Approximability of Clausal Constraints*

Peter Jonsson**
Department of Computer and Information Science
Linköpings Universitet
SE-581 83 Linköping, Sweden
petej@ida.liu.se

Gustav Nordh***
LIX
École Polytechnique
F-91128 Palaiseau CEDEX, France
nordh@lix.polytechnique.fr

Abstract. We study a family of problems, called Maximum Solution (Max Sol), 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 complexity and approximability is completely understood for all restrictions on the underlying constraints. We continue this line of research by considering the Max Sol problem for relations defined by regular signed logic over finite subsets of the natural numbers; the complexity of the corresponding decision problem has recently been classified by Creignou et al. [Theory of Computing Systems, 42(2):239–255, 2008]. We give sufficient conditions for when such problems are polynomial-time solvable and we prove that they are **APX**-hard otherwise. Similar dichotomies are also obtained for variants of the Max Sol problem.

1 Introduction

Our starting-point is the 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

* A preliminary version of this paper appears in *Proceedings of the 31st International Symposium on Mathematical Foundations of Computer Science* (MFCS'06), Lecture Notes in Computer Science, Vol. 4162, Springer, Berlin, 2006, pp. 549-560.

** Communicating author. Partially supported by the *Center for Industrial Information Technology* (CENIIT) under grant 04.01, and by the *Swedish Research Council* (VR) under grant 621-2003-3421.

*** Partially supported by the *Swedish-French Foundation* and the *National Graduate School in Computer Science* (CUGS), Sweden

all constraints while maximising 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 such as Max Independent Set, Max k-Hypergraph Independent Set, and many variants of Max 0/1 Programming. The approximability (and thus the computational complexity) is known for all choices of Γ [26]. 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 [26] where it is also proved that the approximability for the weighted and unweighted versions of the problem coincide.

In this article, we will consider Max Ones generalised to non-Boolean domains which has also recently been studied in [24, 25, 28]. Let the *domain* be a finite set $D = \{0, 1, \ldots, d\}$ equipped with the total order $0 < 1 < \ldots < d$. For pedagogical reasons, we sometimes denote the largest element in the domain by $\max D$, although we reserve d for the largest element in the domain throughout the paper. 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 D is a finite set $\Gamma \subseteq R_D$. Given a relation R, we let $ar(R)$ denote the arity of R. 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 domain, and C is a set of constraints $\{C_1, \ldots, C_q\}$, in which each constraint C_i is a pair (ϱ_i, s_i) with s_i a list of variables of length m_i, called the constraint scope, and ϱ_i 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. The optimisation problem we are going to study, Max Sol, can then be defined as follows:

Definition 1. Weighted Maximum Solution *over the constraint language Γ, denoted* Max Sol(Γ)*, is defined to be the optimisation problem with*

Instance: *Tuple (V, D, C, w), where (V, D, C) is a* Csp *instance over Γ, 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: $\sum_{v \in V} w(v) \cdot f(v)$

For instance, this framework enables the study of certain problems in integer linear programming [18], equation solving over Abelian groups [28], and generalisations of the maximum independent set problem [25]. We restrict ourselves to constraint languages that are definable in *regular signed logic* [20], This logic provides us with convenient concepts for defining a class of relations with strong

modelling capabilities: Jeavons and Cooper [22] have proved that any constraint can be expressed as the conjunction of expressions over this class of relations. A disadvantage with their approach is that the resulting set of constraints may by exponentially large (in the number of tuples in the constraint to be expressed). An improved algorithm solving the same problem has been suggested by Gil et al. [15]. It takes a constraint/relation represented by the set of all assignments/tuples that satisfies it and outputs in polynomial time (in the number of tuples) an expression that is equivalent to the original constraint. The complexity of reasoning within this class of logically defined relations has been considered before in, for example, [10, 22]. One may also note that this class of relations is an instance of disjunctive constraints in the sense of Cohen et al. [9]. However, optimisation within this framework has not been considered earlier.

Let Γ be a finite set of relations definable in regular signed logic. Our aim is to classify the complexity of Max Sol problem when the constraints are restricted to relations in Γ. Thus, we parameterise our problems according to the allowed relations and we denote the restricted problem Max Sol(Γ). We prove the following dichotomy result:

Max Sol(Γ) is either in **PO** or **APX**-hard.

When a problem is **APX**-hard, then there is a constant c such that the problems cannot be approximated in polynomial time within $c - \varepsilon$ for any $\varepsilon > 0$ unless **P**=**NP**. A direct consequence is that these problems do not admit polynomial-time approximation schemes. We also consider the analogous minimisation problem Min Sol and the problem Max AW Sol (where the weights are not assumed to be non-negative); similar dichotomies are obtained for these problems. This kind of dichotomy results are important in computational complexity since they can be seen as exceptions to Ladner's [29] result; he proved that there exists an infinite hierarchy of increasingly difficult problems between **P** and the **NP**-complete problems. Thus, the existence of a complexity dichotomy for a class of problems cannot be taken for granted.

The article is structured as follows: Section 2 contains the basic definitions and results. The results for Max Sol, Min Sol, and Max AW Sol are presented in Sections 3–5, respectively. Finally, some concluding remarks are given in Section 6.

2 Preliminaries

This section contains four parts: in the first, we present regular signed logic and demonstrate how it can be used for defining constraint languages. The second part contains some basic definitions and facts concerning approximation problems and approximability. The final two parts contain material on the algebraic method for studying constraint satisfaction problems.

2.1 Logically defined constraint languages

We will now briefly introduce regular signed logic. Let V be a set of variables. For $x \in V$ and $a \in D$, the inequalities $x \geq a$ and $x \leq a$ are called positive and negative literals, respectively. A *clause* is a disjunction of literals. A *clausal pattern* is a multiset of the form $P = (+a_1, \ldots, +a_p, -b_1, \ldots, -b_q)$ where $p, q \in \mathbb{N}$ and $a_i, b_i \in D$ for all i. The pattern P is said to be *negative* if $p = 0$ and *positive* if $q = 0$. The sum $p + q$, also denoted $|P|$, is the *length* of the pattern.

A *clausal language* L is a set of clausal patterns. Given a clausal language L, an L-*clause* is a pair $(P, \mathbf{x})$, where $P \in L$ is a pattern and $\mathbf{x}$ is a vector of not necessarily distinct variables from V such that $|P| = |\mathbf{x}|$. A pair $(P, \mathbf{x})$ with a pattern $P = (+a_1, \ldots, +a_p, -b_1, \ldots, -b_q)$ and variables $\mathbf{x} = (x_1, \ldots, x_{p+q})$ represents the clause

$$(x_1 \geq a_1 \vee \ldots \vee x_p \geq a_p \vee x_{p+1} \leq b_1 \vee \ldots \vee x_{p+q} \leq b_q),$$

where $\vee$ is the disjunction operator. An L-formula φ is a conjunction of a finite number of L-clauses. An *assignment* is a mapping $I : V \to D$ assigning a domain element $I(x)$ to each variable $x \in V$ and I *satisfies* φ if and only if

$$(I(x_1) \geq a_1 \vee \ldots \vee I(x_p) \geq a_p \vee I(x_{p+1}) \leq b_1 \vee \ldots \vee I(x_{p+q}) \leq b_q)$$

holds for every clause in φ. It can be easily seen that the literals $+0$ and $-d$ are superfluous since the inequalities $x \geq 0$ and $x \leq d$ vacuously hold. Without loss of generality, it is sufficient to only consider patterns and clausal languages without such literals. We see that clausal patterns are nothing more than a convenient way of specifying certain relations — consequently, we can use them for defining constraint languages. Thus, we make the following definitions: given a clausal language L and a clausal pattern $P = (+a_1, \ldots, +a_p, -b_1, \ldots, -b_q)$, we let $Rel(P)$ denote the corresponding relation, i.e. $Rel(P) = \{\mathbf{x} \in D^{p+q} \mid (P, \mathbf{x}) \text{ hold}\}$ and $\Gamma_L = \{Rel(P) \mid P \in L\}$.

It is easy to see that several well-studied optimisation problems are captured by this framework.

Example 2. Let the domain D be $\{0, 1\}$. The problem Independent Set (where the objective is to find an independent set of maximum weight in an undirected graph) can be viewed as the Max Sol($\Gamma_{(-0,-0)}$) problem. Similarly, Max Sol($\Gamma_{\{(-0,\ldots,-0)\}}$) (with k literals) is the Max k-Hypergraph Independent Set problem while Min Sol($\Gamma_{\{(+1,+1)\}}$) is the Minimum Vertex Cover problem.

2.2 Approximability and reductions

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 integer *measure* of s. Note that the measure is often required to be a positive integer (cf. [2]) but this is not possible in this article since we

are considering problems that may have negative optimal measure. Such problems instances will only occur in connection with the MAX AW SOL problem in Section 5, though.

The objective given an instance I is 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. [3]).

We say that a solution $s \in \mathsf{sol}(I)$ to an instance I of an **NPO** problem Π is *r-approximate*, $r \geq 1$, if it is satisfying

$$\frac{\text{OPT}(I)}{r} \leq m(I, s) \leq r \cdot \text{OPT}(I).$$

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. Hardness in **APX** is defined using an appropriate reduction, called AP-reduction [11, 26].

Definition 3. *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 *if every problem in* **APX** *is AP-reducible to it. If, in addition, Π is in* **APX**, *then Π is called* **APX**-complete.

It is a well-known fact (cf. Section 8.2.1 in [3]) that AP-reductions compose. In some proofs, we will use another kind of reduction, S-reductions, defined as follows:

Definition 4. *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 $m(I, G(s')) = m'(I', s')$.

Obviously, the existence of an S-reduction from Π_1 to Π_2 implies 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** [26]. We also note that S-reductions 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 third kind of approximation-preserving reduction known as L-reduction [3]:

Definition 5. *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 [3]) 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. This explains why some of the forthcoming technical results are **APX**-completeness results and not (only) **APX**-hardness results.

We say that an optimisation problem Π admits a *polynomial-time approximation scheme* (PTAS) if there exists an algorithm A satisfying the following property: for any instance I of Π and any rational value $r > 1$, $A(I, r)$ returns an r-approximate solution of I in time polynomial in $|I|$. It is well-known (cf. [2, Corr. 3.13]) that if $\mathbf{P} \neq \mathbf{NP}$, then no **APX**-hard problem admits a polynomial-time approximation scheme.

2.3 Algebraic framework

Our results are to a certain extent based on recent algebraic methods for studying constraint satisfaction problems. The use of algebraic techniques for studying such problems has made it possible to clarify the borderline between polynomial-time solvable and intractable cases. Both our tractability and hardness results exploit algebraic techniques — typically, we prove a restricted base case and then extend the result to its full generality via algebraic techniques. To this end, we introduce (in the next section) the concept of *max-cores* (which is a variant of the algebraic and graph-theoretic concept *core*) and show some of its properties.

An operation on D is an arbitrary function $f : D^k \to D$. Any operation on D can be extended in a standard way to an operation on tuples over D, as follows:

Let f be a k-ary operation on D 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:

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

where $\boldsymbol{t_j}[i]$ is the i-th component in tuple $\boldsymbol{t_j}$. If f is an operation such that for all $\boldsymbol{t_1}, \boldsymbol{t_2}, \ldots, \boldsymbol{t_k} \in R$, $f(\boldsymbol{t_1}, \boldsymbol{t_2}, \ldots, \boldsymbol{t_k}) \in R$, then R is said to be *invariant* (or *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 is invariant under all the operations in F is denoted $Inv(F)$. Sets of operations of the form $\mathrm{Pol}(\Gamma)$ are known as *clones*, and they are well-studied objects in algebra (cf. [33]). We remark that the operators Inv and Pol form a Galois correspondence between the set of relations over D and the set of operations on D. A basic introduction to this correspondence can be found in [32], and a comprehensive study in [33].

A first-order formula φ over a constraint language Γ is said to be *primitive positive* (or *pp-formula* for short) if it is of the form

$$\exists \mathbf{x} : (R_1(\mathbf{x}_1) \wedge \ldots \wedge R_k(\mathbf{x}_k))$$

where $R_1, \ldots, R_k \in \Gamma$ and $\mathbf{x}, \mathbf{x}_1, \ldots, \mathbf{x}_k$ are vectors of variables such that $ar(R_i) = |\mathbf{x}_i|$ for all i and all variables in $\mathbf{x}$ appear in $\mathbf{x}_1, \ldots, \mathbf{x}_k$. Note that a pp-formula φ with m free variables defines an m-ary relation $R \subseteq D^m$, denoted $R \equiv_{pp} \varphi$; the relation R is the set of all m-tuples satisfying the formula φ.

We define a closure operation $\langle\cdot\rangle$ such that $R \in \langle\Gamma\rangle$ if and only if the relation R can be obtained from $\Gamma \cup \{=_D\}$ by pp-formulas, where $=_D$ is the equivalence relation on the domain D. If Γ is a constraint language over a finite domain, then $\langle\Gamma\rangle = Inv(Pol(\Gamma))$ [33].

The following lemma from [24] states that Max Sol over finite subsets of $\langle\Gamma\rangle$ is no harder than Max Sol over Γ itself.

Lemma 6 ([24]). *Let Γ' be a finite constraint language such that $\Gamma' \subseteq \langle\Gamma\rangle$. If* Max Sol($\Gamma$) *is in* **PO**, *then* Max Sol(Γ') *is in* **PO** *and if* Max Sol(Γ') *is* **APX**-*hard then* Max Sol(Γ) *is* **APX**-*hard.*

The lemma above also holds also for Min Sol and Max AW Sol. Also note the following consequence: if Γ and Γ' are finite constraint languages such that $\langle\Gamma'\rangle = \langle\Gamma\rangle$, then Max Sol($\Gamma$) is **APX**-hard (in **PO**) if and only if Max Sol(Γ') is **APX**-hard (in **PO**).

The next lemma simplifies some of the forthcoming proofs and its proof is an easy consequence of the preceding lemma.

Lemma 7. *Let* $P = (+a_1, +a_2, \ldots, +a_p, -b_1, \ldots, -b_q)$ *and*

$$P_1 = (+a_1, + \min\{a_2, \ldots, a_p\}, -b_1, \ldots, -b_q).$$

Then, **APX**-*hardness of* Max Sol($\Gamma_{\{P_1\}}$) *implies the* **APX**-*hardness of* Max Sol($\Gamma_{\{P\}}$). *Similarly, if*

$$P_2 = (+a_1, \ldots, +a_p, -b_1, - \max\{b_2, \ldots, b_q\}),$$

then **APX**-*hardness of* Max Sol($\Gamma_{\{P_2\}}$) *implies* **APX**-*hardness of* Max Sol($\Gamma_{\{P\}}$). *The same results also hold for* Min Sol *and* Max AW Sol.

Proof. We see that $(x \geq a_1 \vee y \geq \min(a_2, \ldots, a_p) \vee z_1 \leq b_1 \vee \ldots \vee z_q \leq b_q) \equiv_{pp}$ $(x \geq a_1 \vee y \geq a_2 \vee \ldots \vee y \geq a_p \vee z_1 \leq b_1 \vee \ldots z_q \leq b_q)$ so if Max Sol($\Gamma_{\{P_1\}}$) is **APX**-hard then, by Lemma 6, so is Max Sol($\Gamma_{\{P\}}$). The other cases can be proved analogously. □

2.4 Max-cores

The concept of a core of a constraint language Γ has previously shown its value when classifying the complexity of CSP(Γ). For instance, it is known that a complete classification of the core languages implies a complete classification of all constraint languages [6, 23].

Definition 8. *An* endomorphism *of Γ is a unary operation $f : D \to D$ such that, for all $R \in \Gamma$ and all $(a_1, \ldots, a_m) \in D^m$:*

$$(a_1, \ldots, a_m) \in R \Rightarrow (f(a_1), \ldots, f(a_m)) \in R.$$

We say that Γ is a core *if every endomorphism of Γ is injective (i.e. a permutation).*

Obviously, the endomorphisms of Γ are the unary operations in $Pol(\Gamma)$. In order to gain some intuition, assume that Γ is not a core. Then, Γ has a non-injective endomorphism f, which implies that, for every assignment φ, there is another assignment $f\varphi$ that satisfies all constraints satisfied by φ. Furthermore, $f\varphi$ uses only a restricted set of values so CSP(Γ) is equivalent to CSP($\Gamma|_{D'}$) where $D' \subset D$. It is known that all cores of $\mathcal{F}$ are isomorphic, so one can speak about *the* core of $\mathcal{F}$.

We define a related concept (called *max-core*) for the Max Sol problem. For a relation $R = \{(d_{11}, \ldots, d_{1m}), \ldots, (d_{t1}, \ldots, d_{tm})\}$ and a unary operation f, let $f(R)$ denote the relation

$$\{(f(d_{11}), \ldots, f(d_{1m})), \ldots, (f(d_{t1}), \ldots, f(d_{tm}))\}.$$

Similarly, let $f(\Gamma)$ denote the constraint language $\{f(R) \mid R \in \Gamma\}$. We will now turn our attention to increasing endomorphisms, i.e. endomorphisms that satisfy $f(d) \geq d$ for all $d \in D$.

Lemma 9. *Let Γ be a finite constraint language over D and f an increasing endomorphism. The following holds:* Max Sol(Γ) *is in* **PO** *(***APX***-hard) if and only if* Max Sol($f(\Gamma)$) *is in* **PO** *(***APX***-hard).*

Proof. We begin by showing that Max Sol($f(\Gamma)$) is *AP*-reducible to Max Sol(Γ). Given an instance $I = (V, D, C, w)$ of Max Sol($f(\Gamma)$), let $F(I) = (V, D', C', w)$ be the instance of Max Sol(Γ) where every constraint relation $f(R_i)$ occurring in a constraint $C_i \in C$ has been replaced by R_i. Given a solution

s' of $F(I)$, let $G(I, s')$ be the solution s of I where $s(x) = f(s'(x))$ for each variable x; note that $m(I, G(I, s')) \geq m(I', s')$ since f is increasing. That f is increasing also implies that $\text{OPT}(I) \leq \text{OPT}(F(I))$. However, $f \in Pol(\Gamma)$ so $\text{OPT}(I) \geq \text{OPT}(F(I))$ and $\text{OPT}(I) = \text{OPT}(F(I))$. Consequently,

$$\frac{\text{OPT}(I)}{m(G(I, s')} \leq \frac{\text{OPT}(F(I))}{m(s')}$$

and we have an AP-reduction with parameter $\alpha = 1$, mapping optimal solutions to optimal solutions, which preserve membership in **PO**.

To prove the other direction, we show that there is a S-reduction from MAX SOL(Γ) to MAX SOL($f(\Gamma)$). Given an instance $I = (V, D, C, w)$ of MAX SOL(Γ), let $F(I) = (V, D', C', w)$ be the instance of MAX SOL($f(\Gamma)$) where every constraint relation R_i occurring in a constraint $C_i \in C$ has been replaced by $f(R_i)$. By an argument similar to the one above, we see that $\text{OPT}(F(I)) = \text{OPT}(I)$. Given a solution s' of $F(I)$, let $G(I, s') = s'$ and note that this is a solution (of the same measure) to I since $f \in Pol(\Gamma)$. □

We define max-cores to be constraint languages that do not satisfy the preconditions of the previous lemma.

Definition 10. *A constraint language Γ is a max-core if and only if Γ has no non-injective increasing endomorphism. A constraint language Γ' is a max-core of Γ if and only if Γ' is a max-core and $\Gamma' = f(\Gamma)$ for some increasing endomorphism f.*

The identity function $\mathbf{id}_D$ (which is increasing) is always an endomorphism so we only consider non-injective operations in the previous definition. We continue to show that it is often sufficient to consider max-cores when studying the approximability and complexity of MAX SOL

Lemma 11. *Assume that Γ' is a max-core of Γ. Then,* MAX SOL(Γ) *is in* **PO** *(***APX***-hard) if and only if* MAX SOL(Γ') *is in* **PO** *(***APX***-hard).*

Proof. By the definition of a max-core we have that $\Gamma' = f(\Gamma)$ for some increasing endomorphism f. Hence, the result follows directly from Lemma 9. □

It is well-known that the cores of a constraint language Γ are isomorphic. We prove something similar for max-cores: every constraint language has a unique max-core.

Lemma 12. *If f and g are endomorphisms of Γ and $f(\Gamma) = \Gamma'$, then $f \circ g$ is an endomorphism of Γ'.*

Proof. Assume a tuple $(a, b, \ldots, c) \in R'$ such that $(f(g(a)), f(g(b)), \ldots, f(g(c))) \notin R'$ where $R' \in \Gamma'$ and $f(R) = R'$ for some $R \in \Gamma$. From $(a, b, \ldots, c) \in R'$ and $g, f \in Pol(\Gamma)$, we get that $(g(a), g(b), \ldots, g(c)) \in R$, and by $f(R) = R'$ we get that $(f(g(a)), f(g(b)), \ldots, f(g(c))) \in R'$ which is a contradiction. □

Let $Im(f) = \{f(d) \mid d \in D\}$ for any function $f : D \to D$.

Lemma 13. *Let f be an increasing unary operation witnessing the fact that Γ' is a max-core of Γ, i.e. Γ' is a max-core and $f(\Gamma) = \Gamma'$. Then, $f(f(d)) = f(d)$ for all $d \in D$ i.e. $f(d) = d$ for all $d \in Im(f)$.*

Proof. We let $f^2(x)$ denote the unary operation $f(f(x))$. Obviously, $Im(f^2) \subseteq Im(f)$. Assume, with the aim of reaching a contradiction, that $f^2(d) \neq f(d)$ for some $d \in D$. Moreover, assume that $f(d)$ is the minimum element in D such that $f^2(d) > f(d)$. Since f is increasing this implies that $f(d) \notin Im(f^2)$, and $Im(f^2) \subset Im(f)$. Note that, by Lemma 12, $f \in Pol(\Gamma)$ and $f(\Gamma) = \Gamma'$ so $f^2 \in Pol(\Gamma')$. This together with $Im(f^2) \subset Im(f)$ contradicts that Γ' is a max-core and, consequently, we have that $f(d) = d$ for all $d \in Im(f)$. □

Theorem 14. *If Γ_1 and Γ_2 are max-cores of Γ, then $\Gamma_1 = \Gamma_2$. In other words, each constraint language has a unique max-core.*

Proof. Assume to the contrary that there is a constraint language Γ having at least two different max-cores, i.e. Γ has two increasing endomorphisms f, g such that $f(\Gamma) = \Gamma_1$, $g(\Gamma) = \Gamma_2$, Γ_1 and Γ_2 are max-cores and $\Gamma_1 \neq \Gamma_2$.

We begin by showing that $Im(f) = Im(g)$. Assume to the contrary that $Im(f) \neq Im(g)$ and that d is the least element such that $d \in Im(f) \cup Im(g)$ but $d \notin Im(f) \cap Im(g)$. Without loss of generality assume that $d \in Im(g)$ but $d \notin Im(f)$. Consider the unary operation $h(x) = g(f(x))$. Obviously, $Im(h) \subseteq Im(g)$ and since d is the least element such that $d \in Im(g)$ but $d \notin Im(f)$, we see that $d \notin Im(h)$ and $Im(h) \subset Im(g)$. By Lemma 12, $h \in Pol(\Gamma_2)$ and this contradicts that Γ_2 is a max-core.

Let S denote the image of f and g. By Lemma 13, we know that $f(d) = g(d) = d$ for all $d \in S$. Obviously, no element $d' \in D$ which is not in S appears in $f(\Gamma) = \Gamma_1$ or $g(\Gamma) = \Gamma_2$. We know that $\Gamma_1 \neq \Gamma_2$ so we assume (without loss of generality) that there is a relation $R_1 \in \Gamma_1$ such that $R_1 \notin \Gamma_2$. Consequently, there is a relation $R \in \Gamma$ such that $f(R) = R_1$ but $g(R) = R_2 \neq R_1$. Given a tuple $(a, b, \ldots, c) \in R$ where $\{a, b, \ldots, c\} \subseteq S$, then

$$(f(a), f(b), \ldots, f(c)) = (a, b, \ldots, c) = (g(a), g(b), \ldots, g(c)).$$

Thus, R_1 and R_2 contain all such tuples. Any tuple t in R containing at least one element which is not in S will by definition neither appear in R_1 nor R_2. Hence, $R_1 = R_2$ and we have a contradiction. Thus, we have proved that $\Gamma_1 = \Gamma_2$ and every constraint language has a unique max-core. □

3 Approximability of Max Sol

In this section, we present sufficient conditions for when Max Sol is tractable and prove that it is **APX**-hard otherwise. To do so, we use a family of operations $\max_u : D^2 \to D$, $u \in D$, defined such that

$$\max_u(a, b) = \begin{cases} u & \text{if } \max(a, b) \leq u \\ \max(a, b) & \text{otherwise} \end{cases}$$

Theorem 15. MAX SOL(Γ_L) *is tractable if* Γ_L *is invariant under* $\max_u$ *for some* $u \in D$*. Otherwise,* MAX SOL(Γ_L) *is* **APX***-hard.*

We divide the proof into three parts which can be found in Sections 3.1-3.3.

3.1 Tractability result

The tractability of MAX SOL(Γ_L) when Γ_L is invariant under $\max_u$ for some $u \in D$ follows from the fact that $\max_u$-closed constraint languages are a special case of the more general tractable class of generalised max-closed constraint languages identified in [24].

Lemma 16 ([24]). *If* Γ_L *is invariant under* $\max_u$ *for some* $u \in D$*, then* MAX SOL(Γ_L) *is in* **PO***.*

Also note that the $\max_u$-closed constraint languages are already known to be tractable for the CSP problem. If $\max_u \in Pol(\Gamma)$, then it is easy to see that the core of Γ is invariant under the max operation and, hence, tractability for CSP(Γ) follows from [22].

3.2 APX-hardness results

The ultimate goal of this section is to prove that MAX SOL($Rel(P)$) is **APX**-hard whenever P is a negative pattern containing at least two literals. In the sequel, we will frequently encounter MAX SOL problems restricted to instances where each variable may occur at most k times. Such problems are denoted MAX SOL(Γ)-k.

We begin by proving an **APX**-hardness result for the pattern $(-0, -1)$ over the domain $D = \{0, 1, 2\}$. The reduction is based on the well known **APX**-complete maximisation problem MAX-E3SAT-5 [13]:

Instance: Set U of variables, collection C of disjunctive clauses containing exactly 3 literals each, and where each variable occurs at most 5 times.
Solution: A truth assignment for U.
Measure: Number of clauses satisfied by the truth assignment.

Note that we prove **APX**-completeness in the next lemma: the reason for this is that an L-reduction will be made from MAX SOL(R)-11 in Lemma 19 and membership in **APX** ensures that this reduction can be transformed into an AP-reduction.

Lemma 17. *Let* $D = \{0, 1, 2\}$ *and* $R = \{(x, y) \in D^2 \mid x \leq 0 \vee y \leq 1\}$*, Then,* MAX SOL($R$)-11 *is* **APX***-complete even if all variables have weight* 1*.*

Proof. Membership in **APX** follows from the fact that the all-1 assignment is a 2-approximation. We prove **APX**-hardness by giving an L-reduction (with $\beta = 14$ and $\gamma = 1$) from MAX-E3SAT-5 to MAX SOL(R). The reduction relies

on the following 'gadget': Let $V = \{A, B, C, a, b, c\}$ be a set of variables and impose the following constraints:

$$R(A, B), R(B, C), R(C, A), R(A, a), R(B, b), R(C, c).$$

One can see that the optimum solution to this gadget is 7 and this optimum appears if exactly one of A, B, C is assigned the value 2.

Let I be an arbitrary MAX-E3SAT-5 instance with m clauses $C_1, \ldots, C_m$. Construct a MAX SOL(R) instance $F(I) = (X, D, C, w)$ as follows:

$$X = \{X_1^1, X_2^1, X_3^1, x_1^1, x_2^1, x_3^1, \ldots, X_1^m, X_2^m, X_3^m, x_1^m, x_2^m, x_3^m\},$$

$w(x) = 1$ for all $x \in X$, and introduce a gadget on $X_1^i, X_2^i, X_3^i, x_1^i, x_2^i, x_3^i$ (as defined above) for each clause $C_i = \{l_1^i, l_2^i, l_3^i\}$. Finally, the clauses are connected by adding the constraints $R(X_j^i, X_{j'}^{i'})$ and $R(X_{j'}^{i'}, X_j^i)$ whenever $l_j^i = \neg l_{j'}^{i'}$.

By well-known arguments, at least half of the clauses in an instance of MAX-E3SAT-5 can be satisfied so $m \leq 2\text{OPT}(I)$. We also know that $\text{OPT}(F(I)) \leq 7m$ since each gadget corresponding to a clause contributes at most 7 to the measure of any solution to $F(I)$. It follows that $\text{OPT}(F(I)) \leq 14 \cdot \text{OPT}(I)$ and we can choose $\beta = 14$.

Now, given $F(I)$ and a solution s to $F(I)$, let $s' = G(F(I), s)$ be the solution to I (the instance of MAX-3SAT) defined as follows: $s'(x) = \mathit{true}$ if there exists a literal $l_j^i = x$ and $s(X_j^i) = 2$, $s'(x) = \mathit{false}$ if there exists a literal $l_j^i = \neg x$ and $s(X_j^i) = 2$, and $s'(x) = \mathit{false}$ for all other variables x. First we note that $s'(x)$ is well-defined; any two contradictory literals are prevented from being assigned the same truth value by the constraints introduced in the last step in the construction of $F(I)$.

We will show that $\text{OPT}(I) - m(I, s') \leq \text{OPT}(F(I)) - m(F(I), s)$ and $\gamma = 1$ is a valid parameter in the L-reduction. We begin by showing that $\text{OPT}(F(I)) - \text{OPT}(I) \geq 6m$. If $\text{OPT}(I) = k$, i.e. k clauses (but no more) can be satisfied, then each of the k satisfied clauses contains a true literal l_j^i. In each of the satisfied clauses C_i we choose one true literal (say l_j^i) and assign 2 to the corresponding variable X_j^i in the corresponding gadget G_i (on variables $\{X_1^i, X_2^i, X_3^i, x_1^i, x_2^i, x_3^i\}$) in $F(I)$. Assign 1 to x_j^i, 2 to the other two x^i variables, and 0 to the two unassigned X^i variables. In each gadget G_j corresponding to an unsatisfied clause C_j (in $\text{OPT}(I)$), assign 0 to all the X^j variables and 2 to all the x^j variables. The resulting solution to $F(I)$ shows that

$$\text{OPT}(F(I)) \geq 7k + 6(m - k) = k + 6m$$

and $\text{OPT}(F(I)) - \text{OPT}(I) \geq 6m$ since $k = \text{OPT}(I)$. Assume now that

$$\text{OPT}(I) - m(I, s) > \text{OPT}(F(I)) - m(F(I), s')$$

which implies $m(F(I), s') - m(I, s) > 6m$. This leads to a contradiction so $\text{OPT}(I) - m(I, s) \leq \text{OPT}(F(I)) - m(F(I), s')$ and $\gamma = 1$ is a valid parameter in the L-reduction. Also note that no variable occurs more than 11 times in the resulting instance $F(I)$ of MAX SOL(R). □

From now on, let the domain $D = \{0, \ldots, d\}$ be fixed and with $d \geq 2$. By combining the notion of max-cores and Lemma 17, we can prove **APX**-hardness for all negative patterns of length at least two:

Lemma 18. *If* $(-c_1, \ldots, -c_k) \in L$, $k \geq 2$, *then* Max Sol(Γ_L) *is* **APX***-hard.*

The proof of Lemma 18 is based on Lemmata 19 and 20.

Lemma 19. *If* $(-a, -b) \in L$ *where* $a < b$, *then* Max Sol(Γ_L)-11 *is* **APX***-complete even if all variables have weight* 1*.*

Proof. First recall that $a < b < d$. Membership in **APX** follows from the fact that the all-b assignment is a $\frac{d}{b} \leq d$ approximate solution.

For **APX**-hardness, consider the operation f on D defined as follows:

$$f(x) = \begin{cases} a & \text{if } x \leq a, \\ b & \text{if } a < x \leq b \\ d & \text{if } b < x \leq d. \end{cases}$$

It is readily verified that f is an increasing endomorphism, and that $R' = f(Rel(-a, -b))$ is a max-core. More specifically,

$$R' = \{(a,a), (a,b), (b,a), (b,b), (a,d), (d,a), (d,b)\}.$$

We know from Lemma 11 that Max Sol(R) is **APX**-hard if Max Sol(R') is **APX**-complete. We give an L-reduction (with parameters $\beta = d$ and $\gamma = 1$) from the **APX**-complete problem Max Sol(R)-11, where

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

is the relation in Lemma 17, to Max Sol(R').

Given an instance I of Max Sol(R), let $F(I)$ be the instance of Max Sol(R') where all occurrences of R 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 a are instead assigned 0, all variables assigned b are assigned 1, and all variables assigned d are assigned 2. We have, $\text{opt}(F(I)) \leq d \cdot \text{opt}(I)$ and

$$\text{opt}(I) - m(I, G(I, s')) \leq \text{opt}(F(I)) - m(F(I), s'),$$

so $\beta = d$ and $\gamma = 1$ are valid parameters in the L-reduction. Thus, Max Sol(Γ_L) is **APX**-hard when $(-a, -b) \in L$ and $a < b$. □

We prove the next lemma by a reduction from the **APX**-complete problem independent set restricted to graphs of maximum degree 3 [1].

Instance: Undirected graph $G = (V, E)$ with maximum degree 3.
Solution: A set $V' \subseteq V$ such that for all $v, w \in V'$, $(v, w) \notin E$.
Measure: Cardinality of V'.

We may additionally and without loss of generality assume that the graphs under consideration do not contain any isolated vertices.

Lemma 20. Max Sol($\Gamma_{\{(-a,-a)\}}$)-3 *is* **APX**-*complete even if all variables have weight* 1.

Proof. We begin by showing membership in **APX**. Let $I = (V, D, C, w)$ be an arbitrary instance of Max Sol($\Gamma_{\{(-a,-a)\}}$)-3 such that all weights equal 1 and let $d = \max D$. If $a > 0$, then the all-1 assignment is a d-approximation of I. If $a = 0$, then construct a graph (V, E) where $(v, v') \in E$ if and only if $(Rel((-0, -0), (v, v'))) \in C$. Clearly, $\text{opt}(I) = d \cdot |M|$ where $M \subseteq V$ is an independent set in G of maximum size. Since the graph G is of maximum degree 3, the independent set problem can be approximated within 4/3 in polynomial time [31, Theorem 13.7] and Max Sol($\Gamma_{\{(-a,-a)\}}$)-3 can be approximated within $4d/3$.

We continue by showing **APX**-hardness: consider the following operation f on D:

$$f(x) = \begin{cases} a & \text{if } x \leq a, \\ d & \text{otherwise.} \end{cases}$$

It can be seen that f is an increasing endomorphism, and that $R' = f(R) = \{(a, a), (a, d), (d, a)\}$ is a max-core. We know from Lemma 11 that Max Sol(R) is **APX**-hard if Max Sol(R') is **APX**-complete.

We give an L-reduction (with $\beta = 4b$ and $\gamma = \frac{1}{b-a}$) from the **APX**-complete problem Independent Set-3 to Max Sol(R'). Given an instance $I = (V, E)$ of Independent Set-3, let $F(I) = (V, D, C, w)$ 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 and let all variables have weight 1. For any feasible solution s' for $F(I)$, let $G(I, s') = \{v \in V \mid s'(v) = b\}$ and observe that $G(I, s')$ is an independent set in the graph (V, E). 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,

$$|\text{opt}(I) - m(I, G(I, s'))| = \text{opt}(I) - K$$

and

$$|\text{opt}(F(I)) - m(F(I), s')| = (b - a)(\text{opt}(I) - K)$$

so

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

and $\gamma = \frac{1}{b-a}$ is an appropriate parameter. Thus, Max Sol(R') is **APX**-hard. □

By combining Lemma 7 with either Lemma 19 or Lemma 20, we see that all negative patterns of length at least 2 are **APX**-hard and Lemma 18 is proved.

3.3 Proof of Theorem 15

Proof. Arbitrarily choose a clausal language L. If a pattern $(-c_1, -c_2, \ldots, -c_k)$, $k \geq 2$, exists in L, then MAX SOL(Γ_L) is **APX**-hard by Lemma 18. Hence, we can assume that for each $P \in L$ such that $|P| \geq 2$, it holds that P contains at least one positive literal. If all patterns in L are of length 1, then MAX SOL(Γ_L) is tractable since Γ_L is invariant under the operation max. Thus, we assume that L contains at least one pattern of length strictly greater than one. Let

$$u = \min\{\max U \mid U \subseteq D \text{ is definable by a pp-formula over } \Gamma_L, \}$$

where $\max U$ denotes the largest element in U. Let ψ be a pp-formula over Γ_L defining the set U, i.e. $U(x) \equiv_{pp} \exists \mathbf{x} : \psi(x, \mathbf{x})$ and $\max U = u$. If there exists a pattern $(+a_1, \ldots, +a_p, -b_1, \ldots, -b_q)$, $q \geq 2$, and $a_i > u$ for all i, then there is a pp-formula that implements the relation $Rel((-b_1, \ldots, -b_q))$:

$$(y_1 \leq b_1 \vee \ldots \vee y_q \leq b_q) \equiv_{pp}$$

$$\exists z : (z \geq a_1 \vee \ldots \vee z \geq a_p \vee y_1 \leq b_1 \vee \ldots \vee y_q \leq b_q) \wedge\ U(z)$$

so MAX SOL(Γ_L) is **APX**-hard by Lemmata 6 and 18. If this is not the case, then we show that Γ_L is invariant under $\max_u$ and, by Lemma 16, that MAX SOL(Γ_L) is tractable. Arbitrarily choose a pattern $P \in L$. If $|P| \geq 2$, then we have two cases: Assume first that $P = (+a_1, \ldots)$ for some $a_1 \leq u$. Since $\max_u(a, b) \geq u$ for all choices of a, b, $Rel(P)$ is invariant under $\max_u$. Otherwise, $P = (+a_1, \ldots, +a_p, -b_1)$ and $a_i > u$ for all i. We see that $b_1 \geq u$ by the definition of u since $Rel((-b_1))$ can be implemented by a pp-formula:

$$(y_1 \leq b_1) \equiv_{pp} \exists z : (z \geq a_1 \vee \ldots \vee z \geq a_p \vee y_1 \leq b_1) \wedge U(z)$$

Arbitrarily choose two tuples $(t_1, \ldots, t_{p+1})$, $(t'_1, \ldots, t'_{p+1})$ from $Rel(P)$. If there exists a t_i, $1 \leq i \leq p$, such that $t_i \geq a_i$, then the tuple

$$(\max_u(t_1, t'_1), \ldots, \max_u(t_{p+1}, t'_{p+1}))$$

is in $Rel(P)$. The situation is analogous if there exists a t'_i, $1 \leq i \leq p$, such that $t'_i \geq a_i$. Assume now that for all $1 \leq i \leq p$, $t_i < a_i$ and $t'_i < a_i$. This implies that $t_{p+1} \leq b_1$ and $t'_{p+1} \leq b_1$. If $\max(t_{p+1}, t'_{p+1}) \leq u$, then $\max_u(t_{p+1}, t'_{p+1}) = u$ and $Rel(P)$ is invariant under $\max_u$ since $b_1 \geq u$. If $\max(t_{p+1}, t'_{p+1}) > u$, then $\max_u(t_{p+1}, t'_{p+1}) = \max(t_{p+1}, t'_{p+1})$ and $Rel(P)$ is invariant under $\max_u$ also in this case.

We are left with the unary patterns in L. Assume that $P = (+r)$ for some r; in this case, $Rel(P)$ is trivially invariant under $\max_u$. If $P = (-r)$, then r must satisfy $r \geq u$ by the definition of u. Arbitrarily choose two elements $a, b \in Rel((-r))$. If $\max(a, b) \leq u$, then $\max_u(a, b) = u$ and $Rel(P)$ is invariant under $\max_u$ since $r \geq u$. If $\max(a, b) > u$, then $\max_u(a, b) = \max(a, b)$ and $Rel(P)$ is invariant under $\max_u$. □

By inspecting the previous proof, we see that a constraint language Γ_L is invariant under $\max_u$, $u \in D$, if and only if each pattern $P \in L$ satisfy at least one of the following conditions:

1. $P = (+a_1, ...)$ and $a_1 \leq u$;
2. $P = (+a_1, \ldots, +a_p, -b_1)$, $a_1, \ldots, a_p > u$, and $b_1 \geq u$;
3. $P = (+a)$; or
4. $P = (-a)$ and $a \geq u$.

This makes it easy to check whether Max Sol(Γ_L) is tractable or not: test if the condition above holds for some $u \in D$. If so, Max Sol(Γ_L) is tractable and, otherwise, Max Sol(Γ_L) is **APX**-hard by Theorem 15. Obviously, this test can be performed in polynomial time in the size of L and D. A simple algorithm that is polynomial in the size of Γ_L[1] also exists, but note that Γ_L can be exponentially larger than L and D.

4 Approximability of Min Sol

We now turn our attention to the problem Min Sol, i.e. the minimisation version of Max Sol We recall that, for instance, Min Sol($(+1, +1)$) (over the domain $D = \{0, 1\}$) is the same problem as the minimum vertex cover problem:

Instance: Undirected vertex-weighted graph $G = (V, E, w)$.
Solution: A vertex cover for G, i.e. a subset $V' \subseteq V$ such that, for each edge $(u, v) \in E$, at least one of u and v belongs to V'.
Measure: Weight of the vertex cover, i.e. $\sum_{v \in V'} w(v)$.

Obviously, the tractability results for Max Sol can be transferred to the Min Sol setting with only minor modifications: If Γ_L is invariant under $\min_u$ for some $u \in D$, then Min Sol(Γ_L) is in **PO**: we define $\min_u : D^2 \to D$, $u \in D$, such that

$$\min_u(a, b) = \begin{cases} u & \text{if } \min(a, b) \geq u \\ \min(a, b) & \text{otherwise} \end{cases}$$

By combining this with a hardness result for Min Sol($\Gamma_{\{(+b_1,\ldots,+b_q)\}}$), $q \geq 2$, one can prove the following:

Theorem 21. *Min Sol(Γ_L) is in* **PO** *if Γ_L is invariant under* $\min_u$ *for some $u \in D$. Otherwise,* Min Sol(Γ_L) *is* **APX***-hard.*

The structure of the proof of this theorem is similar to the corresponding proof for Max Sol(Γ_L), and thus follows from the following lemmata. Note that it easy to check whether Min Sol(Γ_L) is tractable or not: the algorithm is similar to the algorithm for checking tractability of Max Sol(Γ_L). We proceed with a simple hardness result; the straightforward proof is omitted.

[1] The size of a constraint language Γ over a finite domain D is roughly $\sum_{R \in \Gamma} |R| \cdot \log |D| \cdot ar(R)$.

Lemma 22. *Let* $(P, \mathbf{x})$ *be the constraint*

$$(x_1 \geq a_1 \vee \cdots \vee x_i \geq a_p \vee x_{i+1} \leq b_1 \vee \cdots \vee x_j \leq b_q)$$

and let $(\overline{P}, \mathbf{x})$ *be the constraint*

$$(x_1 \leq d - a_1 \vee \cdots \vee x_i \leq d - a_p \vee x_{i+1} \geq d - b_1 \vee \cdots \vee x_j \geq d - b_q)$$

where $d = \max D$*. Then, an assignment* s *satisfies* $(P, \mathbf{x})$ *if and only if* $(\overline{P}, \mathbf{x})$ *is satisfied by* $\overline{s}$ *where* $\overline{s}(x) = d - s(x)$ *for all* $x \in \mathbf{x}$*.*

In the next proof, we will exploit approximability results for the minimum vertex cover problem.

Lemma 23. Min Sol$(\Gamma_{\{(+a,+b)\}})$-11 *is* **APX***-hard even if all variables have weight* 1*.*

Proof. Let $d = \max D$, $R = Rel((+a, +b))$, and $\overline{R} = Rel((-(d-a), -(d-b))$. We give an L-reduction from Max Sol$(\overline{R})$ to Min Sol(R) with parameters $\beta = 12d$ and $\gamma = 1$. If $a \neq b$, then Max Sol$(\overline{R})$-11 is **APX**-complete by Lemma 19 and if $a = b$, then Max Sol$(\overline{R})$-3 is **APX**-complete by Lemma 20. Hence, we prove **APX**-hardness by an L-reduction from Max Sol$(\overline{R})$-11.

Given an instance $I = (V, D, C, w)$ of Max Sol$(\overline{R})$-11 where all variables have weight 1, let $F(I)$ be the instance of Min Sol$(\Gamma_{\{(+a,+b)\}})$ where all occurrences of $\overline{R}$ are replaced by R and all variables are given weight 1. Given a solution s to $F(I)$, let $G(I, s)$ be the solution to I defined as follows: $G(I, s)(x) = d - s(x)$ for all variables x. Consider an arbitrary constraint $C_i = (x_i \leq a_1 \vee x_j \leq b_1)$, then, by Lemma 22, an assignment s satisfies C_i if and only if the assignment $\overline{s} = G(I, s)$, satisfies $\overline{C_i} = (x_i \geq d - a_1 \vee x_j \geq d - b_1)$.

Since no variable occurs more than 11 times in I we have $\text{OPT}(I) \geq \frac{|V|}{12}$, $\text{OPT}(F(I)) \leq 12d \cdot \text{OPT}(I)$, and $\beta = 12d$ is a valid parameter in the L-reduction. Moreover, given a solution s to $F(I)$, we have $m(F(I), s) = |V|d - m(I, G(I, s))$. Furthermore, we have $\text{OPT}(F(I)) = |V|d - \text{OPT}(I)$. By these identities, it follows that:

$$\text{OPT}(I) - m(I, G(I, s)) = |V|d - \text{OPT}(I) - (|V|d - m(I, G(I, s))) =$$

$$= m(F(I), s) - \text{OPT}(F(I)),$$

and $|\text{OPT}(I) - m(I, G(I, s))| \leq |\text{OPT}(F(I)) - m(F(I), s)|$. Thus, $\gamma = 1$ is a valid parameter in the L-reduction. □

Lemma 24. *If* $(+b_1, \ldots, +b_q) \in L$ *and* $q \geq 2$*, then* Min Sol(Γ_L) *is* **APX***-hard.*

Proof. **APX**-hardness follows from Lemmata 7 and 23. □

5 Approximability of Max AW Sol

We continue by presenting a dichotomy result for the approximability of Max AW Sol.

Theorem 25. Max AW Sol(Γ_L) *is in* **PO** *if* Γ_L *is invariant under both* max *and* min. *Otherwise,* Max Sol(Γ_L) *does not admit a* Ptas *unless* **P**=**NP**.

The tractability part is proved in Section 5.1 and the remaining parts can be found in Section 5.2. Note that, in contrast to earlier sections, we do not prove **APX**-hardness results in this section. The reason is that we will now be forced to handle instances with negative optimal measure and **APX**-hardness and *AP*-reductions are only defined for problems with positive measure. However, it is still possible to rule out the existence of polynomial-time approximation schemes. For ordinary approximation problems, we say that a solution s is r-approximate if

$$\frac{\text{OPT}(I)}{r} \leq m(I,s) \leq \text{OPT}(I) \cdot r.$$

This does not work for problems with negative optima since in this case $\frac{\text{OPT}(I)}{r} \geq \text{OPT}(I) \cdot r$. Hence, it is important to check whether an optimum is positive or negative. With this in mind, we say that Π admits a Ptas if there exists an algorithm A satisfying the following property: for any instance I of Π and any rational value $r > 1$, $A(I, r)$ returns a solution s such that

$$m(I,s) \in [\text{OPT}(I)/r, r \cdot \text{OPT}(I)]$$

in time polynomial in $|I|$.

5.1 Tractability results

The polynomial-time algorithm is based on supermodular optimisation so we begin by giving some preliminaries. A partial order on a set X is called a *lattice* if every two elements, $a, b \in X$ have a greatest common lower bound $a \sqcap b$ (meet) and a least common upper bound $a \sqcup b$ (join). Then every lattice can be considered as an algebra $\mathcal{L} = (X, \sqcap, \sqcup)$ with operations meet and join.

A function $f : X \to \mathbb{R}$ is said to be *supermodular* on $\mathcal{L}$ if

$$f(\mathbf{a}) + f(\mathbf{b}) \leq f(\mathbf{a} \sqcap \mathbf{b}) + f(\mathbf{a} \sqcup \mathbf{b})$$

for all $\mathbf{a}, \mathbf{b} \in L$. A function f is called *submodular* if the reverse inequality holds, and *modular* if it is both super- and submodular (that is, the above inequality is an equality).

Given a finite set V, a *ring family* is a collection $\mathcal{V}$ of subsets of V such that $\mathcal{V}$ is closed under intersection and union. Clearly, every ring family is a lattice.

Theorem 26. *[34][2] Let $\mathcal{V}$ be a ring family over a finite set V and let $f : \mathcal{V} \to \mathbb{R}$ be a polynomial-time computable supermodular function on the lattice $(\mathcal{V}, \cap, \cup)$. Assume the following is known:*

1. *for each $v \in V$, the maximal set M_v in $\mathcal{V}$ that contains v (if any);*
2. *the maximal set M in $\mathcal{V}$.*

Then, the set $V^* \in \mathcal{V}$ that maximises f can be found in polynomial time.

Lemma 27. MAX AW SOL(Γ_L) *is in* **PO** *if Γ_L is invariant under both* max *and* min.

Proof. Let $I = (X, D, C, w)$ be an instance of MAX AW SOL(Γ_L), where Γ_L is invariant under both max and min, and $V = \{x_1, \ldots, x_n\}$. Consider the lattice $\mathcal{L} = (A, \sqcap, \sqcup)$ where $A \subseteq \mathbb{Z}^n$ are the solutions to I and for every $\mathbf{a} = (a_1, \ldots, a_n), \mathbf{b} = (b_1, \ldots, b_n) \in A$,

$$\mathbf{a} \sqcap \mathbf{b} = (\min(a_1, b_1), \ldots, \min(a_n, b_n))$$

and

$$\mathbf{a} \sqcup \mathbf{b} = (\max(a_1, b_1), \ldots, \max(a_n, b_n)).$$

We also see that the function $f : D^n \to \mathbb{Z}$ defined such that

$$f(x_1, \ldots, x_n) = \sum_{i=1}^{n} w_i \cdot x_i$$

is modular (and consequently supermodular) on $\mathcal{L}$: Arbitrarily choose $\mathbf{a} = (a_1, \ldots, a_n), \mathbf{b} = (b_1, \ldots, b_n) \in A$ and note that $\max(a_i, b_i) + \min(a_i, b_i) = a_i + b_i$, $1 \leq i \leq n$. Consequently,

$$f(\mathbf{a} \sqcap \mathbf{b}) + f(\mathbf{a} \sqcup \mathbf{b}) = \sum_{i=1}^{n} (w_i \cdot \min(a_i, b_i) + w_i \cdot \max(a_i, b_i)) =$$

$$= \sum_{i=1}^{n} w_i \cdot (a_i + b_i) = f(\mathbf{a}) + f(\mathbf{b}).$$

Finally, we construct a ring family $\mathcal{V}$ that represents $\mathcal{L}$. We choose $V = \{(i, d) \mid 1 \leq i \leq n,\ d \in D\}$ as base set and an element $\mathbf{a} = (a_1, \ldots, a_n) \in A$ is represented by

$$\bigcup_{i=1}^{n} \{(i, d) \mid d \in D \text{ and } d \leq a_i\}.$$

It is easy to see that $\sqcap$ corresponds to intersection and $\sqcup$ to union. For each $v = (i, d) \in V$, we can in polynomial time find the maximal element $M_v \in \mathcal{V}$

[2] The results are presented as the equivalent problem of minimising a submodular function. See also [21].

containing v as follows: define $C' = C \cup \{(x_i \geq d)\}$ and consider the MAX SOL instances

$$I_j = (X, D, C', w_j)$$

where $w_j(x_j) = 1$ and $w_j(x) = 0$ for all $x \neq x_j$. These instances are solvable in polynomial time since MAX SOL(Inv(max)) is a tractable problem. If there are no solutions to these problems, then $s(x_i) < d$ holds in every solution s that satisfies (X, D, C) and M_v is not defined. Otherwise, define $s^* : X \to D$ such that

$$s^*(x) = \max_{1 \leq j \leq n} s_j(x)$$

for all $x \in X$. The function s^* satisfies (X, D, C') since the constraints in C' are invariant under max and it is easy to realise that

$$M_v = \{(i, d) \mid 1 \leq i \leq n \text{ and } d \leq s^*(x_i)\}.$$

Similarly, the maximal element in $\mathcal{V}$ can be found in polynomial time; simply replace the constraint set C' with C. Theorem 26 is thus applicable and the result follows by noting that $|V| = n \cdot |D|$. □

Optimisation over max- and min-closed constraints has been considered in several other contexts, cf. [7, 18].

5.2 Proof of Theorem 25

We begin this section by proving that clausal languages containing a clause with at least 2 positive or 2 negative literals does not admit any PTAS unless **P**=**NP**. Finally, we note that the remaining clausal languages are tractable by the algorithm in the preceding section.

Lemma 28. *If* $(-c_1, \ldots, -c_p, +d_1, \ldots, +d_q) \in L$ *and* $p \geq 2$*, then* MAX AW SOL(Γ_L) *does not admit a* PTAS *unless* **P**=**NP**.

Proof. If $q = 0$, then the result immediately follows from Lemma 18 so we assume that $q \geq 1$. Let $c = c_1$, $b = \max(c_2, \ldots, c_p)$ and $e = \min(d_1, \ldots, d_q)$. By applying Lemma 7, we see that it is sufficient to prove the result for the constraint language $L = \{(-c, -b, +e)\}$. Assume now that MAX AW SOL(Γ_L) admits a PTAS A and arbitrarily choose $r > 1$. We will now construct a PTAS for MAX SOL($\Gamma_{\{(-c,-b)\}}$). We recall that MAX SOL($\Gamma_{\{(-c,-b)\}}$) is **APX**-hard (and do not admit a PTAS unless **P**=**NP**) by Lemmata 19 and 20.

Let $I = (V, D, C, w)$ be an arbitrary instance of MAX SOL($\Gamma_{\{(-c,-b)\}}$). We assume without loss of generality that $w(x) = 1$ for all $x \in V$. Assume that $V = \{x_1, \ldots, x_n\}$. We compute an instance $F(I) = (V', D, C', w')$ of MAX AW SOL(Γ_L) as follows:

- let $Y = \{y_1, \ldots, y_{|C|}\}$ be a set of fresh variables and let $V' = V \cup Y$;

– let $C' = \{(Rel((-c,-b,+e)),(x_i,x_j,y_k)) \mid \text{for each } c_k \in C \text{ where } c_k = (Rel((-c,-b)),(x_i,x_j))\}$; and
– let $w'(x) = -2\max D$ if $x \in Y$ and $w(x) = 1$ otherwise.

We first note that $\text{OPT}(F(I)) \geq \text{OPT}(I)$ since any solution s to I can be extended to a solution s' to $F(I)$ (of the same measure) by assigning 0 to all the variables in Y. Furthermore, $\text{OPT}(F(I)) \leq \text{OPT}(I)$ since there is an optimal solution s' to $F(I)$ such that $s'(y) = 0$ for all $y \in Y$, and hence this solution restricted to the V variables is a solution for I. This follows from the following observation: If s is an optimal solution for $F(I)$ and $s(y) > 0$ ($y \in Y$), then construct a new solution as follows: Let $(Rel((-c,-b,+e)),(x_i,x_j,y))$ be the unique constraint where y appears. Now, construct a modified solution s' such that $s'(z) = s(z)$ if $z \in V' - \{x_i,x_j,y\}$ and $s'(x_i) = s'(x_j) = s'(y) = 0$ otherwise. It is easy to see that s' is still a feasible solution, and, furthermore, that

$$m(F(I),s') = m(F(I),s) + 2s(y)\max D - s(x_i) - s(x_j) \geq$$

$$m(F(I),s) + 2s(y)\max D - 2\max D =$$

$$m(F(I),s) + 2\max D \cdot (s(y)-1) \geq m(F(I),s).$$

Hence, $\text{OPT}(F(I)) = \text{OPT}(I)$ and we have that

$$\text{OPT}(F(I))/r \leq m(F(I),A(F(I),r)) \leq \text{OPT}(F(I)) \cdot r.$$

Now, let $s' = A(F(I),r)$ and define s to be the solution to I where $s(x_i) = 0$ if x_i occurs in a constraint C_j and $y_j > 0$, and $s(x_i) = s'(x_i)$, otherwise. By the same argument as above, it is easy to verify that

$$\text{OPT}(I) - m(I,s) \leq \text{OPT}(F(I)) - m(F(I),s')$$

and

$$m(I,s) \geq m(F(I),s')$$

since $\text{OPT}(I) = \text{OPT}(F(I))$. Finally, we see that

$$\frac{\text{OPT}(I)}{r} = \frac{\text{OPT}(F(I))}{r} \leq m(F(I),s') \leq m(I,s) \leq \text{OPT}(I)$$

and Max Sol($\Gamma_{\{(-c,-b)\}}$) admits a Ptas.

□

Lemma 29. *If $P = (+b_1,\ldots,+b_q) \in L$ and $q \geq 2$, then* Max AW Sol(Γ_L) *does not admit a* Ptas *unless* **P**=**NP**.

Proof. By Lemma 7, we may without loss of generality assume that $q = 2$, $P = (+b,+c)$ and $D = \{0,1,\ldots,d\}$. Assume to the contrary that Max AW Sol($\Gamma_{\{(+b,+c)\}}$) admits a Ptas A. We will now construct a Ptas for the **APX**-complete problem Min Sol($\Gamma_{\{(+b,+c)\}}$). Arbitrarily choose $r > 1$.

Let $I = (V,D,C,w)$ be an arbitrary instance of Min Sol($\Gamma_{\{(+b,+c)\}}$) and construct the instance $I' = (V,D,C,w')$ where $w'(x) = -w(x)$ for all $x \in V$.

We see that if s is a solution to I with measure K, then s' (defined such that $s'(x) = -s(x)$ for all $x \in V$) is a solution to I' with measure $-K$, and vice versa. Now, let $s' = A(I', r)$ and define $s(x) = -s'(x)$ for all $x \in V$. We have that

$$\textsc{opt}(I') \cdot r \leq m(I', s') \leq \frac{\textsc{opt}(I')}{r}.$$

By multiplying this inequality with -1 we get

$$-\textsc{opt}(I') \cdot r \geq -m(I', s') \geq -\frac{\textsc{opt}(I')}{r}$$

and, consequently,

$$\textsc{opt}(I)/r \leq m(I, s) \leq \textsc{opt}(I) \cdot r.$$

Since r was arbitrarily chosen, Min Sol($\Gamma_{\{(+b,+c)\}}$) admits a Ptas. □

Lemma 30. *If* $(-a_1, \ldots, -a_p, +b_1, \ldots, +b_q) \in L$ *and* $q \geq 2$*, then* Max AW Sol(Γ_L) *admits no* Ptas *unless* **P**=**NP**.

Proof. If $p = 0$, then the result follows from Lemma 29 and if $p \geq 2$, then the result follows from Lemma 28 so we can assume that $p = 1$. By Lemma 7, it is sufficient to consider the clausal language $L = \{(-a, +b, +c)\}$.

Let $I = (V, D, C, w)$ be an arbitrary instance of Min Sol($\Gamma_{\{(+b,+c)\}}$)-11 where all variables have weight 1 and $D = \{0, \ldots, d\}$. By Lemma 23, this problem is **APX**-hard and admits no Ptas unless **P**=**NP**. Define $F(I) = (V', C', D, w')$ to be an instance of Max AW Sol($\Gamma_{\{(-a,+b,+c)\}}$) where $V' = V \cup Y$ and $Y = \{y_i \mid C_i \in C\}$, and all constraints $C_k = (Rel((+b, +c)), (x_i, x_j))$ have been replaced by $(Rel((-a, +b, +c)), (y_k, x_i, x_j))$. Moreover, let $w'(x) = -1$ for all $x \in V$ and $w'(y) = 2d$ for all $y \in Y$. We see that

$$m(F(I), s) = 2d \cdot \sum_{y \in Y} s(y) - \sum_{x \in V} s(x).$$

Next, we prove an upper bound on $\textsc{opt}(F(I))$: $\textsc{opt}(F(I)) \leq \alpha$ where $\alpha = 132d^2 \cdot \textsc{opt}(I)$. We begin by proving that $\textsc{opt}(F(I)) = 2d^2|Y| - \textsc{opt}(I)$. To see this, assume that s is an optimal solution to $F(I)$ where $s(y) < d$ for a variable $y \in Y$. Let

$$(R, \mathbf{x}) \equiv (y \leq a \vee x_i \geq b \vee x_j \geq c)$$

be the unique constraint where y appears. Now construct a modified solution s' such that $s'(z) = s(z)$ if $z \in V' \setminus \{y, x_i, x_j\}$, $s(x_i) = s(x_j) = d$, and $s(y) = d$. Obviously s' is a solution to $F(I)$ and

$$m(F(I), s') = m(F(I), s) + 2d(d - s(y)) - (2d - s(x_i) - s(x_j)) \geq m(F(I), s)$$

so there is an optimal solution to $F(I)$ such that d is assigned to all Y variables. Thus, any optimal solution s to I yields an optimal solution s' to $F(I)$ by letting $s'(y) = d$ for all $y \in Y$ and $s'(x) = s(x)$ for all variables in V. Hence, $\textsc{opt}(F(I)) = 2d^2|Y| - \textsc{opt}(I)$.

We can now prove that $\text{OPT}(F(I)) \leq \alpha$. Let $Nbh(v)$ denote the set of variables (including v) that occur in a constraint where v occurs. Since no variable $v \in V$ occurs more than 11 times in I, we have $|Nbh(v)| \leq 12$. Obviously, at least one of the variables in $Nbh(v)$ must be assigned a value > 0 since otherwise there will be a constraint $(v \geq b \vee v' \geq c)$ which is not satisfied (remember that $b, c > 0$). Hence, $\text{OPT}(I) \geq |V|/12$ and the number of constraints in I is at most $\frac{11|V|}{2}$ (i.e. $|Y| \leq \frac{11|V|}{2}$). The earlier result implies that $\text{OPT}(F(I)) = 2d^2|Y| - \text{OPT}(I)$ so $\text{OPT}(F(I)) \leq 2d^2 \cdot \frac{11|V|}{2} - \text{OPT}(I)$. We also know that $\text{OPT}(I) \geq |V|/12$ so $\text{OPT}(F(I)) \leq \alpha$.

We assume that MAX AW SOL($\Gamma_{\{(-a,+b,+c)\}}$) admits a PTAS A and arbitrarily choose $r > 1$. We will now construct a PTAS for MIN SOL($\Gamma_{\{(+b,+c)\}}$).

Let $s = A(F(I), r)$ and recall that $\text{OPT}(F(I))/r \leq m(F(I), s) \leq \text{OPT}(F(I)) \cdot r$. Let $G(I, s)$ be the solution to I defined as: $G(I, s)(x) = d$ if x occurs in a constraint together with a y variable such that $s(y) < d$, and $G(I, s)(x) = s(x)$ otherwise. We prove that $m(I, G(I, s)) - \text{OPT}(I) \leq \text{OPT}(F(I)) - m(F(I), s)$. Let $Y' = \{y \in Y \mid s(y) < d\}$ and note that $|Y'| \leq d|Y| - \sum_{y \in Y} s(y)$. By the definition of $G(I, s)$, we know that d is assigned to all variables x that occur in an equation together with a variable y from Y'. Denote the set of all such variables by X'. By the definition of $G(I, s)$ and the fact that $|X'| \leq 2|Y'|$, we have,

$$m(I, G(I, s)) = d|X'| + \sum_{x \in V \setminus X'} s(x) \leq 2d|Y'| + \sum_{x \in V \setminus X'} s(x).$$

It follows that

$$m(I, G(I, s)) - \text{OPT}(I) \leq$$

$$2d|Y'| + \sum_{x \in V \setminus X'} s(x) - \text{OPT}(I) \leq$$

$$2d|Y'| + \sum_{x \in V} s(x) - \text{OPT}(I) \leq$$

$$2d^2|Y| - 2d \sum_{y \in Y} s(y) + \sum_{x \in V} s(x) - \text{OPT}(I) =$$

$$\text{OPT}(F(I)) - m(F(I), s).$$

Now, $m(I, G(I, s)) - \text{OPT}(I) \leq \text{OPT}(F(I)) - m(F(I), s)$ and since $s = A(F(I), r)$, we have

$$m(I, G(I, s)) - \text{OPT}(I) \leq \text{OPT}(F(I)) - \frac{\text{OPT}(F(I))}{r}$$

implying that

$$m(I, G(I, s)) \leq \left(1 - \frac{1}{r}\right) \cdot \text{OPT}(F(I)) + \text{OPT}(I).$$

By $\textsc{opt}(F(I)) \leq \alpha$, it now follows that

$$m(I, G(I, s)) \leq \left(1 - \frac{1}{r}\right) \cdot \alpha + \textsc{opt}(I),$$

and as a consequence,

$$m(I, G(I, s)) \leq \left(\left(1 - \frac{1}{r}\right) \cdot 132d^2 + 1\right) \cdot \textsc{opt}(I).$$

Since r can be chosen arbitrarily close to 1, Min Sol($\Gamma_{\{(+b,+c)\}}$)-11 admits a Ptas. □

We are left with the case when every pattern P in L contains at most one positive and at most one negative literal. By Jeavons and Cooper [22, Theorem 5.2], if P is a pattern that contains at most one positive literal, then $Rel(P)$ is invariant under max. Similarly, $Rel(P)$ is invariant under min if P contains at most one negative literal. Thus, Max AW Sol(Γ_L) is polynomial-time solvable by Lemma 27. We also note that given a clausal language L, it is obvious how to check (in polynomial time) whether Max AW Sol(Γ_L) is polynomial-time solvable or not.

6 Conclusions and Open Questions

We have presented dichotomy results for the approximability of Max Sol, Min Sol, and Max AW Sol when they are restricted to constraint languages expressed by regular signed logic. The results were partly obtained by exploiting certain algebraic methods that have previously not been widely used for studying optimisation problems. In particular, the concept of a max-core seems to be very useful when classifying the approximability of Max Sol(Γ) for various classes of constraint languages Γ.

It has been noted before that adding non-negative integer weights to combinatorial optimisation problems does not seem to change the complexity of the problems. For example, the unweighted (all variables have weight 1) and weighted versions of Max Ones(Γ) and Min Ones(Γ) have the same approximability [26]. Therefore it is interesting to compare the dichotomies for Max Sol(Γ) and Max AW Sol(Γ) and too see how drastically the situation changes when negative weights are allowed.

We see at least three main ways of extending this work.

Extension 1: Provide a more fine-grained approximability analysis. In the case of boolean domains, such an analysis has been performed by Khanna et al. [26]; they prove that for any choice of allowed relations, the problem is either (1) polynomial-time solvable, (2) **APX**-complete, (3) **poly-APX**-complete, (4) finding a solution of measure > 0 is **NP**-hard; or (5) finding any solution is **NP**-hard. Hence, a venue for future research is to perform a similar analysis for optimisation problems over clausal constraints.

Extension 2: Consider left-hand restrictions. By *left-hand* restrictions, we mean restricting the way constraints are applied to variables (instead of restricting the allowed constraint types). Such problems have been extensively treated in many different contexts [8, 12, 16, 17]. All known results on the complexity of Max Sol(Γ) indicates that there does not exist any Γ such that Max Sol(Γ) has a polynomial-time approximation scheme but Max Sol(Γ) is not in **PO** ([24, 26] and Theorem 15). Natural such classes exist, however, if one restricts the way constraints are applied to variables. Maximum Independent Set (i.e. Max Sol($\Gamma_{(-0,-0)}$) over domain $D = \{0, 1\}$) is one example: the unrestricted problem is **poly-APX**-complete and not approximable within $O(n^{1-\varepsilon})$ for any $\varepsilon > 0$ (unless **P**=**NP**) [37], but the problem restricted to planar instances admits a Ptas [4, 30].

Extension 3: Study optimisation over infinite domains. Such problems have been intensively studied during many years since linear programming and integer programming are examples of such problems. However, the constraint languages are typically severely limited in various ways — for instance, being linear inequalities. Thus, we suggest to study constraint optimisation problem over other kinds of constraints. In particular, it would be interesting to study the complexity of Max Sol$_{\mathbb{N}}$(Γ) due to the additional challenges posed by the existence of unbounded solutions. We note that constraint satisfaction problems over infinite domains has attracted a great amount of interest. Such problems appear, for instance, in temporal reasoning [14] and in connection with tree description languages (which have applications in linguistics [27] and computational biology [35]). Many results on infinite-domain CSPs are collected in [5].

References

1. P. Alimonti and V. Kann. Some APX-completeness results for cubic graphs. *Theoretical Computer Science*, 237:123–134, 2000.
2. G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation*. Springer, 1999.
3. G. Ausiello, A. D'Atri, and M. Protasi. Structure preserving reductions among convex optimization problems. *Journal of Computer and System Sciences*, 21:136–153, 1980.
4. B. Baker. Approximation algorithms for NP-complete problems on planar graphs. *Journal of the ACM*, 41:153–180, 1994.
5. M. Bodirsky. *Constraint Satisfaction with Infinite Domains*. PhD thesis, Humboldt-Universität, Berlin, Germany, 2004.
6. A. Bulatov, P. Jeavons, and A. Krokhin. Classifying the computational complexity of constraints using finite algebras. *SIAM Journal on Computing*, 34(3):720–742, 2005.
7. D. Cohen, M. Cooper, and P. Jeavons. A complete characterization of complexity for Boolean constraint optimization problems. In *Proceedings of the 10th International Conference on Principles and Practice of Constraint Programming (CP-2004)*, pages 212–226, 2004.
8. D. Cohen, P. Jeavons, and M. Gyssens. A unified theory of structural tractability for constraint satisfaction and spread cut decomposition. In *Proceedings of the*

19th International Joint Conference on Artificial Intelligence (IJCAI-2005), pages 72–77, 2005.
9. D. Cohen, P. Jeavons, P. Jonsson, and M. Koubarakis. Building tractable disjunctive constraints. *Journal of the ACM*, 47(5):826–853, 2000.
10. N. Creignou, M. Hermann, A. Krokhin, and G. Salzer. Complexity of clausal constraints over chains. *Theory of Computing Systems*, 42(2):239–255, 2008.
11. N. Creignou, S. Khanna, and M. Sudan. *Complexity classifications of Boolean constraint satisfaction problems*. SIAM, Philadelphia, 2001.
12. V. Dalmau and P. Jonsson. The complexity of counting homomorphisms seen from the other side. *Theoretical Computer Science*, 329(1–3):315–323, 2004.
13. U. Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.
14. Michael Fisher, Dov Gabbay, and Lluís Vila, editors. *Handbook on Temporal Reasoning in Artificial Intelligence*. Elsevier, 2005.
15. À.J. Gil, M. Hermann, G. Salzer, and B. Zanuttini. Efficient algorithms for constraint description problems over finite totally ordered domains. In *Proceedings of Automated Reasoning, Second International Joint Conference (IJCAR-04)*, pages 244–258, 2004.
16. G. Gottlob, N. Leone, and F. Scarcello. The complexity of acyclic conjunctive queries. *Journal of the ACM*, 48(3):431–498, 2001.
17. Martin Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. In *Proceedings of the 44th IEEE Symposium on Foundations of Computer Science (FOCS-2003)*, pages 552–561, 2003.
18. D.S. Hochbaum and J. Naor. Simple and fast algorithms for linear and integer programs with two variables per inequality. *SIAM J. Comput.*, 23(6):1179–1192, 1994.
19. J.N. Hooker and M. Osorio. Mixed logical-linear programming. *Discrete Applied Mathematics*, 96-97:395–442, 1999.
20. R. Hähnle. Complexity of many-valued logics. In *Proceedings of the 31st IEEE International Symposium on Multiple-valued Logic (ISMVL-2001)*, pages 137–148, 2001.
21. S. Iwata, L. Fleischer, and S. Fujishige. A combinatorial strongly polynomial algorithm for minimizing submodular functions. *Journal of the ACM*, 48(4):761–777, 2001.
22. P. G. Jeavons and M. C. Cooper. Tractable constraints on ordered domains. *Artificial Intelligence*, 79:327–339, 1996.
23. Peter Jeavons, David Cohen, and Marc Gyssens. Closure properties of constraints. *Journal of the ACM*, 44(4):527–548, 1997.
24. P. Jonsson, F. Kuivinen, and G. Nordh. Max-Ones generalised to larger domains. *SIAM J. Comput.*, 38(1):329–365, 2008.
25. P. Jonsson, G. Nordh, and J. Thapper. The maximum solution problem on graphs. In *Proceedings of the 32nd International Symposium on Mathematical Foundations of Computer Science (MFCS-2007)*, pages 228–239, 2007.
26. S. Khanna, M. Sudan, L. Trevisan, and D.P. Williamson. The approximability of constraint satisfaction problems. *SIAM J. Comput.*, 30(6):1863–1920, 2001.
27. A. Koller, K. Mehlhorn, and J. Niehren. A polynomial-time fragment of dominance constraints. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL-2000)*, pages 368–375, 2000.
28. 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)*, pages 628–639, 2005.

29. R. E. Ladner. On the structure of polynomial time reducibility. *Journal of the ACM*, 22(1):155–171, 1975.
30. R. Lipton and R. Tarjan. Applications of a planar separator theorem. *SIAM Journal on Computing*, 9:615–627, 1980.
31. Christos H. Papadimitriou. *Computational Complexity*. Addison Wesley, Reading, MA, 1994.
32. R. Pöschel. Galois connections for operations and relations. Technical Report MATH-AL-8-2001, Technische Universität Dresden, Germany, 2001.
33. R. Pöschel and L.A. Kalužnin. *Funktionen- und Relationenalgebren*. DVW, Berlin, 1979.
34. A. Schrijver. A combinatorial algorithm minimizing submodular functions in polynomial time. *Journal of Combinatorial Theory, ser. B*, 80:346–355, 2000.
35. M. Steel. The complexity of reconstructing trees from qualitative characters and subtrees. *Journal of classification*, 9:91–116, 1992.
36. S.A. Wolfman and D.S. Weld. The LPSAT engine & its application to resource planning. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 310–317, 1999.
37. 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)*, pages 681–690, 2006.