_f implies that at least one vertex v on P is reflexive; by using fact (2) once again, we see that there exists exactly one reflexive vertex on P and it must be the last vertex.

Lemma 6.24 *If $f_n(a, b) = c$, then (a, c) is a reflexive vertex in G_f.*

Proof. By Lemma 6.23, every path in G_f of length $n \geq |D|$ ends in a reflexive vertex. Hence, $(a, f_n(a, b)) = (a, c)$ is a reflexive vertex.

Lemma 6.25 *Let f be a binary commutative idempotent operation on D such that Γ is a maximal constraint language satisfying $\langle \Gamma \rangle = Inv(f)$. If there exist $a, b \in D$ such that $a < b$ and $f(a, b) = a$ and $a > 0$ is the minimal such a, then (assuming Conjecture 6.18)* W-Max Sol(Γ) *is* **APX***-complete.*

Proof. The **APX**-hardness part is clear. What remains is to show that the problem is in **APX**. We begin the proof by proving that the unary relation $U = D \setminus \{0\}$ is a member of $Inv(f)$. Consider the digraph G_f. As we have already observed in Lemma 6.22, there are no cycles $(a, b_1), \dots, (a, b_k), (a, b_1)$,

$k \geq 2$, in G_f and every path of length $n \geq |D|$ ends in a reflexive vertex (by Lemma 6.23). Obviously, no vertex $(a, 0)$ $(a > 0)$ in G_f is reflexive since this implies that $f(a, 0) = 0$ which is a contradiction. In particular, there exists no path in G_f of length $n \geq |D|$ starting in a vertex (a, b) $(a > 0)$ and ending in a vertex $(a, 0)$, since this implies that $(a, 0)$ is reflexive by Lemma 6.23.

We can now conclude that $f_n(a, b) > 0$ when $a > 0$: if $f_n(a, b) = 0$, then $(a, 0)$ is reflexive by Lemma 6.24 which would lead to a contradiction. Hence, $f_n(a, b) > 0$ whenever $a, b \in D \setminus \{0\} = U$ so U is in $Inv(f_n)$, and by Lemma 6.19(2), U is in $Inv(f)$, too. We also note that $U \in Inv(f)$ together with the assumption that $f(0, b) > 0$ for all $b > 0$ implies that $f(c, d) = 0$ if and only if $c = d = 0$. The rest of the proof is identical to the second part of the proof of Lemma 6.16.

Lemma 6.26 *If f is a binary commutative idempotent operation such that $f \notin \mathcal{A}$, Γ is a maximal constraint language satisfying $\langle \Gamma \rangle = Inv(f)$, and for all two-element $B \in Inv(f)$ f, acts as the* max *operation on B, then (assuming Conjecture 6.18)* W-MAX SOL(Γ) *is in* **PO**.

Proof. What we will prove is that if $f \notin \mathcal{A}$ and f acts as max on all two-element $B \in Inv(f)$, then there exists a generalised max function f' such that $Inv(f') = Inv(f)$ and, hence, W-MAX SOL(Γ) is in **PO**.

Recall that there are no cycles in G_f and every path of length $n = |D|$ must end in a reflexive vertex by Lemmas 6.22 and 6.23. We also note that if G_f contains a reflexive vertex (a, c) with $a \neq c$, then $f(a, c) = c$ and $c > a$ since f is assumed to act as the max operation on all two-element $B \in Inv(f)$.

We now claim that f_n is a generalised max operation. Arbitrarily choose $a, b \in D$. If $f_n(a, b) = c$ $(a \neq c)$, then there is a path in G_f from (a, b) to a reflexive vertex (a, c), and $c > a$ as explained above. If $f_n(a, b) = a$, then $\{a, b\} \in Inv(f)$ by Corollary 6.21. Since $f(a, b) = \max(a, b)$, Lemma 6.19(1) implies that $f_n(a, b) = \max(a, b)$. Thus, f_n is a generalised max-operation.

7 Homogeneous Constraint Languages

In this section, we will classify the complexity of MAX SOL when the constraint language is *homogeneous*. A constraint language is called homogeneous if every *permutation relation* is contained in the language.

Definition 7.1 *A relation R is a* permutation relation *if there is a permutation $\pi : D \to D$ such that*

$$R = \{(x, \pi(x)) \mid x \in D\}.$$

Let Q denote the set of all permutation relations on D. The main result of this section is Theorem 7.16 which gives a complete classification of the complexity of W-MAX SOL(Γ) when $Q \subseteq \Gamma$. The theorem provide the exact borderlines between tractability, **APX**-completeness, **poly-APX**-completeness, and **NP**-hardness of finding a feasible solution.

As a direct consequence of Theorem 7.16, we get that the class of injective relations is a maximal tractable class for W-MAX SOL(Γ). That is, if we add a single relation which is not an injective relation to the class of all injective relations, then the problem is no longer in **PO** (unless **P** = **NP**).

Dalmau has completely classified the complexity of CSP(Γ) when Γ is a homogeneous constraint language [23], and this classification relies heavily on the structure of homogeneous algebras. An algebra is called *homogeneous* if and only if every permutation on its universe is an automorphism of the algebra. We will not need a formal definition of homogeneous algebras and refer the reader to [39, 40, 48] for further information on their properties. All homogeneous algebras have been completely classified by Marczewski [40] and Marchenkov [39].

Our classification for the approximability of W-MAX SOL(Γ) when Γ is a homogeneous constraint language uses the same approach as in [23], namely, we exploit the inclusion structure of homogeneous algebras (as proved in [39, 40]). By Theorem 3.3, it is sufficient to consider constraint languages Γ that are relational clones. This is where the homogeneous algebras comes in: the classification of homogeneous algebras gives us a classification of all homogeneous relational clones, and in particular their inclusion structure (lattice) under set inclusion. We refer the reader to [48] for a deeper treatment of homogeneous algebras and their inclusion structure.

The lattice of all homogeneous relational clones on a domain D having n (≥ 5) elements is given in Figure 2. The lattices for the corresponding relational clones over smaller domains (i.e., $2 \leq |D| \leq 4$) contains some exceptional relational clones and are presented separately in Figures 3–5. Note that the corresponding lattices presented in [23] and [48] are the dual of ours, since they instead consider the inclusion structure among the corresponding clones of operations (but the two approaches are in fact equivalent as shown in [43, Satz 3.1.2]). To understand the lattices we first need some definitions.

Throughout this section n denotes the size of the domain D, i.e., $n = |D|$.

Definition 7.2

- *The* switching operation s *is defined by*

$$s(a,b,c) = \begin{cases} c & \textit{if } a = b, \\ b & \textit{if } a = c, \\ a & \textit{otherwise.} \end{cases}$$

- *The* discriminator operation t *is defined by*

$$t(a,b,c) = \begin{cases} c & \textit{if } a = b, \\ a & \textit{otherwise.} \end{cases}$$

- *The* dual discriminator operation d *is defined by*

$$d(a,b,c) = \begin{cases} a & \textit{if } a = b, \\ c & \textit{otherwise.} \end{cases}$$

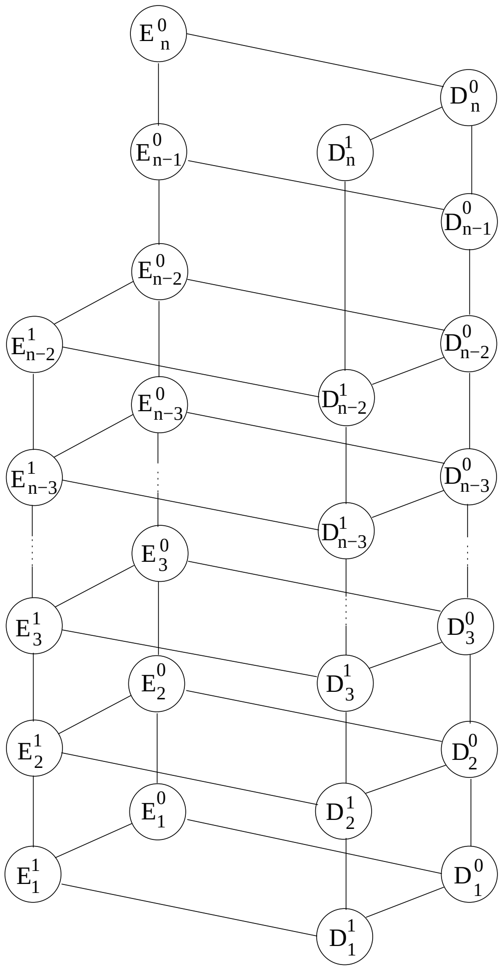

Figure 2: Lattice of all homogeneous relational clones over domain size $n \geq 5$.

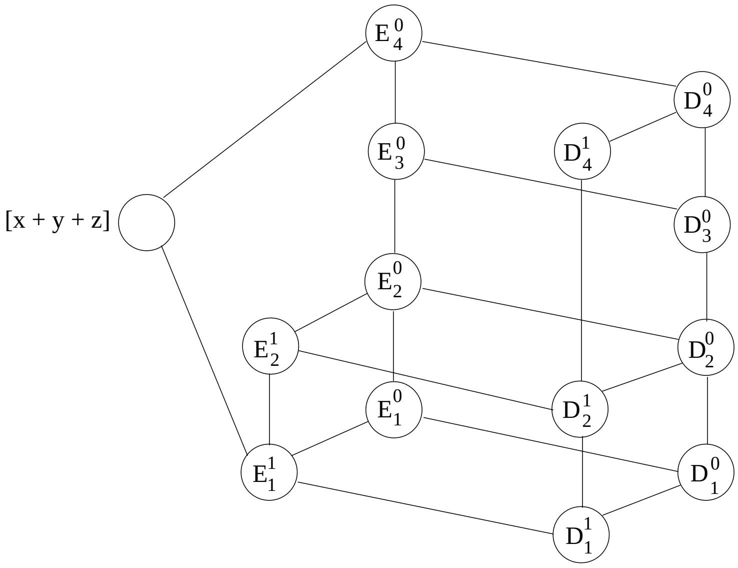

Figure 3: Lattice of all homogeneous relational clones over domain size $n = 4$.

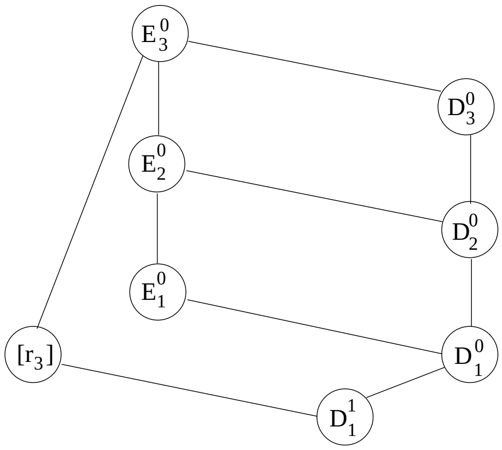

Figure 4: Lattice of all homogeneous relational clones over domain size $n = 3$.

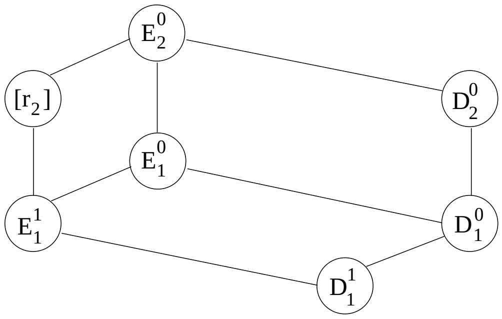

Figure 5: Lattice of all homogeneous relational clones over domain size $n = 2$.

- *The k-ary* near projection operation l_k $(3 \leq k \leq n)$ *defined by*

$$l_k(a_1, \ldots, a_k) = \begin{cases} a_1 & \text{if } |\{a_1, \ldots, a_k\}| < k, \\ a_k & \text{otherwise.} \end{cases}$$

- *The $(n-1)$-ary operation r_n defined by*

$$r_n(a_1, \ldots, a_{n-1}) = \begin{cases} a_1 & \text{if } |\{a_1, \ldots, a_{n-1}\}| < n-1, \\ a_n & \text{otherwise.} \end{cases}$$

  In the second case we have $\{a_n\} = D \setminus \{a_1, \ldots, a_{n-1}\}$.

- *The $(n-1)$-ary operation d_n (where $n \geq 4$) defined by*

$$d_n(a_1, \ldots, a_{n-1}) = \begin{cases} d(a_1, a_2, a_3) & \text{if } |\{a_1, \ldots, a_{n-1}\}| < n-1, \\ a_n & \text{otherwise.} \end{cases}$$

  In the second case we have $\{a_n\} = D \setminus \{a_1, \ldots, a_{n-1}\}$.

- *the operation $x + y + z$ where $(D, +, -)$ is a 4-element group of exponent* 2.

The notation in the lattices in Figures 2–5 is explained below.

- $D_1^0 = Inv(t)$;
- $D_i^0 = Inv(\{d, l_{i+1}\})$ for $2 \leq i \leq n-1$;
- $D_n^0 = Inv(d)$;
- $D_1^1 = Inv(\{t, r_n\})$;
- $D_i^1 = Inv(\{d, l_{i+1}, r_n\})$ for $2 \leq i \leq n-2$;
- $D_n^1 = Inv(d_n)$;

- $E_1^0 = Inv(s)$;
- $E_i^0 = Inv(l_{i+1})$ for $2 \leq i \leq n-1$;
- $E_n^0 = R_D$, i.e., all finitary relations over D;
- $E_1^1 = Inv(\{s, r_n\})$ for $n \neq 3$;
- $E_i^1 = Inv(\{l_{i+1}, r_n\})$ for $2 \leq i \leq n-3$;
- $E_{n-2}^1 = Inv(r_n)$ for $n \geq 4$.

Note that the relational clones described above depend on the size of the domain $|D| = n$. Hence, E_2^0 in Figure 4 ($|D| = 3$) is not the same relational clone as E_2^0 ($|D| = 4$) in Figure 3. Also note that when we state our (in)approximability results by saying, for example that, W-Max Sol(E_1^1) is **APX**-complete, we mean that W-Max Sol(E_1^1) is **APX**-complete for all sizes of the domain where E_1^1 is defined (e.g., E_1^1 is not defined for $n = 3$). We always assume that $n = |D| \geq 2$.

We now state Dalmau's classification for the complexity of Csp(Γ) for homogeneous constraint languages Γ.

Theorem 7.3 ([23]) *Let Γ be a homogeneous constraint language. Then,* Csp(Γ) *is in* **P** *if $Pol(\Gamma)$ contains the dual discriminator operation d, the switching operation s, or an affine operation. Otherwise,* Csp(Γ) *is* **NP**-*complete.*

We have the following corollary of Dalmau's classification.

Corollary 7.4 *Let Γ be a homogeneous constraint language. Then,* W-Max Sol(Γ) *is in* **poly-APX** *if $Pol(\Gamma)$ contains the dual discriminator operation d, the switching operation s, or an affine operation. Otherwise, it is* **NP**-*hard to find a feasible solution to* W-Max Sol(Γ).

Proof. All dual discriminator operations, switching operations, and affine operations are idempotent (i.e., $f(x, x, x) = x$ for all $x \in D$). Hence, Γ is invariant under a dual discriminator operation, switching operation, or an affine operation if and only if $\Gamma^c = \{\Gamma \cup \{\{(d_1)\}, \ldots, \{(d_n)\}\}$ is invariant under the corresponding operation. Thus, it follows from Theorem 7.3 that Csp(Γ^c) is in **P** if $Pol(\Gamma)$ contains the dual discriminator operation d, the switching operation s, or an affine operation. This together with Lemma 4.2 gives us that W-Max Sol(Γ) is in **poly-APX** if $Pol(\Gamma)$ contains the dual discriminator operation d, the switching operation s, or an affine operation.

The **NP**-hardness part follows immediately from Theorem 7.3.

We begin by investigating the approximability of W-Max Sol(Γ) for some particular homogeneous constraint languages Γ.

Lemma 7.5 W-Max Sol(D_n^0) *is in* **APX** *if* $0 \notin D$ *and in* **poly-APX** *otherwise.*

Proof. Remember that $D_n^0 = Inv(d)$. Hence, membership in **poly-APX** follows directly from Corollary 7.4. It is known from Dalmau's classification that $\text{CSP}(D_n^0)$ is in **P**. Thus, by Proposition 4.1, it follows that W-MAX SOL(D_n^0) is in **APX** when $0 \notin D$.

Lemma 7.6 W-MAX SOL(D_2^1) *is* **APX**-*complete if* $0 \notin D$ *and* **poly-APX**-*complete if* $0 \in D$.

Proof. Choose any $a, b \in D$ such that $a < b$. The relation $r = \{(a,a),(a,b),(b,a)\}$ is in $D_i^1 = Inv(\{d, l_{i+1}, r_n\})$ for $2 \leq i \leq n-2$. Hence, by Lemmas 4.3 and 4.4, it follows that W-MAX SOL(D_2^1) is **APX**-hard if $0 \notin D$ and **poly-APX**-hard if $0 \in D$. This together with Lemma 7.5 and fact that $D_i^1 \subseteq D_n^0$ give us that W-MAX SOL(D_2^1) is **APX**-complete if $0 \notin D$ and **poly-APX**-complete if $0 \in D$.

Lemma 7.7 *Finding a feasible solution to* W-MAX SOL(E_2^1) *is* **NP**-*hard.*

Proof. Remember that $E_2^1 = Inv(\{l_3, r_n\})$ when $n > 4$ and $E_2^1 = Inv(r_n)$ when $n = 4$, so it follows from Dalmau's classification that $\text{CSP}(E_2^1)$ is **NP**-complete.

Lemma 7.8 W-MAX SOL(D_1^0) *is in* **PO**.

Proof. It is well-known that $D_1^0 = Inv(t) = \langle I^D \rangle$; for instance, it is a direct consequence of Theorem 4.2 in [48]. Hence, W-MAX SOL(D_1^0) is in **PO** by the results in §5.1.

Lemma 7.9 W-MAX SOL(E_1^0) *is in* **APX**.

Proof. Remember that $E_1^0 = Inv(s)$. Dalmau gives a polynomial-time algorithm for $\text{CSP}(Inv(s))$ in [23] (he actually gives a polynomial-time algorithm for the more general class of para-primal problems). Dalmau's algorithm exploits in a clever way the internal structure of para-primal algebras to show that any instance I of $\text{CSP}(Inv(s))$ can be split into independent subproblems $I_1, \ldots, I_j$, such that

- the set of solutions is preserved (i.e., any solution to I is also a solution to each of the independent subproblems, and any solution to all of the independent subproblems is also a solution to I); and
- each I_i ($1 \leq i \leq j$) is either an instance of $\text{CSP}(Inv(t))$ or an instance of $\text{CSP}(Inv(a))$, where t is the discriminator operation and a is an affine operation.

Hence, to show that W-MAX SOL(E_1^0) is in **APX**, we first use Dalmau's algorithm to reduce the problem (in a solution preserving manner) to a set of independent W-MAX SOL(Γ) problems where Γ is either invariant under an affine operation or the discriminator operation. We know from Lemma 7.8 that W-MAX SOL(Γ) is in **PO** when Γ is invariant under the discriminator operation, and from Theorem 6.12 we know that W-MAX SOL(Γ) is in **APX** when Γ is invariant under an affine operation. Since all the independent subproblems are in **APX**, we get that the original W-MAX SOL(E_1^0) problem is also in **APX**.

Lemma 7.10 W-MAX SOL(E_1^1) *is* **APX**-*complete.*

Proof. Remember that $E_1^1 = Inv(\{s, r_n\})$. Note that $E_1^1 \subseteq E_1^0$ so membership in **APX** follows from Lemma 7.9.

For the hardness part, we begin by considering the general case when $n = |D| \geq 4$. Choose an arbitrary two-element subset $\{a, b\}$ of D (without loss of generality assume that $a < b$) and let $(G, +, -)$ be the two element group on $G = \{a, b\}$ defined by $a + a = a$, $a + b = b + a = b$, and $b + b = a$. Let Γ be the set of all relations expressible as the set of solutions to equations over $(G, +, -)$. It is easy to realise that Γ is invariant under r_n (since r_n ($n \geq 4$) acts as a projection on $\{a, b\}$). Furthermore Γ is invariant under s since $s(x, y, z)$ acts as the affine operation $x + y + z$ on $\{a, b\}$. It is proved in Lemma 6.8 that W-MAX SOL EQN($\mathbb{Z}_2, g$) is **APX**-hard, and hence W-MAX SOL(Γ) is **APX**-complete. For $n = |D| = 3$ there is no relational clone of the type $E_1^1 = Inv(\{s, r_n\})$ so the only case that remains to be dealt with is the case where $n = |D| = 2$.

Without loss of generality assume that $D = \{a, b\}$ where $a < b$. Again consider the group $(G, +, -)$ on $\{a, b\}$. Let $\Gamma = \{R_1, R_2\}$ where R_1 is the (4-ary) relation on D which is the set of solutions to the equation $x_1 + x_2 + x_3 + x_4 = a$ and R_2 is the (binary) relation representing the set of solutions to the equation $y_1 + y_2 = b$. Furthermore, let $\Gamma_{0,1} = \{R_1, R_2\}$ denote the special case where $a = 0, b = 1$ (i.e., $D = \{0, 1\}$). It has been proved in [37, Lemma 6.9] that MAX SOL($\Gamma_{0,1}$) is **APX**-complete. Let I be an instance of MAX SOL($\Gamma_{0,1}$) containing k variables. It is easy to realise that if I has a solution, then $\text{OPT}(I) \geq k/2$ (just note that the complement of any solution to I is also a solution to I). We give an L-reduction from MAX SOL($\Gamma_{0,1}$) to MAX SOL($\Gamma_{a,b}$). Let $F(I)$ be the instance of MAX SOL($\Gamma_{a,b}$) where all occurrences of 0 has been replaced by a and all occurrences of 1 has been replaced by b. Since $\text{OPT}(I) \geq k/2$, we get that $\text{OPT}(F(I)) \leq bk \leq 2b \cdot \text{OPT}(I)$ and $\beta = 2b$ is a valid parameter in the L-reduction. Let s be an arbitrary solution to $F(I)$ and define $G(F(I), s)$ to be the corresponding solution to I where a is replaced by 0 and b is replaced by 1. Then,

$$|m(I, G(F(I), s)) - \text{OPT}(I)| \leq \frac{1}{b - a} |m(F(I), s) - \text{OPT}(F(I))|$$

and $\gamma = \frac{1}{b-a}$ is a valid parameter in the L-reduction. This completes the **APX**-hardness proof for MAX SOL($\Gamma_{a,b}$). Now, the unary operation r_2 acts as the non-identity permutation on $\{a, b\}$ (i.e., $r_2(a) = b$, and $r_2(b) = a$) so both R_1 and R_2 are invariant under r_2. As we have already observed, s acts as the affine operation on $\{a, b\}$ and R_1 and R_2 are invariant under s, too. Hence, $\Gamma_{a,b} \subseteq Inv(\{s, r_2\})$ which concludes the proof.

We have now proved all the results needed to give a complete classification for the approximability of W-MAX SOL(Γ) for all homogeneous relational clones Γ over domains D of size at least 5. In order to complete the classification also for domains of size 2, 3 and 4, we need to consider some exceptional homogeneous relational clones. For domains of size 4, we need to consider the homogeneous relational clone $Inv(x + y + z)$ where $+$ is the operation of a 4-element group $(D, +, -)$ of exponent 2 (i.e., a 4-element group such that for all $a \in D$, $a + a = e$ where e is the identity element in $(D, +, -)$).

Lemma 7.11 *Let $f(x, y, z) = x + y + z$ where $(D, +, -)$ is a group of exponent 2. Then,* W-MAX SOL($Inv(f)$) *is* **APX**-*complete.*

Proof. The operation $x+y+x$ is the affine operation on $(D, +, -)$ (since $-y = y$ in $(D, +, -)$) and it follows directly from Theorem 6.11 and Theorem 6.12 that W-MAX SOL($Inv(x + y + z)$) is **APX**-complete.

For 3-element domains, it remains to classify the approximability of W-MAX SOL($Inv(r_3)$) where r_3 is the binary operation defined as follows:

$$r(a_1, a_2) = \begin{cases} a_1 & \text{if } a_1 = a_2, \\ a_3 & \text{where } \{a_3\} = D \setminus \{a_1, a_2\} \text{ otherwise.} \end{cases}$$

Lemma 7.12 W-MAX SOL($Inv(r_3)$) *is* **APX**-*complete.*

Proof. The operation $r_3(x, y)$ is actually an example of an operation of the type $\frac{p+1}{2}(x + y)$ (where $+$ is the operation of an Abelian group of order $|D| = p$) from Lemma 6.15. In our case $p = 3$ and $r_3(x, y) = 2x + 2y$ where $+$ is the operation of the Abelian group $(D, +, -)$ isomorphic to $\mathbb{Z}_3$. Hence, it follows from Lemma 6.15 that W-MAX SOL($Inv(r_3)$) is **APX**-complete.

For 3-element domains we also need to classify the approximability of W-MAX SOL(E_2^0) since hardness no longer follows from the hardness of W-MAX SOL(E_2^1) (there is no relational clone E_2^1 over 3-element domains).

Lemma 7.13 *It is* **NP**-*hard to find a feasible solution to* W-MAX SOL(E_2^0).

Proof. Immediate consequence of Theorem 7.3.

Similarly, we also need to classify the approximability of W-MAX SOL(D_2^0) since hardness no longer follows from the hardness of W-MAX SOL(D_2^1) (there is no relational clone D_2^1 over 3-element domains).

Lemma 7.14 W-MAX SOL(D_2^0) *is* **APX**-*complete if* $0 \notin D$ *and* **poly-APX**-*complete if* $0 \in D$.

Proof. Remember that $D_2^0 = Inv(\{d, l_3\})$. It follows from the proof of Lemma 7.6 that W-MAX SOL(D_2^0) is **APX**-complete if $0 \notin D$ and **poly-APX**-complete if $0 \in D$.

For two-element domains $\{a, b\}$, we need to classify the unary operation r_2 which acts as the non-identity permutation (i.e., $r_2(a) = b$ and $r_2(b) = a$).

Lemma 7.15 *Finding a feasible solution to* W-Max Sol($Inv(r_2)$) *is* **NP**-*hard.*

Proof. Follows from Dalmau's classification.

Finally, we are in the position to present the complete classification for the approximability of all homogeneous constraint languages.

Theorem 7.16 *Let* Γ *be a homogeneous constraint language.*

1. *If* $\langle\Gamma\rangle \in \{E_j^i \mid i \in \{0,1\}, j \geq 2\}$ *or* $\langle\Gamma\rangle = Inv(r_2)$, *then it is* **NP**-*hard to find a feasible solution to* W-Max Sol(Γ);
2. *else if,* $0 \in D$ *and* $\langle\Gamma\rangle \in \{D_j^i \mid i \in \{0,1\}, j \geq 2\}$, *then* W-Max Sol($\Gamma$) *is* **poly-APX**-*complete;*
3. *else if,*

$$\langle\Gamma\rangle \in \{E_1^1, E_1^0, Inv(x+y+z), Inv(r_3)\} \quad \textit{or}$$
$$\langle\Gamma\rangle \in \{D_j^i \mid i \in \{0,1\}, j \geq 2\} \textit{ and } 0 \notin D,$$

   then W-Max Sol(Γ) *is* **APX**-*complete;*
4. *otherwise,* $\langle\Gamma\rangle \in \{D_1^1, D_1^0\}$ *and* W-Max Sol(Γ) *is in* **PO**.

Proof. We know from Theorem 3.3 that it is sufficient to consider constraint languages that are relational clones.

1. It is proved in Lemmas 7.7 and 7.13 that it is **NP**-hard to find a feasible solution to W-Max Sol(E_2^1) and W-Max Sol(E_2^0). **NP**-hardness of finding a feasible solution to W-Max Sol($Inv(r_2)$) was proved in Lemma 7.15. The result follows from the fact that $E_2^1 \subseteq E_j^1$ and $E_2^0 \subseteq E_j^0$ $(2 \leq j \leq n)$.
2. Membership in **poly-APX** is proved in Lemma 7.5 and **poly-APX**-hardness of W-Max Sol(D_2^1) and W-Max Sol(D_2^0) when $0 \in D$ is proved in Lemmas 7.6 and 7.14. The result follows from the fact that $D_2^1 \subseteq D_j^1 \subseteq D_n^0$ and $D_2^0 \subseteq D_j^0 \subseteq D_n^0$ $(2 \leq j \leq n)$.
3. It is proved in Lemmas 7.11 and 7.12 that W-Max Sol($Inv(x+y+z)$) and W-Max Sol($Inv(r_3)$) are **APX**-complete. It is proved in Lemma 7.9 that W-Max Sol(E_1^0) is in **APX**. **APX**-hardness for W-Max Sol(E_1^1) is proved in Lemma 7.10. For $n \geq 4$ or $n = 2$ we have that $E_1^1 \subseteq E_1^0$ and it follows that W-Max Sol(E_1^0) and W-Max Sol(E_1^1) are **APX**-complete. For $n = 3$, there exists no E_1^1 so **APX**-hardness for W-Max Sol(E_1^0) must be proved separately. Since $E_1^0 = Inv(s)$ it is easy to see that the proof of Lemma 7.10 gives **APX**-hardness for W-Max Sol(E_1^0).

   Membership in **APX** for W-Max Sol(D_n^0) when $0 \notin D$ is the first part of Lemma 7.5. **APX**-hardness of W-Max Sol(D_2^1) and W-Max

Sol(D_2^0) are proved in Lemmas 7.6 and 7.14, respectively. Hence, **APX**-completeness of W-Max Sol(D_j^0) and W-Max Sol(D_j^1) $(2 \leq j \leq n)$ when $0 \notin D$ follows from the fact that $D_2^1 \subseteq D_j^1 \subseteq D_n^0$, $D_2^0 \subseteq D_j^0 \subseteq D_n^0$ $(2 \leq j \leq n)$.

4. It is proved in Lemma 7.8 that W-Max Sol(D_1^0) is in **PO** and the result follows from the fact that $D_1^1 \subseteq D_1^0$.

As a direct consequence of the preceding theorem, we get that the class of injective relations is a maximal tractable class for W-Max Sol(Γ). That is, if we add a single relation which is not an injective relation to the class of all injective relations, then the problem is no longer in **PO** (unless **P** = **NP**).

Corollary 7.17 *Let Γ_I^D be the class of injective relations and R an arbitrary relation which is not in Γ_I^D. Then,* W-Max Sol($\Gamma_I^D \cup \{R\}$) *is not in* **PO** *(unless* **P** = **NP***).*

Proof. We know from the proof of Lemma 7.8 that $D_1^0 = Inv(t) = \Gamma_I^D = \langle I^D \rangle$. It follows from the lattices of homogeneous relational clones that either $E_1^0 \subseteq \langle D_1^0 \cup \{R\} \rangle$ or $D_2^0 \subseteq \langle D_1^0 \cup \{R\} \rangle$. We know from Lemma 7.10 and Lemma 7.6 that both W-Max Sol(E_1^0) and W-Max Sol(D_2^0) are **APX**-hard. Thus, W-Max Sol($D_1^0 \cup \{R\}$) is **APX**-hard and it follows that W-Max Sol($\Gamma_I^D \cup \{R\}$) is not in **PO** (unless **P** = **NP**).

8 Conclusions

We view this article as a first step towards a better understanding of the approximability of non-Boolean W-Max Sol. The ultimate long-term goal for this research is, of course, to completely classify the approximability for all finite constraint languages. However, we expect this to be a hard problem since not even a complete classification for the corresponding decision problem Csp is known. A more manageable task would be to completely classify W-Max Sol for constraint languages over small domains (say, of size 3 or 4). For size 3, this has already been accomplished for Csp [10] and Max Csp [34]. Another obvious way to extend the results of this paper would be to complete the classification of maximal constraint languages over arbitrary finite domains, perhaps by proving Conjecture 131 from [47].

Our results combined with Khanna et al.'s [37] results for Boolean domains suggest the following conjecture:

Conjecture. For every constraint language Γ, one of the following holds:

1. W-Max Sol(Γ) is in **PO**;

2. W-Max Sol(Γ) is **APX**-complete;

3. W-Max Sol(Γ) **poly-APX**-complete;

4. it is **NP**-hard to find a non-zero solution to W-MAX SOL(Γ); or

5. it is **NP**-hard to find any solution to W-MAX SOL(Γ).

If this conjecture is true, then there does not exist any constraint language Γ_1 such that W-MAX SOL(Γ_1) has a polynomial-time approximation scheme (PTAS) but W-MAX SOL(Γ_1) is not in **PO**. Natural such classes exist, however, if one restricts the way constraints are applied to variables (instead of restricting the allowed constraint types). MAXIMUM INDEPENDENT SET (and, equivalently, MAX ONES($\{(0,0),(1,0),(0,1)\}$)) is one example: the unrestricted problem is **poly-APX**-complete and not approximable within $O(n^{1-\epsilon})$, $\epsilon > 0$ (unless **P**=**NP**) [52], but the problem restricted to planar instances admits a PTAS [4]. One may ask several questions in connection with this: is there a constraint language with the properties of Γ_1 above? For which constraint languages does W-MAX SOL admit a PTAS on planar instances? Or more generally: under which restrictions on variable scopes does W-MAX SOL(Γ) admit a PTAS?

It is interesting to note that the applicability of the algebraic approach to W-MAX SOL demonstrated in this article also holds for the corresponding minimisation problem W-MIN SOL, that is, Theorem 3.3 still holds. The question whether the algebraic approach can shed some new light on the intriguing approximability of minimisation problems (as manifested, e.g., in [37]) is an interesting open question.

As mentioned in the introduction, it is known from [37] that the approximability of the weighted and unweighted versions of (W)-MAX SOL coincide for all Boolean constraint languages. We remark that the same result holds for all constraint languages considered in this article. This can be readily verified by observing that the *AP*-reduction from W-MAX ONES to MAX ONES in the proof of Lemma 3.11 in [37] easily generalises to arbitrary finite domains. Hence, we get an *AP*-reduction from W-MAX SOL to MAX SOL. Furthermore, our tractability proofs are given for the weighted version of the problem. In general, it is still an open problem if tractability (i.e., membership in **PO**) of MAX SOL(Γ) implies tractability of W-MAX SOL(Γ) for every constraint language Γ (the *AP*-reduction used above do not give us this result as *AP*-reductions do not, in general, preserve membership in **PO**).

Acknowledgements

The authors would like to thank the anonymous referees for many valuable comments and in particular for pointing out a serious flaw in the (previous) proof of Lemma 6.14. Peter Jonsson is supported by the *Center for Industrial Information Technology* (CENIIT) under grant 04.01, and the *Swedish Research Council* (VR) under grant 621-2003-3421, Fredrik Kuivinen is supported by the *Swedish Research Council* (VR) under grant 621-2002-4126, and Gustav Nordh is supported by the *National Graduate School in Computer Science* (CUGS), Sweden.

References

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

[2] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. M. Spaccamela, and M. Protasi, *Complexity and approximation: Combinatorial optimization problems and their approximability properties*, Springer, 1999.

[3] G. Ausiello, A. D'Atri, and M. Protasi, *Structure preserving reductions among convex optimization problems*, J. Comput. System Sci., 21 (1980), pp. 136–153.

[4] B. Baker, *Approximation algorithms for NP-complete problems on planar graphs*, J. ACM, 41 (1994), pp. 153–180.

[5] S. Bistarelli, U. Montanari, and F. Rossi, *Semiring-based constraint satisfaction and optimization*, J. ACM, 44 (1997), pp. 201–236.

[6] F. Börner, A. Bulatov, P. Jeavons, and A. Krokhin, *Quantified constraints: algorithms and complexity*, in Proceedings of the 17th International Workshop on Computer Science Logic (CSL-2003), 2003, pp. 58–70.

[7] F. Börner, A. Krokhin, A. Bulatov, and P. Jeavons, *Quantified constraints and surjective polymorphisms*, tech. report, RR-02-11, Computing Laboratory, University of Oxford, 2002.

[8] A. Bulatov, *A graph of a relational structure and constraint satisfaction problems*, in Proceedings of the 19th IEEE Symposium on Logic in Computer Science (LICS 2004), 2004, pp. 448–457.

[9] ——, *Combinatorial problems raised from 2-semilattices*, J. Algebra, 298 (2006), pp. 321–339.

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

[11] A. Bulatov and V. Dalmau, *Towards a dichotomy for the counting constraint satisfaction problem*, in Proceedings of the 44th IEEE Symposium on Foundations of Computer Science (FOCS-2003), 2003, pp. 562–571.

[12] ——, *A simple algorithm for Mal'tsev constraints*, SIAM J. Comput., 36 (2006), pp. 16–27.

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

[14] A. Bulatov, A. Krokhin, and P. Jeavons, *The complexity of maximal constraint languages*, in Proceedings of the 33rd ACM Symposium on Theory of Computing (STOC 2001), 2001, pp. 667–674.

[15] E. Böhler, N. Creignou, S. Reith, and H. Vollmer, *Playing with boolean blocks, part I: Post's lattice with applications to complexity theory*, ACM SIGACT-Newsletter, 34 (2003), pp. 38–52.

[16] ——, *Playing with boolean blocks, part II: Constraint satisfaction problems*, ACM SIGACT-Newsletter, 35 (2004), pp. 22–35.

[17] H. Chen, *The complexity of quantified constraint satisfaction: Collapsibility, sink algebras, and the three-element case.* http://arxiv.org/abs/cs.LO/0607106.

[18] ——, *Quantified constraint satisfaction, maximal constraint languages, and symmetric polymorphisms*, in Proceedings of the 22nd Annual Symposium on Theoretical Aspects of Computer Science (STACS-2005), 2005, pp. 315–326.

[19] D. Cohen, *Tractable decision for a constraint language implies tractable search*, Constraints, 9 (2004), pp. 219–229.

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

[21] ——, *The complexity of soft constraint satisfaction*, Artificial Intelligence, 170 (2006), pp. 909–1030.

[22] N. Creignou, S. Khanna, and M. Sudan, *Complexity classifications of Boolean constraint satisfaction problems*, SIAM, Philadelphia, 2001.

[23] V. Dalmau, *A new tractable class of constraint satisfaction problems*, Ann. Math. Artif. Intell., 44 (2005), pp. 61–85.

[24] V. Dalmau and P. Jeavons, *Learnability of quantified formulas*, Theoret. Comput. Sci., (2003), pp. 485–511.

[25] T. Feder and M. Vardi, *The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory*, SIAM J. Comput., 28 (1999), pp. 57–104.

[26] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, 1979.

[27] D. Hochbaum, N. Megiddo, J. Naor, and A. Tamir, *Tight bounds and 2-approximation algorithms for integer programs with two variables per inequality*, Math. Program., 62 (1993), pp. 69–84.

[28] D. Hochbaum and J. Naor, *Simple and fast algorithms for linear and integer programs with two variables per inequality*, SIAM J. Comput., 23 (1994), pp. 1179–1192.

[29] J. Håstad, *Some optimal inapproximability results*, J. ACM, 48 (2001), pp. 798–859.

[30] P. Janssen, P. Jegou, B. Nouguier, and M. Vilarem, *A filtering process for general constraint satisfaction problems: achieving pair-wise consistency using an associated binary representation*, in Proceedings of the IEEE Workshop on Tools for Artificial Intelligence, 1989, pp. 420–427.

[31] P. Jeavons, *On the algebraic structure of combinatorial problems*, Theoret. Comput. Sci., (1998), pp. 185–204.

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

[33] P. Jeavons and M. Cooper, *Tractable constraints on ordered domains*, Artificial Intelligence, 79 (1996), pp. 327–339.

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

[35] P. Jonsson, F. Kuivinen, and G. Nordh, *Approximability of integer programming with generalised constraints*, in Proceedings of the 12th International Conference on Principles and Practice of Constraint Programming (CP-2006), 2006, pp. 256–270.

[36] P. Jonsson and G. Nordh, *Generalised integer programming based on logically defined relations*, in Proceedings of the 31st International Symposium on Mathematical Foundations of Computer Science (MFCS 2006), 2006, pp. 549–560.

[37] S. Khanna, M. Sudan, L. Trevisan, and D. Williamson, *The approximability of constraint satisfaction problems*, SIAM J. Comput., 30 (2001), pp. 1863–1920.

[38] F. Kuivinen, *Tight approximability results for the maximum solution equation problem over Z_p*, in Proceedings of the 30th International Symposium on Mathematical Foundations of Computer Science (MFCS 2005), 2005, pp. 628–639.

[39] S. Marchenkov, *Homogeneous algebras*, Problemy Kibernetiki, 39 (1982), pp. 85–106.

[40] E. Marczewski, *Homogeneous algebras and homogeneous operations*, Fund. Math., 56 (1964), pp. 81–103.

[41] A. Pixley, *Functionally complete algebras generating distributive and permutable classes*, Mathematische Zeitschrift, 114 (1970), pp. 361–372.

[42] E. Post, *The two-valued iterative systems of mathematical logic*, Ann. of Math. Stud., 5 (1941), pp. 1–122.

[43] R. Pöschel and L. Kaluznin, *Funktionen- und Relationenalgebren*, DVW, Berlin, 1979.

[44] R. Quackenbush, *A survey of minimal clones*, Aequationes Mathematicae, 50 (1995), pp. 3–16.

[45] I. Rosenberg, *Minimal clones I: the five types*, in Lectures in Universal Algebra, L. Szabó and Á. Szendrei, eds., North-Holland, 1986.

[46] H. Schnoor and I. Schnoor, *Enumerating all solutions for constraint satisfaction problems*, tech. report, Theoretische Informatik, Universität Hannover, 2006. Available from http://www.thi.uni-hannover.de/forschung/publikationen/daten/schsch06.pdf.

[47] B. Szczepara, *Minimal clones generated by groupoids*, PhD thesis, Université de Móntreal, 1996.

[48] Á. Szendrei, *Clones in Universal Algebra*, vol. 99 of Seminaries de Mathématiques Supéreiores, University of Montreal, 1986.

[49] ———, *Idempotent algebras with restrictions on subalgebras*, Acta Sci. Math. (Szeged), 51 (1987), pp. 57–65.

[50] P. van Hentenryck, Y. Deville, and C.-M. Teng, *A generic arc-consistency algorithm and its specializations*, Artificial Intelligence, 57 (1992), pp. 291–321.

[51] G. Woeginger, *An efficient algorithm for a class of constraint satisfaction problems*, Oper. Res. Lett., 30 (2002), pp. 9–16.

[52] D. Zuckerman, *Linear degree extractors and the inapproximability of max clique and chromatic number.*, in Proceedings of the 38th ACM Symposium on Theory of Computing (STOC 2006), 2006, pp. 681–690.