Generalised Integer Programming Based on Logically Defined Relations

Peter Jonsson* and Gustav Nordh**

Department of Computer and Information Science, Linköpings Universitet
S-581 83 Linköping, Sweden, {petej, gusno}@ida.liu.se

Abstract. Many combinatorial optimisation problems can be modelled as integer linear programs. We consider a class of generalised integer programs where the constraints are allowed to be taken from a broader set of relations (instead of just being linear inequalities). The set of allowed relations is defined using a many-valued logic and the resulting class of relations have provably strong modelling properties. We give sufficient conditions for when such problems are polynomial-time solvable and we prove that they are **APX**-hard otherwise.

1 Introduction

Combinatorial optimisation problems can often be formulated as *integer programs*. In its most general form, the aim in such a program is to assign integers to a set of variables such that a set of linear inequalities are satisfied and a linear goal function is maximised. That is, one wants to solve the optimisation problem

$$\begin{aligned} &\max c^T x \\ &Ax \geq b \\ &x \in \mathbb{N}^n \end{aligned}$$

where A is an $m \times n$ rational matrix, b is a rational m-vector, and c is a rational n-vector. It is widely acknowledged that many real-world problems can be conveniently captured by integer programs. In its general form, integer programming is **NP**-complete to solve exactly [4].

Many restricted variants of the integer programming problem are still fairly expressive and have gained much attention in the literature. One such restriction is to restrict x such that $x \in \{0,1\}^n$. This problem, commonly called MAXIMUM 0-1 PROGRAMMING, is still very hard, in fact it is **NPO**-complete [9]. Another common restriction is to only allow c to contain non-negative values. Under these restrictions, the complexity and approximability of MAXIMUM 0-1 PROGRAMMING is very well-studied, cf. [3, 10]. In fact, much is known even if the constraints imposed by the inequality $Ax \geq b$ are replaced by other types of constraints. For instance, MAX ONES is MAXIMUM 0-1 PROGRAMMING with non-negative weights and arbitrary constraints over $\{0,1\}$.

* 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.
** Supported by the *National Graduate School in Computer Science* (CUGS), Sweden.

The approximability of MAX ONES has been completely classified for every set Γ of allowed constraints over $\{0,1\}$ [10].

In this paper, we consider a class of generalised integer programming problems where the variable domains are finite (but not restricted to be 2-valued); and the constraints are allowed to be taken from a set of logically defined relations. The set of relations that we consider is based on *regular signed logic* [6], where the underlying finite domain is a totally-ordered set of integers $\{0,\ldots,d\}$. This logic provides us with convenient concepts for defining a class of relations with strong modelling capabilities. Jeavons and Cooper [8] 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 given by Gil et al. [5]. 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, [2, 8]. However, optimisation within this framework has not been considered earlier. To make the presentation simpler, let MAX SOL denote the maximisation problem restricted to positively weighted objective functions, MIN SOL denote the minimisation version of MAX SOL, and let MAX AW SOL denote the problem without restrictions on the weights (here, AW stands for arbitrary weights). The reader is referred to Section 2 for exact definitions.

Let $\mathcal{R}$ denote the class of relations that can be defined by the regular signed logic. Our aim is to, given a subset Γ of $\mathcal{R}$, classify the complexity of the optimisation 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 problems MAX SOL(Γ), MIN SOL(Γ), and MAX AW SOL(Γ).

Our main results are: For these three problems, we give sufficient conditions for when they are polynomial-time solvable and we prove that they are **APX**-hard otherwise. We also show that the tractable fragments can be efficiently recognised. 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. This kind of dichotomy results are important in computational complexity since they can be seen as exceptions to Ladner's [11] 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.

There has been much research on combining integer (linear) programming and logic/constraint programming in order to benefit from strengths of the respective areas (cf. [7, 15]). One approach is to increase the modeling power of integer (linear) programming by allowing models to contain logic formulas. Our work can be seen as a crude estimation for the price, in terms of computational complexity, that comes with the additional expressive power of such an approach. Our results can also be seen as a first step towards extending the approximability classification for MAX ONES in [10]

to non-Boolean domains. One of the observations of [10] was that none of the (infinite number) of combinatorial optimisation problems captured by MAX ONES have a polynomial time approximation scheme (unless **P**=**NP**). Our results strongly indicates that the situation remains the same for MAX ONES generalised to arbitrary finite domains. Our work also complement the recent dichotomy result due to Creignou et al. [2] for the decision version (decide whether there is a solution at all) of exactly the same framework that we study in this paper.

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 the concept of *max-cores* (which is a variant of the algebraic and graph-theoretic concept *core*).

The paper is structured as follows: Section 2 contains some basic definitions of constraint satisfaction and logical methods for constructing constraint languages. Section 3 presents the methods used for proving the results and the main results for the three problems are presented in Sections 4 and 5. Finally, some concluding remarks are given in Section 6. Due to space limitations, many proofs have been moved to an appendix.

2 Integer programming and logic

We begin by presenting the *constraint satisfaction problem* and how it is connected with integer programming. We continue by showing how logics can be used for defining relations suited for integer programming.

We define constraint satisfaction as follows: Let the *domain* $D = \{0, 1, \ldots, d\}$ be equipped with the total order $0 < 1 < \ldots < d$. 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 $\text{CSP}(\Gamma)$, 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* $\text{MAX SOL}(\Gamma)$*, 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: $\sum_{v \in V} w(v) \cdot f(v)$

We will also consider the analogous minimisation problem MIN SOL and the problem MAX AW SOL where the weights are arbitrary, i.e., not assumed to be non-negative. By choosing Γ appropriately, many integer programming problems can be modelled by using these problems. For instance, the well-known 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($\{(0,0),(0,1),(1,0)\}$ problem. Sometimes we will consider problems restricted to instances where each variable may occur at most k times, denoted MAX SOL(Γ)-k. Next, we consider a framework based on regular signed logic for expressing constraint languages using logic which was introduced by Creignou et al. in [2].

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 also 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 $\{0, 1\}$, then MAX SOL($(-0, -0)$) is exactly MAXIMUM INDEPENDENT SET, and MIN SOL($(+1, +1)$) is exactly MINIMUM VERTEX COVER.

3 Methods

3.1 Approximability and reductions

A *combinatorial optimisation problem* is defined over a set $\mathcal{I}$ of *instances* (admissible input data); each instance $I \in \mathcal{I}$ has a finite set $\mathsf{sol}(I)$ of *feasible solutions* associated with it. The objective is, given an instance I, to find a feasible solution of *optimum* value with respect to some measure function $m : \mathcal{I} \times \mathsf{sol}(I) \to \mathbb{N}$. The optimal value is the largest one for *maximisation* problems and the smallest one for *minimisation* problems. A combinatorial optimisation problem is in **NPO** if its instances and solutions can be recognised in polynomial time, the solutions are polynomially-bounded in the input size, and the measure function can be computed in polynomial time (see, e.g., [1]).

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 $\max\left\{\frac{m(I,s)}{\mathrm{OPT}(I)}, \frac{\mathrm{OPT}(I)}{m(I,s)}\right\} \leq r$, where $\mathrm{OPT}(I)$ 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.

Let **PO** denote 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. Completeness in **APX** is defined using a reduction called AP-reduction [3, 10]. 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 well-known (and not difficult to prove) that every **APX**-hard problem is **NP**-hard. In our proofs it will be more convenient for us to use another type of approximation-preserving reduction called L-reduction [1].

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

A well-known fact (see, e.g., Lemma 8.2 in [1]) is 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. Hence, when proving **APX**-hardness results we can use L-reductions instead of AP-reductions as long as the problem we are reducing from is in **APX**. It is well-known (cf. [1, Corr. 3.13]) that if **P** $\neq$ **NP**, then no **APX**-complete problem can have a **PTAS**.

3.2 Algebraic framework

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 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 Pol(Γ) are known as *clones*, and they are well-studied objects in algebra (cf. [13]).

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 $\mathbf{x}_1, \ldots, \mathbf{x}_k$ are vectors of variables such that $ar(R_i) = |\mathbf{x}_i|$ for all i. 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 Γ by pp-formulas. The following lemma states that MAX SOL over finite subsets of $\langle\Gamma\rangle$ is no harder than MAX SOL over Γ itself. The lemma provides a strong approximation preserving reduction (sometimes called S-reduction) which is simultaneously an AP- and L-reduction and preserves membership in approximations classes such as **PO** and **APX**.

Lemma 4. *Let Γ and Γ' be finite constraint languages 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.*

Proof. We give an approximation preserving reduction from MAX SOL(Γ') to MAX SOL(Γ). Consider an instance $I = (V, D, C, w)$ of MAX SOL(Γ'). We transform I into an instance $F(I) = (V', D, C', w')$ of MAX SOL(Γ). For every constraint $C = (R, (v_1, \ldots, v_m))$ 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 $(R_1, (v_{11}, \ldots, v_{1n_1})), \ldots, (R_k, (v_{k1}, \ldots, v_{kn_k}))$, 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 it results in an equivalent instance of MAX SOL($\Gamma \cup \{=_D\}$).

Each equality constraint can be removed by identifying variables that are forced to be equal, replacing them with a single variable, and updating the weight function w accordingly. The resulting instance $F(I) = (V', D, C', w')$ of MAX SOL(Γ) has the same optimum as I (i.e., OPT(I) = OPT($F(I)$)) and can be 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 approximation preserving reduction from MAX SOL(Γ') to MAX SOL(Γ). □

The lemma above obviously 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 easy.

Lemma 5. *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(Γ_{P_1}) *implies the* **APX***-hardness of* MAX SOL(Γ_P)*. Similarly, if $P_2 = (+a_1, \ldots, +a_p, -b_1, -\max\{b_2, \ldots, b_q\})$, then* **APX***-hardness of* MAX SOL(Γ_{P_2}) *implies* **APX***-hardness of* MAX SOL(Γ_P)*. The same results also holds for* MIN SOL *and* MAX AW SOL.

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 $f(R) = \{(f(d_{11}), \ldots, f(d_{1m})), \ldots, (f(d_{t1}), \ldots, f(d_{tm}))\}$. Similarly, let $f(\Gamma)$ denote the constraint language $\{f(R) \mid R \in \Gamma\}$.

Lemma 6. *Let Γ be a finite constraint language over D and f a unary operation in $Pol(\Gamma)$ such that $f(d) \geq d$ for all $d \in D$. If* MAX SOL($f(\Gamma)$) *is* **APX***-complete, then* MAX SOL(Γ) *is* **APX***-hard, and if* MAX SOL($f(\Gamma)$) *is in* **PO***, then so is* MAX SOL(Γ).

Proof. We prove that MAX SOL($f(\Gamma)$) is L-reducible to MAX SOL(Γ). Given an instance $I = (V, D, C, w)$ of MAX SOL($f(\Gamma)$) we 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. Since $f \in Pol(\Gamma)$ and $f(d) \geq d$ for all $d \in D$, we have that $\text{OPT}(I) = \text{OPT}(F(I))$ and $\text{OPT}(I) - m(G(I, s') \leq \text{OPT}(F(I)) - m(s')$. As for the other direction, we can give a reduction similar to the one above from MAX SOL(Γ) to MAX SOL($f(\Gamma)$) mapping optimal solutions back to optimal solutions. Hence, if MAX SOL($f(\Gamma)$) is in **PO**, then so is MAX SOL(Γ). □

The concept of a core of a constraint language Γ has previously shown its value when classifying the complexity of CSP(Γ). We define a related concept for MAX SOL(Γ) and call it *max-core*.

Definition 7. *A constraint language Γ is a max-core if and only if there is no non-injective unary operation f in $Pol(\Gamma)$ such that $f(d) \geq d$ for all $d \in D$. 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 unary operation $f \in Pol(\Gamma)$ such that $f(d) \geq d$ for all $d \in D$.*

The next lemma follows directly from Lemma 6.

Lemma 8. *If Γ' is a max-core of Γ and if* MAX SOL(Γ') *is* **APX***-complete, then* MAX SOL(Γ) *is* **APX***-hard and if* MAX SOL(Γ') *is in* **PO** *then so is* MAX SOL(Γ).

4 Approximability of MAX SOL

In this section present sufficient conditions for when MAX SOL is tractable and prove that it is **APX**-hard otherwise. To do so, we need 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 9. 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 4.1-4.3.

4.1 Tractability result

Before we can prove the tractability of MAX SOL($Inv(\max_u)$), we need to introduce some terminology: Let $\mathbf{x} = (x_1, \ldots, x_r)$ be a list of r variables and let $(R, \mathbf{x})$ be a constraint on $\mathbf{x}$. For any sublist $\mathbf{x}' = (x_{i_1}, \ldots x_{i_k})$ of $\mathbf{x}$, define the *projection* of $(R, \mathbf{x})$ onto $\mathbf{x}'$, denoted $\pi_{\mathbf{x}'}(R, \mathbf{x})$, as follows: $\pi_{\mathbf{x}'}(R, \mathbf{x}) = \{\text{assignments to } (x_{i_1}, \ldots, x_{i_k}) \mid (R, (x_1, \ldots, x_r)) \text{ has a solution}\}$. A CSP instance is said to be *pair-wise consistent* if for any pair of constraints $(R, \mathbf{x}), (R', \mathbf{y})$, $\pi_{\mathbf{x}\cap\mathbf{y}}((R, \mathbf{x})) = \pi_{\mathbf{x}\cap\mathbf{y}}((R', \mathbf{y}))$.

Lemma 10. *If* Γ_L *is invariant under* $\max_u$ *for some* $u \in D$*, then* MAX SOL(Γ_L) *is in* **PO**.

Proof. We begin by observing that if Γ_L is invariant under $\max_u$, then Γ_L is also invariant under the unary operation $u(x) = \max_u(x, x)$ (satisfying the condition $u(x) \geq x$). Hence, by Lemma 6, MAX SOL(Γ_L) is in **PO** if MAX SOL(Γ'_L) is in **PO** where $u(\Gamma_L) = \Gamma'_L$. Now, Γ'_L contains no tuple with an element $a < u$ and hence Γ'_L is invariant under $\max$ (since it is invariant under $\max_u$ and $\max_u$ acts as $\max$ on elements $\geq u$). Hence, it is sufficient to give a polynomial-time algorithm solving MAX SOL(Γ'_L) for $\max$-closed constraint languages Γ'_L. This algorithm is a straightforward modification of the polynomial-time algorithm for CSP($Inv(\max_D)$) presented in [8].

Let $I = (V, D, C, w)$ be an instance of MAX SOL(Γ'_L). Assume I to be pair-wise consistent. If I contains the empty constraint, then I has no solution. Otherwise, define $f : V \to D$ such that $f(x_i) = \max\{\pi_{(x_i)}((R, \mathbf{x})) \mid (R, \mathbf{x}) \in C\}$, i.e., f assigns to each variable the maximum value it is allowed by any constraint. We claim that this f is a solution to I. Consider any constraint $(R, \mathbf{x})$ where, say for simplicity, $\mathbf{x} = (x_1, \ldots, x_r)$. For each variable x_j, we must have some tuple $t_i \in R$ such that $t_i[j] = f(x_j)$ by pair-wise consistency and the choice of f. Since R is closed under $\max$, the maximum of all these tuples belong to R. This maximum tuple equals $(f(x_1), \ldots, f(x_r))$ and f satisfies the constraint.

Assume now that there exists a function $f' : V \to D$ such that f' satisfies all constraints in C and $\sum_{i=1}^{n} w(x_i) \cdot f'(x_i) > \sum_{i=1}^{n} w(x_i) \cdot f(x_i)$. Since $w(x_i) \geq 0$ for every $x_i \in V$, this implies that there exists at least one variable x_i such that $f'(x_i) > f(x_i)$. However, this is impossible by the choice of f.

To conclude the proof, f can be constructed in $O(|C|^2 a^2)$ time (where a is the maximum arity of the constraints in C) by Corollary 4.3 in [8]. □

4.2 APX-hardness results

We will show that whenever P is a negative pattern containing at least two literals, then MAX SOL($Rel(P)$) is **APX**-hard. We begin by presenting 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:

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.

Lemma 11. *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.*

Proof. Membership in **APX** follows from the fact that the all-1 assignment is a 2-approximation. We prove **APX**-hardness by giving a 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 $\max\{\sum_{v \in V} M(v) \mid M \text{ is a satisfying assignment}\} = 7$ and the optimum appears if and only 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: $s'(x) = true$ if there exists a literal $l_j^i = x$ and $s(X_j^i) = 2$, $s'(x) = false$ if there exists a literal $l_j^i = \neg x$ and $s(X_j^i) = 2$, and $s'(x) = false$ 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

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

or, equivalently, $m(F(I), s') - m(I, s) > 6m$. It is easy to reach a contradiction from this 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 $\text{MAX SOL}(r)$. □

By combining the notion of max-cores and Lemma 11, we can prove **APX**-hardness for all negative patterns of length at least two:

Lemma 12. *If* $(-c_1, \ldots, -c_k) \in L$, $k \geq 2$, *then* $\text{MAX SOL}(\Gamma_L)$ *is* **APX***-hard.*

4.3 Proof of Theorem 9

Proof. Arbitrarily choose a clausal language L. If a pattern $(-c_1, -c_2, \ldots, -c_k)$, $k \geq 2$, exists in L, then $\text{MAX SOL}(\Gamma_L)$ is **APX**-hard by Lemma 12. 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 $\text{MAX SOL}(\Gamma_L)$ 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

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

Let ψ be a pp-formula defining the set U, i.e., $U(x) \equiv \exists \mathbf{x} : \psi(x; \mathbf{x})$ 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 $\text{MAX SOL}(\Gamma_L)$ is **APX**-hard by Lemmata 4 and 12. If this is not the case, then we show that Γ_L is invariant under $\max_u$ and, by Lemma 10, that $\text{MAX SOL}(\Gamma_L)$ 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 $(\max_u(t_1, t'_1), \ldots, \max_u(t_{p+1}, t'_{p+1}))$ 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 $a, b \in (-r)$. 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 closed 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 9. 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.

5 Approximability of MIN SOL and MAX AW SOL

We now turn our attention to the two problems MIN SOL (i.e., the minimization version of MAX SOL) and MAX AW SOL (i.e., MAX SOL without the restriction of non-negative weights). We see, for instance, that MIN SOL($(+1, +1)$) (over the domain $D = \{0, 1\}$) is the same problem as the minimum vertex cover problem.

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**. The operations $\min_u$ and $\max_u$ are symmetrically defined. By combining this with certain hardness results, one can prove the following:

Theorem 13. *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.*

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 continue by presenting sufficient and necessary conditions for tractability of MAX AW SOL.

Theorem 14. *MAX AW SOL(Γ_L) is in* **PO** *if Γ_L is invariant under both* max *and* min*. Otherwise,* MAX SOL(Γ_L) *is* **APX***-hard.*

The tractability part is based on supermodular optimisation [14] while the inapproximability results are proved by appropriate reductions from MAX SOL and MIN SOL.

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

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.

One way to extend this work is to provide a more fine-grained approximability analysis of these problems. In the case of boolean domains, such an analysis has been performed by Khanna et al. [10]; 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. Another venue for future research would be investigate the approximability of MAX (AW) SOL(Γ) for arbitrary finite domain constraint languages Γ, i.e., constraint languages not necessarily expressed by regular signed logic.

References

1. G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation*. Springer, 1999.
2. N. Creignou, M. Hermann, A. Krokhin, and G. Salzer. Complexity of clausal constraints over chains. 2006. To appear in: Theory of Computing Systems. Preliminary version available from: `www.cis.syr.edu/~royer/lcc/LCC05`.
3. N. Creignou, S. Khanna, and M. Sudan. *Complexity Classifications of Boolean Constraint Satisfaction Problems*, volume 7 of *SIAM Monographs on Discrete Mathematics and Applications*. SIAM, 2001.
4. M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, 1979.
5. 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.
6. R. Hähnle. Complexity of many-valued logics. In *Proceedings of the 31st IEEE International Symposium on Multiple-valued Logic (ISMVL-01)*, pages 137–148, 2001.
7. J.N. Hooker and M. Osorio. Mixed logical-linear programming. *Discrete Applied Mathematics*, 96-97:395–442, 1999.
8. P. G. Jeavons and M. C. Cooper. Tractable constraints on ordered domains. *Artificial Intelligence*, 79:327–339, 1996.
9. P. Jonsson. Boolean constraint satisfaction: complexity results for optimization problems with arbitrary weights. *Theoretical Computer Science*, 244(1-2):189–203, 2000.
10. S. Khanna, M. Sudan, L. Trevisan, and D.P. Williamson. The approximability of constraint satisfaction problems. *SIAM J. Comput.*, 30(6):1863–1920, 2000.
11. R. E. Ladner. On the structure of polynomial time reducibility. *Journal of the ACM*, 22(1):155–171, 1975.
12. C.H. Papadimitriou. *Computational Complexity*. Addison Wesley, Reading, MA, 1994.
13. R. Pöschel and L. Kalužnin. *Funktionen- und Relationenalgebren*. DVW, Berlin, 1979.
14. A. Schrijver. A combinatorial algorithm minimizing submodular functions in polynomial time. *Journal of Combinatorial Theory, ser. B*, 80:346–355, 2000.
15. 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.

A Proof of Lemma 12

The proof of Lemma 12 is based on Lemmas 15 and 16.

Lemma 15. *If $(-a,-b) \in L$ where $a < b$, then* MAX SOL(Γ_L)-11 *is* **APX***-complete.*

Proof. Let $d = \max D$ and 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.

As for **APX**-hardness, first 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 $f \in Pol(R)$, $f(x) \geq x$ for all $x \in D$, and that $R' = f(R)$ 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 8 that MAX SOL(R) is **APX**-hard if MAX SOL(R') is **APX**-complete. Hence, to prove that MAX SOL(R) is **APX**-hard, it is sufficient to prove that 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 11, 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 instead assigned 1, and all variables assigned d are instead assigned 2. We have, OPT$(F(I)) \leq d \cdot$ OPT(I) and

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

hence, $\beta = d$ 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 [16].

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 also, without loss of generality, assume that the graphs under consideration do not contain any isolated vertices.

Lemma 16. MAX SOL$(\Gamma_{(-a,-a)})$-3 *is* **APX***-complete even if all weights equal* 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, 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 [12, 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 $f \in Pol(R)$, $f(x) \geq x$ for all $x \in D$, and that $R' = f(R) = \{(a, a), (a, d), (d, a)\}$ is a max-core. We know from Lemma 8 that MAX SOL(R) is **APX**-hard if MAX SOL(R') is **APX**-complete. Hence, to prove that MAX SOL(R) is **APX**-hard, it is sufficient to prove that MAX SOL(R') is **APX**-hard.

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 $G(I, s')$ be the solution for I where all vertices corresponding to variables assigned b in S' form the independent set. We have $|V|/4 \leq$ OPT(I) and OPT$(F(I)) \leq b|V|$ so OPT$(F(I)) \leq 4b$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. This L-reduction ensures that MAX SOL(R') is **APX**-hard.

By combining Lemma 5 with either Lemma 15 or Lemma 16, we see that all negative patterns of length at least 2 are hard and Lemma 12 is proved.

B Approximability of MIN SOL

We prove that MIN SOL(Γ) is **APX**-hard unless Γ is invariant under $\min_u$ for some $u \in D$.

Lemma 17. *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* $x \in \mathbf{x}$*.*

Proof. To realise this, assume without loss of generality that $s(x_1) \geq a_1$ (i.e., s satisfies $(P, \mathbf{x})$ by fulfilling $x_1 \geq a_1$), which is equivalent to $-s(x_i) \leq -a_1$ (or $d - s(x_i) \leq d - a_1$) and hence $\overline{s}$ satisfies $(\overline{P}, \mathbf{x})$ by satisfying $x_1 \leq d - a_1$. The other direction works analogously. □

In the next proof, we will exploit 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., $\sum_{v \in V'} w(v)$.

Lemma 18. MIN SOL($\Gamma_{(+a,+b)}$)-11 *is* **APX**-*complete.*

Proof. We begin by showing membership in **APX**. Let $I = (V, D, C, w)$ be an arbitrary instance of MIN SOL($\Gamma_{(+a,+b)}$) and assume that $b \geq a$. Now, construct a vertex-weighted graph (V, E, w) where $(v, v') \in E$ if and only if $(Rel((-0, -0), (v, v'))) \in C$. Clearly, $|M| \cdot b \geq \text{OPT}(I)$ where $M \subseteq V$ is a vertex cover in G of minimum weight. The minimum weight vertex cover problem is approximable within 2 [17] which implies that MIN SOL($\Gamma_{(+a,+b)}$) is approximable within $2b$.

We continue with the hardness result. 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($\overline{R}$) is **APX**-complete by Lemma 15 and if $a = b$, then MAX SOL($\overline{R}$)-3 is **APX**-complete by Lemma 16. Hence, we prove **APX**-hardness by an L-reduction from either MAX SOL($\overline{R}$) or MAX SOL($\overline{R}$)-3 depending on a and b.

Given an instance $I = (V, D, C, w)$ of either MAX SOL($\overline{R}$) or MAX SOL($\overline{R}$)-3 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 17, 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 19. *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 5 and 18. □

The proof of the main theorem for MIN SOL(Γ_L) is analogous to the corresponding proof for MAX SOL(Γ_L) and is therefore omitted.

Theorem 20. 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.*

C Approximability of MAX AW SOL

In this section, we give sufficient conditions for the tractability of MAX AW SOL and prove that it is **APX**-hard otherwise.

Theorem 21. MAX AW SOL(Γ_L) *is in* **PO** *if Γ_L is invariant under both* max *and* min*. Otherwise,* MAX SOL(Γ_L) *is* **APX***-hard.*

The tractability part is proved in Section C.1 and the remaining parts can be found in Section C.2

C.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 22. *[14][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:*

[2] The results are presented as the equivalent problem of minimising a submodular function. See also [19].

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 23. 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 $\mathcal{L} = (L, \sqcap, \sqcup)$ where $L \subseteq \mathbb{Z}^n$ are the solutions to I and for every $\mathbf{a} = (a_1, \ldots, a_n), \mathbf{b} = (b_1, \ldots, b_n) \in L$,

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

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 $\mathbf{a} = (a_1, \ldots, a_n), \mathbf{b} = (b_1, \ldots, b_n) \in L$ 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 $\mathbf{a} = (a_1, \ldots, a_n) \in L$ 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 (remember that MAX SOL($Inv(\max)$ is a tractable problem) find the maximal element $M_v \in \mathcal{V}$ containing v as follows: Solve the MAX SOL instance $I' = (X, D, C, w')$ where $w'(x_i) = 1$ and $w(x) = 0$ whenever $x \neq x_i$. If $w'(\text{OPT}(I')) \leq d$, then $M_v = \text{OPT}(I')$ and, otherwise, M_v is undefined. Similarly, the maximal element in $\mathcal{V}$ can be found in polynomial time. Consequently, Theorem 22 is applicable and the result follows by noting that $|V| = n \cdot |D|$. □

Supermodular maximisation over Boolean max- and min-closed constraints has been considered in other contexts, cf. [18].

C.2 Proof of Theorem 14

In this section we begin by proving **APX**-hardness results for all clausal languages containing a clause with at least 2 positive or 2 negative literals. Finally, we note that the remaining clausal languages are tractable by the algorithm in the preceding section.

Lemma 24. *If* $(-c_1, \ldots, -c_p, +d_1, \ldots, +d_q) \in L$ *and* $p \geq 2$*, then* MAX AW SOL(Γ_L) *is* **APX***-hard.*

Proof. If $q = 0$, then the result immediately follows from Lemma 12 so we assume that $q \geq 1$. Let $c = c_1$, $d = \max(c_2, \ldots, c_k)$ and $e = \min(d_1, \ldots, d_q)$. By applying Lemma 5, we see that it is sufficient to prove the result for the constraint language $L = \{(-c, -d, +e)\}$. If $c \neq d$, then MAX SOL$(\Gamma_{(-c,-d)})$ is **APX**-complete by Lemma 15 and if $c = d$, then MAX SOL$(\Gamma_{(-c,-d)})$-3 is **APX**-complete by Lemma 16. Hence, we give an L-reduction from either MAX SOL$(\Gamma_{(-c,-d)})$ or MAX SOL$(\Gamma_{(-c,-d)})$-3 depending on c and d.

If $c \neq d$, then let $I = (V, D, C, w)$ be an arbitrary instance of MAX SOL$(\Gamma_{(-c,-d)})$ and, otherwise, let $I = (V, D, C, w)$ be an arbitrary instance of MAX SOL$(\Gamma_{(-c,-d)})$-3. 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 $C' = \{(Rel((-c, -d, +e)), (x_i, x_j, y_k)) \mid \text{for each } c_k \in C \text{ where } c_k = (Rel((-c, -d)), (x_i, x_j))\}$; and
- let $w'(x) = -2 \max D$ if $x \in Y$ and $w(x) = 1$ otherwise.

We first note that OPT$(F(I)) \geq$ 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 y variables. Furthermore, OPT$(F(I)) \leq$ 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 $(Rel((-c, -d, +e)), (x_i, x_j, y))$ 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

$$m(F(I), s') = m(F(I), s) - s(x_i) - s(x_j) + 2 \max D \cdot s(y) \geq$$
$$\geq m(F(I), s) - 2 \max D + 2 \max D \cdot s(y) =$$
$$= m(F(I), s) + 2 \max D \cdot (s(y) - 1) \geq m(F(I), s)$$

Hence, OPT$(F(I)) =$ OPT(I) and $\beta = 1$ is a valid parameter in the L-reduction.

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

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

Hence, $\gamma = 1$ is a valid parameter in the L-reduction. □

Lemma 25. *If $P = (+b_1, \ldots, +b_q) \in L$ and $q \geq 2$, then* MAX AW SOL(Γ_L) *is* **APX**-*hard.*

Proof. By Lemma 5, we may without loss of generality assume that $q = 2$ and $P = (+b, +c)$. Now, let $D = \{0, 1, \ldots, d\}$, $b = b_1$, $R = Rel((+b, +c))$, and $\overline{R} = Rel(\ (-(d-b), -(d-c)))$. We give a L-reduction from MAX SOL($\overline{R}$) to MAX AW SOL(Γ_L) with parameters $\beta = 1$ and $\gamma = 1$.

If $b \neq c$, then MAX SOL($\overline{R}$) is **APX**-complete by Lemma 15 and if $b = c$, then MAX SOL($\overline{R}$)-3 is **APX**-complete by Lemma 16. Hence, we prove **APX**-hardness by an L-reduction from either MAX SOL($\overline{R}$) or MAX SOL($\overline{R}$)-3 depending on b and c.

Given an instance $I = (V, D, C, w)$ of MAX SOL($\overline{R}$) (or MAX SOL($\overline{R}$)-3) where all variables have weight 1, let $F(I)$ be the instance of MAX AW SOL(Γ_L) 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 in the instance $F(I)$:

$$C_i = (x_1 \geq b \vee x_2 \geq c).$$

By Lemma 17 the assignment s satisfies C_i if and only if the assignment $\overline{s} = G(I, s)$, satisfies $\overline{C_i} = (x_1 \leq d - b \vee x_2 \leq d - c)$. Moreover, given any solution s to $F(I)$, we have $m(F(I), s) = m(I, G(I, s)) - |V|d$ so $\text{OPT}(F(I)) = \text{OPT}(I) - |V|d$ and thus $\beta = 1$ is a valid parameter in the L-reduction. By these identities we have:

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

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

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 26. *If $(-a_1, \ldots, -a_p, +b_1, \ldots, +b_q) \in L$ and $q \geq 2$, then* MAX AW SOL(Γ_L) *is* **APX**-*hard.*

Proof. If $p = 0$, then the result follows from Lemma 25 and if $p \geq 2$, then the result follows from Lemma 24 so we can assume that $p = 1$. By Lemma 5, it is sufficient to consider the constraint language $L = \{(-a, +b, +c)\}$.

To prove that MAX AW SOL(Γ_L) is **APX**-hard, we will give a L-reduction from **APX**-complete problem MIN SOL($\Gamma_{(+b,+c)}$)-11. Given an instance $I = (V, D, C, w)$ of MIN SOL($\Gamma_{(+b,+c)}$)-11 (where all variables have weight 1 and $D = \{0, \ldots, d\}$), let $F(I) = (V', C', D, w')$ be the 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

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

It is easy to verify that $\text{OPT}(F(I)) = 2d^2|Y| - \text{OPT}(I)$. To see this, assume that s is an optimal solution to $F(I)$ where $s(y) < d$ for a variable $y \in Y$. Let $(y \leq a \vee x_i \geq$

$b \vee x_j \geq c$) 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. We note that $\text{OPT}(F(I)) = 2d^2|Y| - \text{OPT}(I)$.

Since no variable in I occurs more than 11 times, we have $\text{OPT}(I) \geq |V|/12$ and the number of constraints in I is at most $\frac{11|V|}{2}$ (i.e., $|Y| \leq \frac{11|V|}{2}$). Since $\text{OPT}(F(I)) = 2d^2|Y| - \text{OPT}(I)$, we have $\text{OPT}(F(I)) \leq 2d^2 \cdot \frac{11|V|}{2} - \text{OPT}(I)$, and using the fact that $\text{OPT}(I) \geq |V|/12$ gives us $\text{OPT}(F(I)) \leq 132d^2 \cdot \text{OPT}(I)$. Thus, $\beta = 132d^2$ is a valid parameter in the L-reduction.

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

$$\begin{aligned}
&m(I, G(I, s)) - \text{OPT}(I) \leq \\
&2d|Y'| + \sum_{x \in V \setminus X'} s(x) - \text{OPT}(I) \leq \\
&2d|Y'| + \sum_{x \in V} s(x) - \text{OPT}(I) \leq \\
&2d^2|Y| - 2d \sum_{y \in Y} s(y) + \sum_{x \in V} s(x) - \text{OPT}(I) = \\
&\text{OPT}(F(I)) - m(F(I), s)
\end{aligned}$$

and $\gamma = 1$ is a valid parameter in the L-reduction. □

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 [8, 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 $\text{SOL}(\Gamma_L)$ is polynomial-time solvable by Lemma 23. We also note that given a clausal language L, it is obvious how to check (in polynomial time) whether MAX AW $\text{SOL}(\Gamma_L)$ is polynomial-time solvable or not.

Appendix References

16. P. Alimonti and V. Kann. Some APX-completeness results for cubic graphs. *Theoretical Computer Science*, 237:123–134, 2000.
17. R. Bar-Yehuda and S. Even. A local-ratio theorem for approximating the weighted vertex cover problem. *Annals of Disc. Math.*, 25:27–46, 1985.
18. 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.
19. 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.