Enumerating All Solutions of a Boolean CSP by Non-Decreasing Weight

Nadia Creignou, Frédéric Olive, and Johannes Schmidt

Laboratoire d'Informatique Fondamentale de Marseille, CNRS UMR 6166,
Aix-Marseille Université, 163, avenue de Luminy, F-13288 Marseille Cedex 9, France
[nadia.creignou,frederic.olive,johannes.schmidt]@lif.univ-mrs.fr

Abstract. We address the problem of enumerating all models of Boolean formulæ in order of non-decreasing weight in Schaefer's framework. The weight of a model is the number of variables assigned to 1. Tractability in this context amounts to enumerating all models one after the other in sorted order, with polynomial delay between two successive outputs. The question of model-enumeration has already been studied in Schaefer's framework, but without imposing a specific order. The order of non-decreasing weight changes the complexity considerably. We obtain a new dichotomous complexity classification. On the one hand, we develop new polynomial delay algorithms for Horn and 2-XOR-formulæ to enumerate the models by non-decreasing weight. On the other hand, we prove that in all other cases such a polynomial delay algorithm does not exist, unless $\mathrm{P} = \mathrm{NP}$.

Keywords: Enumeration, complexity, polynomial delay, generalized satisfiability, CSP.

1 Introduction

This paper is concerned with algorithmic and complexity of *enumeration*, the task of generating all solutions of a given problem. The area of enumeration algorithms has experienced tremendous growth over the last decade. This is motivated by the explosion in the size of the data that algorithms are called upon to process in everyday applications. The prime application is query answering in databases, where huge answer sets arise naturally. Computing queries incrementally and efficiently has become an increasingly important issue. For instance users of web search engines want to obtain the first results of their keyword search as quickly as possible. Other application domains include constraint solving, operations research, data mining, Web mining, bioinformatics and computational linguistics (see e.g. [17,8,1]).

Because of the amount of solutions that enumeration algorithms possibly produce, the size of their output is often much larger (e.g. exponentially larger) than the size of their input. Therefore, polynomial time complexity is not a suitable yardstick of efficiency when analyzing their performance. Actually, one would be interested in the regularity of these algorithms rather than in their total running time. For this reason, *polynomial delay* is customarily regarded as the good notion of tractability for enumeration complexity: an enumeration algorithm has polynomial delay $p(n)$ if the elapsed time between two successive outputs is polynomial in the size of the input.

Since the seminal result of Schaefer [19], the theoretical interest of the CSP point of view on complexity questions has been largely assessed (see e.g. [5]). By offering a unified framework which has intimate connections with various problems in database, it allowed numerous classification results for a variety of computational tasks, see [7] for a survey. The present paper refers to this line of research.

In the context of non-uniform Boolean CSP we fix a constraint language Γ, which is a finite set of Boolean relations. A Γ-formula is then a conjunction of clauses where the form of the clauses is restricted by Γ. Thus the problem of enumeration can be phrased as follows: given a Γ-formula, can we efficiently enumerate all its models? Prior works handled that question. In [4], Creignou and Hébrard proved a first classification result about enumeration for Γ-formulæ: if Γ is Horn, dual-Horn, bijunctive or affine, there exists a polynomial delay algorithm that enumerates all models of any Γ-formula; otherwise, such an algorithm does not exist unless $\mathrm{P} = \mathrm{NP}$. But their result ignores an important feature in the design of enumeration algorithms: the specification of the order in which we wish the solutions to be output. This is a fundamental aspect of enumeration because in many cases we cannot afford to enumerate all the solutions, but rather we want to produce the most "important" ones in some *metric*. In other cases we need to find a solution that satisfies some other complicated side conditions and thus we generate the solutions in *order of preference* until we find an acceptable one (see some examples in [15,26]). Besides, it turns out that the order affects heavily the complexity. Johnson et al. [10] prove for instance that maximal independent sets of a graph can be enumerated in lexicographical order by a polynomial delay algorithm, while there is no such algorithm for the reverse lexicographical order, unless $\mathrm{P} = \mathrm{NP}$.

In this paper, we specify the order in which we wish the models of Γ-formulæ to be output. We deal with enumeration of models *according to their weight*, which is the number of variables they assign to 1. The weight is a natural parameter in Boolean CSPs [16,18,6], that can be seen as a cost of the assignment. Hence our approach refers to numerous works that focus on enumeration by non-decreasing cost [25,15,26]. Thus, the key problem addressed in this paper is: can one enumerate efficiently all models of a Γ-formula by non-decreasing weight? We answer this question with a dichotomous classification result: If a set of relations Γ is Horn or width-2 affine, there is a polynomial-delay algorithm that generates all models of a Γ-formula by non-decreasing weight. Otherwise such an algorithm does not exist, unless $\mathrm{P} = \mathrm{NP}$. The proof of this theorem reveals new enumeration algorithms for Boolean CSPs, different from the ones developed so far, in particular in the case of Horn formulæ.

The paper is organized as follows. In Sect. 2 we give the relevant material on Boolean CSPs and enumeration algorithms. We also state our main result, Theorem 2. The proof of this theorem is presented throughout the following sections. Section 3 deals with efficient enumeration algorithms in the case where Γ is width-2 affine or Horn. In Sect. 4 we prove the negative part of Theorem 2: for any relation Γ that is neither Horn nor width-2 affine, the existence of a polynomial delay algorithm for enumerating the models of a Γ-formula by non-decreasing weight would imply $\mathrm{P} = \mathrm{NP}$. We conclude and briefly point out open questions in Sect. 5.

2 Material

2.1 Constraint languages and Γ-formulæ

A *logical relation* of arity k is a relation $R \subseteq \{0,1\}^k$. By abuse of notation we do not make a difference between a relation and its predicate symbol. A *constraint*, C, is a formula $C = R(x_1, \ldots, x_k)$, where R is a logical relation of arity k and the x_i's are (not necessarily distinct) variables. If u and v are two variables, then $C[u/v]$ denotes the constraint obtained from C in replacing each occurrence of v by u. If V is a set of variables, then $C[u/V]$ denotes the result of substituting u to every occurrence of every variable of V in C. An assignment m of truth values to the variables *satisfies* the constraint C if $\big(m(x_1), \ldots, m(x_k)\big) \in R$. A *constraint language* Γ is a finite set of logical relations. A *Γ-formula* φ, is a conjunction of constraints using only logical relations from Γ and is hence a quantifier-free first order formula. With $\mathrm{Var}(\varphi)$ we denote the set of variables appearing in φ. A Γ-formula φ is satisfied by an assignment $m : \mathrm{Var}(\varphi) \to \{0,1\}$ if m satisfies all constraints in φ simultaneously (such a satisfying assignment is also called a *model* of φ). Assuming a canonical order on the variables we can regard models as tuples in the obvious way and we do not distinguish between a formula φ and the logical relation R_φ it defines, i.e., the relation consisting of all models of φ.

Throughout the text we refer to different types of Boolean relations following Schaefer's terminology [19]. We say that a Boolean relation R is *Horn* (resp. *dual Horn*) if R can be defined by a CNF formula which is Horn (resp. dual Horn). A relation R is *bijunctive* if it can be defined by a 2-CNF formula. A relation R is *affine* if it can be defined by an *affine* formula, i.e., conjunctions of XOR-clauses (consisting of an XOR of some variables plus maybe the constant 1) — such a formula may also be seen as a system of linear equations over GF[2]. A relation is *affine with width* 2 (width-2 affine, for short) if it is definable by a conjunction of clauses, each of them being either a unary clause or a 2-XOR-clause (consisting of an XOR of 2 variables plus maybe the constant 1) — such a conjunctive formula may also be seen as a system of linear equations over GF[2] with at most two variables per equation. A relation R is *0-valid* (resp., *1-valid*) if $R(0, \ldots, 0) = 1$ (resp., $R(1, \ldots, 1) = 1$). Finally, a constraint language Γ is Horn (resp. dual Horn, bijunctive, affine, width-2 affine, 0-valid, 1-valid) if every relation in Γ is Horn (resp. dual Horn, bijunctive, affine, width-2 affine). We say that a constraint language is *Schaefer* if Γ is either Horn, dual Horn, bijunctive, or affine.

There exist easy criteria to determine if a given relation is Horn, dual Horn, bijunctive, or affine. Indeed all these classes can be characterized by their polymorphisms (see e.g. [7] for a detailed description). We recall some of these properties here briefly for completeness. The operations of conjunction, disjunction, and addition applied on k-ary Boolean vectors are applied coordinate-wise.

- R is Horn if and only if $m, m' \in R$ implies $m \wedge m' \in R$.
- R is dual Horn if and only if $m, m' \in R$ implies $m \vee m' \in R$.
- R is affine if and only if $m, m', m'' \in R$ implies $m \oplus m' \oplus m'' \in R$.
- R is affine and 0-valid if and only if $m, m' \in R$ implies $m \oplus m' \in R$.

The satisfiability problem for Γ formulæ, denoted by $\text{SAT}(\Gamma)$, was first studied by Schaefer [19] who obtained a famous dichotomous classification: If Γ is Schaefer or 0-valid or 1-valid, then $\text{SAT}(\Gamma)$ is in P; otherwise $\text{SAT}(\Gamma)$ is NP-complete. The complexity of finding a non-trivial solution (i.e., a solution different from all-zero and all-one), $\text{SAT}^*(\Gamma)$, was studied in [4]: If Γ is Schaefer, then $\text{SAT}^*(\Gamma)$ is in P; otherwise $\text{SAT}^*(\Gamma)$ is NP-complete. Since then and in the recent past, complexity classifications for many further computational problems for Γ-formulæ have been obtained (see [7] for a survey).

2.2 Enumeration

In this paper, we focus on enumeration by non-decreasing weight of the models of Boolean constraint formulæ, the weight of an assignment being the number of variables assigned to 1. The corresponding problem can be displayed as follows:

Problem:	$\text{ENUM-SAT}_w(\Gamma)$
Input:	a Γ-formula φ.
Output:	generate all models of φ by non-decreasing weight.

We say that an algorithm $\mathcal{A}$ computes the enumeration problem $\text{ENUM-SAT}_w(\Gamma)$ if for a given input φ, $\mathcal{A}$ generates one by one the models of φ, by non-decreasing weight, without repetition, and stops after writing the last one.

Polynomial time complexity is not a suitable yardstick of efficiency when analyzing an enumeration algorithm since the output size is usually exponential in the size of the input. In [10] several notions of efficiency are discussed. The least we could ask is that the enumeration algorithm runs in *polynomial total time*, that is that the time required to output all solutions be polynomial in the size of the input and in the number of solutions (i.e., the size of the output). This notion is also referred to as *output polynomial*. An important feature of an enumeration algorithm is the ability to start generating configurations as soon as possible, and more generally to generate configurations in a regular way with a limited delay between two successive outputs. Hence we say that an enumeration algorithm runs in *polynomial delay* if the delay until the first solution is output and thereafter the delay between any two consecutive solutions is bounded by a polynomial $p(n)$ in the input size. It is worth noticing that such an algorithm generates the first k outputs in time $k \cdot p(n)$. This is an important property of polynomial delay algorithms, since when one has not enough time to enumerate all solutions, at least the first k solutions (the top-k-ranked in the case of an enumeration in a ranked order) can be efficiently enumerated. If $\text{ENUM-SAT}_w(\Gamma)$ is computable by a polynomial delay algorithm, we write $\text{ENUM-SAT}_w(\Gamma) \in \text{delayP}$.

For characterizing space efficiency we ignore the amount of space needed for writing the output, only the space used for storing intermediate results is measured. Enumeration algorithms are sometimes required to run in *polynomial space*, which means that the amount of space involved during the whole computation is polynomial in the size of the input. This requirement is restrictive, even for polynomial delay algorithms. Indeed there are polynomial delay algorithms, especially when a specified order is required, that build exponentially large data structures (see [10,14,24]). This is also the

case for the polynomial delay algorithm for Horn formulæ, described in Sect. 3. Nevertheless, any polynomial delay algorithm runs in *incremental polynomial space*, which means that the space needed for generating the first k solutions is bounded by k times a polynomial in the input size.

2.3 Main result

The complexity of enumerating all models of generalized Boolean formulæ, without specifying any order, has been studied in [4]. An alternative proof making use of partial polymorphisms was given later in [21].

Theorem 1. *(*Model enumeration [4]*.) If Γ is Schaefer, then there is a polynomial-delay algorithm that generates all models of a Γ-formula. Otherwise such an algorithm does not exist unless* $\mathrm{P} = \mathrm{NP}$.

In this paper we are interested in enumerating all models by non-decreasing weight. Of course, when Γ is Schaefer, Theorem 1 enables to do that in polynomial total time: first, generate all solutions in lexicographic order with the algorithm underlying the proof of this theorem; then, sort the solutions in order to output them by non-decreasing weight. However such a procedure forbids any control on the regularity of the enumeration since for instance, the first solution is output after an exponential amount of time if there is an exponential number of models. Therefore, we have to develop specific techniques to perform efficient enumeration in this order. Sets of relations that admit a good behavior with respect to this task do not coincide with Schaefer's ones. Our main theorem details this situation, stating a new dichotomy result concerning the enumeration of models by non-decreasing weight.

Theorem 2. *(*Enumeration by non-decreasing weight*.) If Γ is Horn or width-2 affine, then there is a polynomial-delay algorithm that generates all models of a Γ-formula by non-decreasing weight. Otherwise such an algorithm does not exist unless* $\mathrm{P} = \mathrm{NP}$.

3 Efficient enumeration algorithms

The efficient enumeration algorithms proposed earlier in [4] (see Theorem 1) were based on self-reducibility ([23,13,20]). The self-reducibility property of a problem allows a "search-reduces-to-decision" algorithm to enumerate the solutions. As a consequence, the models are provided in lexicographical order. Moreover the algorithms use only polynomial space. Enumerating the solutions in order of non-decreasing weight requires new algorithms. The two classes of constraint languages under examination in this section, namely width-2 affine and Horn, invoke different algorithmic approaches. Indeed they differ in the complexity of their associated k-ONES problem, in which, given a formula and an integer k, we want to know whether there exists a model with exactly k ones. For width-2 affine formulæ this problem is in P, whereas for Horn formulæ it is NP-complete (see [6]). As a consequence for width-2 affine formulæ one can use the tractability of the k-ONES problem to get an efficient enumeration algorithm. In contrast, Horn formulæ require another strategy. It is natural to use a data structure

which maintains an ordered set of elements and which supports efficient operations of insertion and extraction. The algorithm we will present for the Horn case makes use of a priority queue in order to produce the right order of the output solutions. This method was already used in e.g. [10,14]. As in these papers, our priority queue allows insertion of elements and extraction of the top element in logarithmic time in the size of the queue. Thus the size of the queue may grow exponentially whereas polynomial delay is still maintained.

Proposition 1. *If Γ is width-2 affine, then there is a polynomial-space polynomial-delay algorithm that generates all models of a Γ-formula by non-decreasing weight.*

Proof. Let Γ be width-2 affine and let φ be a Γ-formula. Without loss of generality we can suppose that φ does not contain unitary clauses. Then each clause of φ expresses either the equality or the inequality between two variables. Using the transitivity of the equality relation and the fact that in the Boolean case $a \neq b \neq c$ implies $a = c$, we can identify equivalence classes of variables such that each two classes are either independent or they must have contrary truth values. We call a pair (A, B) of classes with contrary truth values *cluster*, B may be empty. It follows easily that any two clusters are independent and thus to obtain a model of φ, we choose for each cluster (A, B) either $A = 1, B = 0$ or $A = 0, B = 1$. We suppose in the following that φ is satisfiable (otherwise, we will detect a contradiction while constructing the clusters). Let $n \geq 1$ be the number of clusters, then the number of models will be 2^n. The weight contribution of each cluster to a model is either $|A|$ or $|B|$, where $|A| = |B|$ may occur. We represent a model by an n-tuple $s \in \{0, 1\}^n$, indicating for each cluster which of the two assignments is taken. In the case $|A| \neq |B|$ we indicate by 0 the light assignment and by 1 the heavy assignment. Surely $(0, 0, \ldots, 0)$ will represent a model of minimal weight, and $(1, 1, \ldots, 1)$ will represent a model of maximal weight. For enumeration we may consider only the weight difference $\big||A| - |B|\big|$ of each cluster, since we can subtract the weight of a minimal model. Setting $(w_1, \ldots, w_n)$ to these weight differences of the clusters, we reduce our problem to the following enumeration problem:

Problem:	SUBSET-SUM
Input:	A sequence of non-negative integers $(w_1, \ldots, w_n) \in \mathbb{N}^n$
Output:	generate all n-tuples $s \in \{0, 1\}^n$ by non-decreasing weight $\delta(s)$, where $\delta(s) = \Sigma_{i=1}^{n} s_i \cdot w_i$

To solve this enumeration problem we make use of the fact that in our case the sum of the weights $W := \Sigma_{i=1}^{n} w_i$ is linearly bounded by the number of variables of the original formula φ. This allows a strategy of dynamic programming to compute in polynomial time a matrix $A \in \{0, 1\}^{(n+1,\, W+1)}$ such that $A(i, k) = 1$ if and only if with the weights $w_1, \ldots, w_i$ one can construct the sum k, where $0 \leq i \leq n,\ 0 \leq k \leq W$. The matrix A is constructed by first setting $A(0, 0) = 1$ and $A(0, k) = 0$ for all $k \geq 1$, and then filling the other fields row by row according to the rule $A(i, k) = 1$ if and only if $A(i - 1, k) = 1$ or $A(i - 1, k - w_i) = 1$. Thus the computation of A takes time $O(n \cdot W)$. After this precomputation, for each k for which there is at least one solution of weight k we enumerate all such solutions by constructing the solution strings from ϵ (the empty string) recursively.

Algorithm 1 Algorithm for SUBSET-SUM.

MAIN$(w_1, \ldots, w_n)$

1: compute $A \in \{0,1\}^{(n+1,\ W+1)}$
2: **for** $k = 0$ to W **do**
3: **if** $A(n,k) = 1$ **then**
4: CONSTRUCTSOLUTIONS$(n,\ k,\ \epsilon)$ /* enumerate all solutions of weight k */
5: **end for**

CONSTRUCTSOLUTIONS$(i,\ j,\ s)$

1: **if** $i = 0$ **then**
2: output s
3: **else**
4: **if** $A(i-1, j-w_i) = 1$ **then**
5: CONSTRUCTSOLUTIONS$(i-1,\ j-w_i,\ 1 \circ s)$ /* $\circ$ stands for the concatenation operator */
6: **if** $A(i-1, j) = 1$ **then**
7: CONSTRUCTSOLUTIONS$(i-1,\ j,\ 0 \circ s)$

The reader may convince himself or herself that Algorithm 1 enumerates all solutions s of the SUBSET-SUM problem by non-decreasing weight $\delta(s)$. Since both n and W are linearly bounded by the number of variables of φ, Algorithm 1 has a quadratic precomputation time and a linear delay thereafter. The translations between our original problem and SUBSET-SUM can be performed in polynomial time. □

Proposition 2. *If Γ is Horn, then there is a polynomial-delay algorithm that generates all models of a Γ-formula by non-decreasing weight.*

Proof. Let Γ be Horn and let φ be a Γ-formula. Then φ is equivalent to a conjunction of Horn clauses. We will use a priority queue Q to respect the order of non-decreasing weight and to avoid duplicates. The command Q.enqueue(s, k) enqueues an element s with an integer key-value k (a weight). The queue sorts by non-decreasing key-value and inserts an element s only if it is not yet present in the queue.

For notational convenience we represent a model by the set of variables it sets to 1. We use the well-known fact that for Horn formulæ the intersection of all models is the unique *minimal model* which is polynomial time computable. For a satisfiable Horn formula φ we indicate the minimal model by $mm(\varphi)$. Note that for a set of variables $V \subseteq Vars(\varphi)$ the formula $\varphi \wedge V := \varphi \wedge \bigwedge_{v \in V} v$ is still representable as a Horn formula and thus, if $\varphi \wedge V$ is satisfiable, also $mm(\varphi \wedge V)$ can be computed in polynomial time.

We claim that Algorithm 2 enumerates the models of a given Horn formula with polynomial delay, by non-decreasing weight. The polynomial delay is easily seen. By definition of the priority queue and by the fact that the models m' generated out of m in line 12 are always of bigger weight than m itself, it is also easily seen that the models are output in the right order and that no model is output twice. To prove that no model is omitted, it suffices to show that for every model $m' \neq mm(\varphi)$ there exists a submodel $m \subsetneq m'$ such that in line 12 the algorithm generates m' out of m. That is,

Algorithm 2 Algorithm for HORN-SAT.

Require: φ a Horn formula
1: **if** φ unsatisfiable **then**
2: return 'no'
3: Q = newPriorityQueue
4: $m := mm(\varphi)$
5: Q.enqueue(m, $|m|$)
6: **while** Q not empty **do**
7: m := Q.dequeue
8: output m
9: **for all** $x \in Vars(\varphi) \setminus m$ **do**
10: **if** $\varphi \wedge m \wedge x$ satisfiable **then**
11: $m' := mm(\varphi \wedge m \wedge x)$
12: Q.enqueue(m', $|m'|$)
13: **end for**
14: **end while**

there must be an $x \in m' \setminus m$ such that $m' = mm(\varphi \wedge m \wedge x)$. Consider for this the set $H := \{m \mid m \text{ a model of } \varphi \text{ and } m \subsetneq m'\}$. The set H is not empty since it contains at least the minimal model $mm(\varphi)$. A maximal element m of H fulfills our needs, since it satisfies $m' = mm(\varphi \wedge m \wedge x)$ for any $x \in m' \setminus m$.

Let us finally stress that in contrast to Algorithm 1, Algorithm 2 potentially runs in exponential space. □

4 Hardness results

In this section we investigate the case where Γ is neither Horn nor width-2 affine. Clearly, in order to enumerate the models of a Γ-formula by non-decreasing weight, it is a necessary condition to be able to find the lightest model efficiently. As we will prove, this is not a sufficient condition, we need also to be able to find the second one efficiently. So let us introduce the following problems.

Problem:	MIN-ONES(Γ)
Input:	a Γ-formula φ, an integer W
Question:	Is there a model of φ that assigns 1 to at most W variables?

Problem:	MIN-ONES*(Γ)
Input:	a Γ-formula φ, an integer W
Question:	Is there a model of φ different from all-0 that assigns 1 to at most W variables?

From the classification obtained in [12] for the corresponding optimization problem, one can deduce the following.

Proposition 3. *(*Minimum ones satisfiability [12]*.) If Γ is* 0*-valid or Horn or width-*2 *affine, then* MIN-ONES(Γ) *is in* P*, otherwise* MIN-ONES(Γ) *is* NP*-complete.*

Our main contribution in this section is the following hardness result, which obviously proves that when Γ is neither Horn nor width-2 affine, there is no polynomial delay algorithm that enumerates all models of a Γ-formula in order of non-decreasing weight, unless P = NP.

Proposition 4. *Let Γ be a set of relations which is neither Horn nor width-2 affine. Then* MIN-ONES$^*(\Gamma)$ *is* NP*-complete.*

Proof. If Γ is not Schaefer, then SAT$^*(\Gamma)$ is NP-complete [19] and hence so is the problem MIN-ONES$^*(\Gamma)$. If Γ is not 0-valid, then, since it is neither Horn nor width-2 affine, the result follows from the NP-completeness of MIN-ONES(Γ) (Proposition 3). Therefore, it remains to study sets Γ that are Schaefer and 0-valid but that are neither Horn nor width-2 affine. There are three cases to analyse.

- Γ is bijunctive and 0-valid but neither Horn nor width-2 affine.
- Γ is affine and 0-valid but neither Horn nor width-2 affine.
- Γ is dual Horn and 0-valid but neither Horn nor width-2 affine.

Observe that a 2-CNF formula which is 0-valid is also Horn. So the first case does not occur. Besides, one can easily prove that a 0-valid affine relation which is not Horn cannot be width-2 affine. Therefore the proof of the proposition will be completed when we successively prove the NP-completeness of MIN-ONES$^*(\Gamma)$ for any set Γ such that:

1. Γ is affine and 0-valid but not Horn, or
2. Γ is dual Horn and 0-valid but neither affine nor Horn.

The NP-completness of MIN-ONES$^*(\Gamma)$ for any set Γ fulfilling the description 1 or 2 above is settled, respectively, by the forthcoming Proposition 5 and Proposition 6. □

4.1 Affine, 0-valid, not Horn

In this section we deal with relations that are 0-valid and affine but not Horn. We will prove that for such a relation R, finding a non-all-0 model of minimal weight of an R-formula is NP-hard. In order to do so, we need some technical lemmas. The two first ones are definability results, while the third is a basic hardness result. One of the most successful techniques to obtain results on the complexity of constraints related problems (including enumeration), has been the application of tools from universal algebra. A Galois connection relates the expressive power of a constraint language to its set of so-called polymorphisms or partial polymorphisms (see e.g. [9,22]). However here it is not worth using this algebraic tool. The technical results that are needed concern only very restrictive sets of relations and can be obtained "by hand".

In the proofs of all the following lemmas, R will denote a relation of arity k and V a set of k distinct variables, say $V = \{x_1, \ldots, x_k\}$.

Lemma 1. *Let R be a relation which is 0-valid and affine but neither Horn nor 1-valid. Then there exists an R-formula equivalent to $\neg w \wedge (x \oplus y \oplus z = 0)$.*

Proof. Consider the constraint $C = R(x_1, \ldots, x_k)$. Since R is non-Horn there exist m_1 and m_2 in R such that $m_1 \wedge m_2 \notin R$. Since R is 0-valid and affine, we have $m_1 \oplus m_2 \in R$. For $i, j \in \{0, 1\}$, set $V_{i,j} = \{x \mid x \in V, m_1(x) = i \wedge m_2(x) = j\}$. Observe that $V_{0,1} \neq \varnothing$ (respectively, $V_{1,0} \neq \varnothing$), otherwise $m_1 \wedge m_2 = m_2$ (resp.,

$m_1 \wedge m_2 = m_1$), contradicting the fact that $m_1 \wedge m_2 \notin R$. Moreover $V_{1,1} \neq \varnothing$, otherwise $m_1 \wedge m_2 = \vec{0}$, a contradiction. Consider the $\{R\}$-constraint: $M(w,x,y,z) = C[w/V_{0,0},\, x/V_{0,1},\, y/V_{1,0},\, z/V_{1,1}]$. According to the above remark the three variables x, y and z effectively occur in this constraint. Let us examine the set of models of M assigning 0 to w: it contains 0011 (since $m_1 \in R$), 0101 (since $m_2 \in R$), 0110 (since $m_1 \oplus m_2 \in R$) and 0000 (since R is 0-valid). But it does not contain 0001 (since by assumption $m_1 \wedge m_2 \notin R$). Thus it does not contain 0111 either. Indeed, otherwise it would contain $0011 \oplus 0101 \oplus 0111$ (since R is affine), which is equivalent to 0001, a contradiction. From this one can prove that it contains neither 0010 nor 0100 (since $0000 \oplus 0011 \oplus 0010 = 0001$ and $0110 \oplus 0101 \oplus 0100 = 0111$). Note that since R is 0-valid but not 1-valid $C[w/V] \equiv \neg w$. Hence, let us consider

$$\varphi(w,x,y,z) = C[w/V] \wedge M(w,x,y,z).$$

The R-formula φ is equivalent to $\neg w \wedge (x \oplus y \oplus z = 0)$, thus concluding the proof. □

Lemma 2. *Let R be a relation which is 0-valid, 1-valid, affine but not Horn. Then there exists an R-formula equivalent to $(w \oplus x \oplus y \oplus z = 0)$.*

Proof. Observe that an affine relation R which is both 0-valid and 1-valid is necessarily complementive, i.e. for all $m \in R$ we have also $\vec{1} \oplus m \in R$. We can mimic the analysis made in the previous lemma and consider the constraint $M(w,x,y,z) = C[w/V_{0,0}, x/V_{0,1}, y/V_{1,0}, z/V_{1,1}]$. Thus, the formula $\varphi(w,x,y,z) = M(w,x,y,z) \wedge M(w,y,z,x) \wedge M(w,z,x,y)$ verifies $\varphi(w,x,y,z) \equiv (w \oplus x \oplus y \oplus z = 0)$. □

Lemma 3. MIN-ONES$^*(x \oplus y \oplus z = 0)$ *and* MIN-ONES$^*(w \oplus x \oplus y \oplus z = 0)$ *are NP-complete.*

Proof. Consider a homogeneous linear system over the finite field GF(2). Finding the non-all-0 solution with minimum weight of such a system is known to be NP-hard (see [2, Theorem 4.1]). In order to prove the lemma we have to show that this problem remains hard when restricted to systems that have three (resp., four) variables by equation. Let S be a homogeneous linear system over GF(2). Suppose that S has n variables, $x_1, \ldots, x_n$. In order to reduce the number of variables in each equation we introduce auxiliary variables. If there is an equation $x_{i_1} \oplus x_{i_2} \oplus \cdots \oplus x_{i_k} = 0$ for some $k \geq 4$, we introduce a new variable y_{i_1,i_2} and replace the original equation by the two equations $y_{i_1,i_2} \oplus x_{i_1} \oplus x_{i_2} = 0$ and $y_{i_1,i_2} \oplus x_{i_3} \oplus \ldots \oplus x_{i_k} = 0$. We repeat this process until all equations have three variables. The satisfiability is preserved during this transformation. The number of auxiliary variables is bounded from above by the number of occurrences of variables in the original system. In order to keep the information on the weight of the solutions we need to introduce enough copies of the original variables, which make the auxiliary variables neglectable. Let N be the number of occurrences of variables in S. Let f be a fresh variable that will play the role of the constant 0. For each $i = 1, \ldots, n$, we introduce N copy-variables $x_i^1, \ldots, x_i^N$ of x_i and add the equations $x_i \oplus x_i^j \oplus f = 0$ for $j = 1, \ldots N$. Finally we add the equation $f \oplus f \oplus f = 0$, i.e., $f = 0$ (this will ensure that $x_i = x_i^j$ for all j). There is a one-to-one correspondence between the solutions of S and the solutions of the so-obtained system S'. Moreover S has a non-trivial

solution of weight at most W if and only if S' has a non-trivial solution of weight at most $W(N+1)+N$. Since the system S' can be seen as an $(x \oplus y \oplus z = 0)$-formula we have thus proved the NP-hardness of MIN-ONES$^*(x \oplus y \oplus z = 0)$.

Let us now reduce MIN-ONES$^*(x \oplus y \oplus z = 0)$ to MIN-ONES$^*(w \oplus x \oplus y \oplus z = 0)$. Let S be a homogeneous linear system over n variables such that each equation has exactly three variables. Let w and w_i for $i = 1, \ldots, n+1$ be fresh variables. Transform S into S' as follows: transform every equation $x \oplus y \oplus z = 0$ into $w \oplus x \oplus y \oplus z = 0$ and add the $n+1$ equations $w \oplus w \oplus w \oplus w_i = 0$ for $i = 1, \ldots, n+1$. Solutions of S' assigning 0 to w coincide with the solutions of S. Moreover any solution of S' assigning 1 to w has weight at least $n+1$. Therefore, S has a non-trivial solution of weight at most W ($W \leq n$) if and only if S' has a non-trivial solution of weight at most W. This completes the proof. □

Proposition 5. *If R is 0-valid and affine but not Horn, then the problem* MIN-ONES$^*(R)$ *is* NP*-complete.*

Proof. It follows from the three lemmas above: if R is not 1-valid, then Lemma 1 allows a reduction from MIN-ONES$^*(x \oplus y \oplus z = 0)$ to MIN-ONES$^*(R)$ (replace each constraint $(x \oplus y \oplus z = 0)$ by the R-formula equivalent to $\neg w \wedge (x \oplus y \oplus z = 0)$, where w is a fresh variable). If the relation R is 1-valid, then Lemma 2 allows a reduction from MIN-ONES$^*(w \oplus x \oplus y \oplus z = 0)$ to MIN-ONES$^*(R)$. In both cases one can conclude with Lemma 3. □

4.2 Dual Horn, 0-valid, neither affine nor Horn

In this section we deal with relations that are 0-valid and dual Horn but neither affine nor Horn. The method of proof is not the same as in the previous section. We will also need some intermediate lemmas.

Lemma 4. *Let R be a relation which is 0-valid, dual Horn but neither affine nor 1-valid. Then there exists an R-formula equivalent to $\neg t \wedge (u \rightarrow v)$.*

Proof. Consider the constraint $C = R(x_1, \ldots, x_k)$. Observe that since R is 0-valid but not 1-valid, $C[t/V] \equiv \neg t$. Since R is 0-valid and non-affine there exist two distinct tuples m_1 and m_2 in R such that $m_1 \oplus m_2 \notin R$. Since R is dual Horn, we have $m_1 \vee m_2 \in R$. For $i, j \in \{0,1\}$, let $V_{i,j} = \{x \mid x \in V, m_1(x) = i \wedge m_2(x) = j\}$. Observe that $V_{1,1} \neq \varnothing$, otherwise $m_1 \vee m_2 = m_1 \oplus m_2$, a contradiction. Moreover, since $m_1 \neq m_2$ either $V_{0,1}$ or $V_{1,0}$ is nonempty. Suppose first that they are both nonempty. Consider the R-constraint $M(w,x,y,z) = C[w/V_{0,0}, x/V_{0,1}, y/V_{1,0}, z/V_{1,1}]$. The three variables x, y and z effectively appear in this constraint. Let us examine the set of models of M assigning 0 to w: it contains 0011 (since $m_1 \in R$), 0101 (since $m_2 \in R$), 0111 (since $m_1 \vee m_2 \in R$) and 0000 (since R is 0-valid), but does not contain 0110 (since by assumption $m_1 \oplus m_2 \notin R$). The membership of 0100, 0010, 0001 is open:

- If it does not contain 0100, then consider the R-formula $\varphi(t,u,v) := C[t/V] \wedge M(t,u,v,v)$. Its set of models is $\{001, 011, 000\}$ and therefore, $\varphi(t,u,v) \equiv \neg t \wedge (u \rightarrow v)$.

– If it contains 0100, then it does not contain 0010. Indeed otherwise, since R is dual-Horn it would also contain 0110, which provides a contradiction. Thus consider the R-formula $\varphi(t,u,v) := C[t/V] \wedge M(t,v,u,v)$. Its set of models is $\{001, 011, 000\}$ and therefore, $\varphi(t,u,v) \equiv \neg t \wedge (u \to v)$.

If for instance $V_{0,1} = \varnothing$, then consider $M(w,y,z) = C[w/V_{0,0}, y/V_{1,0}, z/V_{1,1}]$. In this case $\varphi(t,u,v) := C[t/V] \wedge M(t,u,v)$ is equivalent to $\neg t \wedge (u \to v)$. □

Lemma 5. *Let R be a relation which is 0-valid, 1-valid, dual Horn but not Horn, then there exists an R-formula equivalent to $(u \to v)$.*

Proof. Since R is dual Horn but non Horn, it is non-complementive, that is, there exists an $m \in R$ such that $m \oplus \vec{1} \notin R$. Consider the constraint $C = R(x_1, \ldots, x_k)$. For $i \in \{0,1\}$, let $V_i = \{x \mid x \in V \wedge m(x) = i\}$. Consider the R-formula $\varphi(u,v) = C[u/V_0, v/V_1]$. Then $\varphi(u,v) \equiv \neg u \vee v \equiv u \to v$. □

Proposition 6. *If R is 0-valid and dual Horn but neither affine nor Horn, then the problem* MIN-ONES$^*(R)$ *is* NP*-complete.*

Proof. Let R be a relation which is 0-valid and dual Horn but neither affine nor Horn. Let T be the constant unary relation $T = \{1\}$. According to Proposition 3, the problem MIN-ONES(R,T) is NP-complete. We reduce MIN-ONES(R,T) to MIN-ONES$^*(R)$. Let φ be an $\{R,T\}$-formula, $\varphi = \psi \wedge \bigwedge_{x \in V} T(x)$ where ψ is an R-formula. Let t be a fresh variable and consider

$$\varphi' = \psi[t/V] \wedge \bigwedge_{x \in \mathrm{Var}(\varphi) \setminus V} x \to t.$$

Observe that the only solution that assigns 0 to t in φ' is the all-0 one. Therefore it is clear that φ has a solution of weight at most W ($W \geq \mid V \mid$) if and only if φ' has a non-trivial solution of weight at most $W - \mid V \mid + 1$. The two above lemmas allow to express φ' as an R-formula (modulo the introduction of an additional variable that will always take the value 0 when R is not 1-valid), thus concluding the proof. □

5 Conclusion

We have classified the complexity of enumerating all models of a Γ-formula by non-decreasing weight. We have proved that in the case of Boolean CSPs a necessary and sufficient condition for enumerating all solutions in order of non-decreasing weight with polynomial delay is the ability to efficiently find a non-all-zero solution of minimal weight. Note that by duality, under the assumption P $\neq$ NP, one can enumerate the models of a Γ-formula by *non-increasing* weight with polynomial delay if and only if Γ is width-2 affine or dual Horn.

Another related question is: When does exist a so-called *polynomial time iterator* for the solutions' weight? That is, given a model m, when are we able to efficiently compute a model of the next weight level? In the width-2 affine case this task is tractable since k-ONES is tractable. In the Horn case this task becomes NP-hard: If it were

tractable, by iteration we would be able to efficiently compute a model of maximal weight, which is NP-hard [12]. In the remaining cases, that is when enumeration by non-decreasing weight can not be done with polynomial delay unless $\mathrm{P} = \mathrm{NP}$, both situations may occur. A complete classification might reveal new classes of formulæ interesting in the context of enumeration.

We could also have dealt with the weighted version of the problem, i.e., we have a weight function $w\colon V \to \mathbb{N}$ and the weight of a model m is given by $\sum_{v \in V} w(v) \cdot m(v)$. The algorithm proposed in Proposition 2 for Horn formulæ could also tackle this variant. But the algorithm proposed in Proposition 1 for width-2 affine formulæ does not run in polynomial delay when the weights are not polynomially bounded. However, for cost of exponential space, one can also construct a polynomial delay algorithm for this case, using a priority queue in a similar way as in the Horn case.

Asking the enumeration to be performed in order of non-decreasing weight has revealed new enumeration algorithms for Boolean CSPs, different from the ones developed so far. The algorithm developed for Horn formulæ requires potentially an exponential amount of space. Ideally we would like to avoid this. Interesting open questions are to find out whether there is a space/delay trade-off and whether the exponential space requirement is inherent to the order of non-decreasing weight for Horn formulæ.

Another interesting direction of research is the study of enumeration in order of non-decreasing weight for CSPs over arbitrary finite domains. Enumeration of all solutions of such CSPs has been studied in [3,21]. As mentioned in [21] considering different orderings could be the key to discover further enumeration algorithms. Also, in [11] the authors studied the complexity of the so-called MAX-SOL(Γ) problem, which generalizes the MAX-ONES problem to arbitrary finite domains. They identified two tractable classes of constraint languages, namely injective and generalized max-closed constraint languages. These two classes can be seen as substantial and nontrivial generalizations of the tractable classes known for the MAX-ONES problem over the Boolean domain, namely width-2 affine, 1-valid and dual-Horn. It would be interesting to examine whether these classes give rise to polynomial delay algorithms for the enumeration of solutions by non-increasing weight.

Acknowledgment

We thank Arnaud Durand for providing us with the reference [2]. This work was partially supported by the project ANR ENUM (ANR 07-BLAN-0327-04).

References

1. G. Bagan, A. Durand, and E. Grandjean. On acyclic conjunctive queries and constant delay enumeration. In J. Duparc and T. A. Henzinger, editors, *CSL*, volume 4646 of *LNCS*, pages 208–222. Springer, 2007.
2. A. Barg. Complexity issues in coding theory. *Electronic Colloquium on Computational Complexity (ECCC)*, 4(46), 1997.
3. D. A. Cohen. Tractable decision for a constraint language implies tractable search. *Constraints*, 9(3):219–229, 2004.

4. N. Creignou and J.-J. Hébrard. On generating all solutions of generalized satisfiability problems. *Theoretical Informatics and Applications*, 31(6):499–511, 1997.
5. N. Creignou, Ph. G. Kolaitis, and H. Vollmer, editors. *Complexity of Constraints - An Overview of Current Research Themes*, volume 5250 of *LNCS*. Springer, 2008.
6. N. Creignou, H. Schnoor, and I. Schnoor. Nonuniform boolean constraint satisfaction problems with cardinality constraint. *ACM Trans. Comput. Log.*, 11(4), 2010.
7. N. Creignou and H. Vollmer. Boolean constraint satisfaction problems: When does Post's lattice help? In Creignou et al. [5], pages 3–37.
8. M. Hagen. Lower bounds for three algorithms for transversal hypergraph generation. *Discrete Applied Mathematics*, 2009.
9. P. G. Jeavons. On the algebraic structure of combinatorial problems. *Theoretical Computer Science*, 200:185–204, 1998.
10. D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis. On generating all maximal independent sets. *Inf. Process. Lett.*, 27(3):119–123, 1988.
11. P. Jonsson and G. Nordh. Introduction to the maximum solution problem. In Creignou et al. [5], pages 255–282.
12. S. Khanna, M. Sudan, and D. Williamson. A complete classification of the approximability of maximization problems derived from Boolean constraint satisfaction. In *Proceedings 29th Symposium on Theory of Computing*, pages 11–20. ACM Press, 1997.
13. S. Khuller and V. V. Vazirani. Planar graph coloring is not self-reducible, assuming $\mathrm{P} \neq \mathrm{NP}$. *Theoretical Computer Science*, 88(1):183–189, 1991.
14. B. Kimelfeld and Y. Sagiv. Incrementally computing ordered answers of acyclic conjunctive queries. In O. Etzion, T. Kuflik, and A. Motro, editors, *NGITS*, volume 4032 of *LNCS*, pages 141–152. Springer, 2006.
15. B. Kimelfeld and Y. Sagiv. Efficiently enumerating results of keyword search over data graphs. *Inf. Syst.*, 33(4-5):335–359, 2008.
16. A. A. Krokhin and D. Marx. On the hardness of losing weight. In L. Aceto, I. Damgård, L. A. Goldberg, M. M. Halldórsson, A. Ingólfsdóttir, and I. Walukiewicz, editors, *ICALP (1)*, volume 5125 of *LNCS*, pages 662–673. Springer, 2008.
17. K. Makino and T. Uno. New algorithms for enumerating all maximal cliques. In *Scandinavian Workshop on Algorithm Theory*, pages 260–272, 2004.
18. D. Marx. Parameterized complexity of constraint satisfaction problems. *Computational Complexity*, 14(2):153–183, 2005.
19. T. J. Schaefer. The complexity of satisfiability problems. In *Proccedings 10th Symposium on Theory of Computing*, pages 216–226. ACM Press, 1978.
20. J. Schmidt. Enumeration: Algorithms and complexity. Preprint (2009), available at `http://www.thi.uni-hannover.de/fileadmin/forschung/arbeiten/schmidt-da.pdf`.
21. H. Schnoor and I. Schnoor. Enumerating all solutions for constraint satisfaction problems. In W. Thomas and P. Weil, editors, *STACS*, volume 4393 of *LNCS*, pages 694–705. Springer, 2007.
22. H. Schnoor and I. Schnoor. Partial polymorphisms and constraint satisfaction problems. In Creignou et al. [5], pages 229–254.
23. C. P. Schnorr. Optimal algorithms for self-reducible problems. In *International Conference on Automata, Languages and Programming*, pages 322–337, 1976.
24. Y. Strozecki. Enumeration complexity and matroid decomposition. *Phd thesis*, 2010.
25. V. Vazirani and M. Yannakakis. Suboptimal cuts: Their enumeration, weight and number. In W. Kuich, editor, *Automata, Languages and Programming*, volume 623 of *Lecture Notes in Computer Science*, pages 366–377. Springer Berlin / Heidelberg, 1992.
26. L.-P. Yeh, B.-F. Wang, and H.-H. Su. Efficient algorithms for the problems of enumerating cuts by non-decreasing weights. *Algorithmica*, 56(3):297–312, 2010.