ELSEVIER

Theoretical Computer Science 288 (2002) 85–100

Theoretical Computer Science

www.elsevier.com/locate/tcs

On the Hamming distance of constraint satisfaction problems ☆

P. Crescenzi, G. Rossi *

Dipartimento di Sistemi e Informatica, Università di Firenze, Via C. Lombroso 6/17, 50134 Firenze, Italy

Abstract

In this paper we consider a new optimization problem, called MAX HAMMINGDISTANCE($\mathscr{F}$) where $\mathscr{F}$ is a family of Boolean constraints. This problem consists in finding two satisfying assignments that differ in the maximum number of variable values: in other words, the problem looks for the maximum difference between two models of the constraints given in input. We give a complete classification of the approximability properties of MAX HAMMINGDISTANCE($\mathscr{F}$) by using a specialization of the criteria introduced by Schaefer in order to classify constraint satisfaction problems and subsequently used by Khanna, Sudan, Trevisan, and Williamson to classify constraint satisfaction optimization problems. © 2002 Elsevier Science B.V. All rights reserved.

Keywords: Satisfiability; Computational complexity; Approximation algorithms

1. Introduction

Many researchers encounter problems which can be modelled as special cases of the so-called constraint satisfaction problem (CSP). For example, the problems of scheduling a collection of tasks, or laying out a silicon chip, or interpreting a visual image, can all be seen in this way. In a CSP, we distinguish one or more variables, each variable having a given (finite or infinite) set of possible values (domain of a variable), and a finite set of constraints on these variables. The problem is to find a feasible variable-value assignment, for example, an assignment of each variable to exactly one element from its domain, such that all constraints are satisfied.

☆ Research partially supported by Italian MURST Project "Algoritmi per Grandi Insiemi di Dati: Scienza ed Ingegneria".

* Corresponding author.

E-mail addresses: piluc@dsi.unifi.it (P. Crescenzi), rossig@dsi.unifi.it (G. Rossi).

PII: S0304-3975(01)00146-3

If both the number of variables and the variable domains are finite, then any CSP can be solved in finite time. Thus the main area of concern when solving these types of problems is doing so in a computationally efficient manner. There are two main families of algorithms used to solve CSPs [10]: backtrackers and constraint propagators. Backtracking involves instantiating each variable (that is, giving it a value from its domain) sequentially, and checking to see if the set of instantiated variables satisfies all constraints involving the variables instantiated so far. Constraint propagation, instead, involves propagating constraints between different variables, to derive a simpler problem: in some cases (depending on the problem and the degree of constraint propagation applied), the resulting CSP is so simple that its solution can be found without any search.

Although the performance of backtracking and constraint propagation can be improved by means of several techniques, the running time of the resulting algorithms is still exponential in the general case. This is due to the intrinsic computational complexity of CSPs. Indeed, it is well-known that even 3-SATISFIABILITY, that is, the CSP restricted to binary domains with constraints formed by the disjunction of at most three (negated) variables, is NP-complete [4] and, hence, does not admit a polynomial-time algorithm (unless $P = NP$).

There exist, however, special cases of the CSP which are efficiently solvable. For example, if we consider binary domains and constraints formed by the disjunction of at most two (negated) variables, then the corresponding CSP (called 2-SATISFIABILITY) is solvable in polynomial time.

The need for a unified study of the computational complexity of the CSP is then quite natural. In this paper, we restrict ourselves to the so-called Boolean CSPs, that is, CSPs in which each constraint belongs to a finite set $\mathscr{F}$ of Boolean functions and an instance of the problem consists of a finite number of constraint applications. The complete classification of this kind of CSPs has been given by Schaefer [14] who describes six classes of function families, such that if $\mathscr{F}$ is a subset of one of these classes, then the corresponding CSP is solvable in polynomial time, otherwise it is NP-complete: this classification is presented in Section 2.

In several applications, not all solutions (that is, variable assignments) of an instance of a CSP are equally good and an optimization function on the solutions is defined. In these cases, the problem consists of finding the best solution among all feasible ones: since this kind of problems is obtained by a CSP plus an optimization criterion, we call it constraint satisfaction optimization problem (CSOP). The solution of a CSOP may use techniques suitable for optimization problems, such as branch-and-bound or genetic algorithms, but it should be clear that a CSOP cannot be easier to solve than the underlying CSP. The advantage of defining an optimization function on the solutions, however, is that in this case, whenever determining an optimal solution is extremely time consuming due to the inherent complexity of the CSOP, we can restrict ourselves to compute an approximate solution.

In this paper, we are interested in the so-called performance guarantee approximate solutions, that is, solutions whose performance ratio with respect to the optimal solution

is guaranteed to be bounded by a function of the size of the instance of the problem (see, for example, [1]). In particular, we will consider two CSOPs: the maximum ones problem and the maximum Hamming distance problem.

The maximum ones problem consists in finding a satisfying assignment with the maximum number of variables whose value has been set to 1. This problem has been studied in [9] where a complete classification of its approximability properties is given: this classification is presented in Section 3. It is interesting to observe that this result is based on the same classes of function families introduced by Schaefer: indeed, the only difference is that two of these classes have to be split into two subclasses. This is a sign that Schaefer's criteria are sufficiently robust to classify CSOPs even though they were originally introduced to classify CSPs.

In corrobation of the above statement, we consider in this paper a new optimization problem, called MAX HAMMINGDISTANCE($\mathscr{F}$) where $\mathscr{F}$ is a constraint family. This problem consists in finding two satisfying assignments that differ in the maximum number of variable values (see Section 4): in other words, the problem looks for the maximum difference between two worlds consistent with the constraints given in input. In Section 5 we give a complete classification of the approximability properties of MAX HAMMINGDISTANCE($\mathscr{F}$) by using, once again, a specialization of Schaefer's criteria (see also Fig. 2). In particular, even though this new problem is strictly related to the maximum ones problem, our classification leads to the definition of three new criteria and does not make use of three criteria that appear in the classification of the maximum ones problem.

Finally, we recall that in [7–9] classification results based on similar criteria have been obtained both for minimization CSPs (such as the minimum ones problem) and for partial CSPs. These latter problems consist of finding an assignment that maximizes (respectively, minimizes) the number of satisfied (respectively, unsatisfied) constraints.

1.1. Preliminaries

We assume the reader to be familiar with basic notions of computational complexity theory (see [2, 3, 12]). We now review some definitions related to CSPs and approximation theory.

Constraints, constraint families, and satisfiability: A (k-ary) *Boolean constraint* is a function $f : \{0,1\}^k \to \{0,1\}$. A constraint f is *satisfied* by an assignment $\tau \in \{0,1\}^k$ if $f(\tau) = 1$. A *Boolean constraint family* $\mathscr{F}$ is a finite collection of Boolean constraints. A *constraint application* of a Boolean constraint f on n Boolean variables is a pair $\langle f, \langle i_1, \ldots, i_k \rangle \rangle$ where i_j belongs to $\{1, \ldots, n\}$ and indicates the variable that is the jth input of f, for $j = 1, \ldots, k$. A constraint application $\langle f, \langle i_1, \ldots, i_k \rangle \rangle$ on n Boolean variables is satisfied by an assignment $\tau \in \{0,1\}^n$ if the restriction of τ to the indices $i_1, \ldots, i_k$ satisfies f. A collection of constraint applications on the same set of n variables is satisfied by an assignment $\tau \in \{0,1\}^n$ if each constraint application is satisfied by τ.

Approximability, reductions and completeness: The basic ingredients of an NP optimization problem are the set of *instances*, the set of *feasible solutions* associated to any instance, and the integer *measure* $\mathtt{m}(x, y)$ defined for any instance x and for any feasible solution y. The problem is specified as a maximization problem or a minimization problem depending whether its goal is to find a solution whose measure is maximum or minimum (in the following $\mathtt{opt}$ will denote the function mapping an instance x to the measure of an optimal solution). The *class* NPO is the set of all NP optimization problems (for a formal definition of this class see [1, 3]). An NPO problem A belongs to the *class* PO if there exists a polynomial-time algorithm T that, for any instance x of A, returns an optimal solution for x.

Let A be an NPO problem. Given an instance x and a feasible solution y of x, we define the *performance ratio of y with respect to x* as

$$R(x, y) = \max\left\{\frac{\mathtt{m}(x, y)}{\mathtt{opt}(x)}, \frac{\mathtt{opt}(x)}{\mathtt{m}(x, y)}\right\}.$$

Let T be an algorithm that, for any instance x of A, returns a feasible solution $T(x)$. Given an arbitrary function $r : N \to (1, \infty)$, we say that T is an *$r(n)$-approximate algorithm for A* if, for any instance x, the performance ratio of the feasible solution $T(x)$ with respect to x verifies the following inequality:

$$R(x, T(x)) \leqslant r(|x|).$$

Given a class of functions F, an NPO problem A belongs to the *class* F-APX if there exists an $r(n)$-approximate polynomial-time algorithm T for A, for some function $r \in F$. In particular, APX and POLYAPX will denote the classes F-APX with F equal to the set of constant functions and to the set of polynomials, respectively.

An NPO problem A belongs to the *class* PTAS if there exists an algorithm T such that, for any fixed rational $r > 1$, $T(\cdot, r)$ is a polynomial-time r-approximate algorithm for A.

Let A and B be two NPO problems. A is said to be *L-reducible* to B if there exist two polynomial-time computable functions f and g and two positive constants α and β such that

(i) For any instance x of A, $f(x)$ is an instance of B such that

$$\mathtt{opt}(f(x)) \leqslant \alpha\mathtt{opt}(x).$$

(ii) For any instance x of A and for any feasible solution y of $f(x)$, $g(x, y)$ is a feasible solution of x such that

$$|\mathtt{opt}(x) - \mathtt{m}(x, g(x, y))| \leqslant \beta|\mathtt{opt}(f(x)) - \mathtt{m}(f(x), y)|.$$

The quadruple (f, g, α, β) is said to be an L-reduction from A to B. In [13] it is shown that if A L-reduces to B and $A \notin$ PTAS, then $B \notin$ PTAS.

Let A and B be two NPO problems. A is said to be *A-reducible* to B if there exist two polynomial-time computable functions f and g and a positive constant α such that

(i) For any instance x of A, $f(x)$ is an instance of B.

(ii) For any instance x of A and for any feasible solution y of $f(x)$, $g(x,y)$ is a feasible solution of x.

(iii) For any rational $r>1$, for any instance x of A, and for any feasible solution y of $f(x)$, $R(f(x),y)\leqslant r$ implies $R(x,g(x,y))\leqslant \alpha r$.

The triple (f,g,α) is said to be an A-reduction from A to B. In [5] it is shown that if A A-reduces to B and $A\notin$ APX, then $B\notin$ APX.

An optimization problem A in APX is said to be APX-*complete* if, for any $B\in$ APX, B L-reduces to A. Finally, an optimization problem A in POLYAPX is said to be POLYAPX-*complete* if, for any $B\in$ POLYAPX, B A-reduces to A. Intuitively, the APX-complete (respectively, POLYAPX-complete) problems are the hardest ones within the class APX (respectively, POLYAPX).

2. The Schaefer's theorem

The first classification result we present in this paper is due to Schaefer [14] and deals with the following CSP (observe that the following definition depends on the specific constraint family $\mathscr{F}$).

Definition 1 (SATISFIABILITY($\mathscr{F}$)). Let C be a collection of m constraint applications of the form $\{\langle f_j,\langle i_1(j),\ldots,i_{k_j}(j)\rangle\rangle:\ j=1,\ldots,m\}$, on Boolean variables $X=\{x_1,\ldots,x_n\}$ where $f_j\in\mathscr{F}$ and k_j is the arity of f_j. The SATISFIABILITY($\mathscr{F}$) problem with input C consists in finding an assignment that satisfy all the constraint applications in C.

In order to classify the above problem, Schaefer introduces the following six criteria: [1]

0-*validity*: A Boolean constraint f is 0-*valid* if $f(0,\ldots,0)=1$.

1-*validity*: A Boolean constraint f is 1-*valid* if $f(1,\ldots,1)=1$.

Weakly positivity: A Boolean constraint f is *weakly positive* if f can be expressed as a CNF-formula with at most one negated variable in each clause.

Weakly negativity: A Boolean constraint f is *weakly negative* if f can be expressed as a CNF-formula with at most one positive variable in each clause.

Affinity: A Boolean constraint f is *affine* if it can be expressed as a conjunction of linear equalities in $\mathscr{Z}_2$.

2*CNF*: A Boolean constraint f is 2*CNF* if it can be expressed as a CNF-formula with at most two literals per clause.

[1] Recall that a Boolean formula is in *conjunctive normal form* (CNF) if it is the conjunction of a finite set of *clauses*. A clause is the disjunction of a finite set of *literals*, where a literal is either a variable or the negation of a variable.

A constraint family $\mathscr{F}$ is 0-valid (respectively, 1-valid, weakly positive, weakly negative, affine, and 2CNF) if all its constraints are 0-valid (respectively, 1-valid, weakly positive, weakly negative, affine, and 2CNF).

We are now ready to state Schaefer's result.

Theorem 2 (SATISFIABILITY($\mathscr{F}$) classification, Schaefer [14]). *Given a constraint family* $\mathscr{F}$, SATISFIABILITY($\mathscr{F}$) *is either in* P *or* NP*-complete. In particular, if* $\mathscr{F}$ *is* 0*-valid*, 1*-valid, weakly positive, weakly negative, affine, or 2CNF, then* SATISFIABILITY($\mathscr{F}$) *is in* P. *Otherwise,* SATISFIABILITY($\mathscr{F}$) *is* NP*-complete.*

3. The KSW's theorem

The next classification result deals with a CSOP. In order to define this problem, we introduce the following notation: given an assignment $\tau \in \{0,1\}^n$, $\texttt{ones}(\tau)$ denotes the number of ones in τ.

Definition 3 (MAX ONES($\mathscr{F}$)). Let C be a collection of m constraint applications of the form $\{\langle f_j, \langle i_1(j), \ldots, i_{k_j}(j)\rangle\rangle\colon\ j = 1, \ldots, m\}$, on Boolean variables $X = \{x_1, \ldots, x_n\}$ where $f_j \in \mathscr{F}$ and k_j is the arity of f_j. The MAX ONES($\mathscr{F}$) problem with input C consists in finding an assignment τ that satisfy all the constraint applications in C and such that $\texttt{ones}(\tau)$ is maximum.

In order to classify the above problem, Khanna, Sudan, and Williamson (KSW) introduce the following two new criteria:

Strongly 0*-validity*: A Boolean constraint f is *strongly* 0*-valid* if it is satisfied by all assignments with at most one 1.

Width-2 *affinity*: A Boolean constraint f is *width*-2 *affine* if it can be expressed as a conjunction of linear equalities in $\mathscr{L}_2$ with at most 2 terms.

A constraint family $\mathscr{F}$ is strongly 0-valid (respectively, width-2 affine) if all its constraints are strongly 0-valid (respectively, width-2 affine).

We are now ready to state KSW's result.

Theorem 4 (MAX ONES($\mathscr{F}$) classification, Khanna et al. [9]). *Given a constraint family* $\mathscr{F}$, MAX ONES($\mathscr{F}$) *is either in* PO *or* APX*-complete or* POLYAPX*-complete or in* NPO *but not approximable to within any factor. In particular, the following hold*:

(i) *If* $\mathscr{F}$ *is* 1*-valid or weakly positive or width*-2 *affine, then* MAX ONES($\mathscr{F}$) *is in* PO.

(ii) *Else if* $\mathscr{F}$ *is affine, then* MAX ONES($\mathscr{F}$) *is* APX*-complete.*

(iii) *Else if* $\mathscr{F}$ *is strongly* 0*-valid or weakly negative or 2CNF, then* MAX ONES($\mathscr{F}$) *is* POLYAPX*-complete.*

(iv) *Else if* $\mathscr{F}$ 0*-valid, then finding a solution of* MAX ONES($\mathscr{F}$) *with positive measure is* NP*-hard.*

(v) *Else finding a feasible solution of* MAX ONES($\mathscr{F}$) *is* NP*-hard.*

4. The maximum Hamming distance problem

Boolean constraint families are often intended to tell us things about the world. These things can be either objects or properties of objects. For instance, let us consider an instance of the *blocks world* problem that is a standard problem in knowledge representation [6]. Assume that the only knowledge we have of the instance is that there are three blocks a, b, and c and that block a is above block c. Clearly, if a block is on another block then it cannot be on the table and at least one block must be on the table. This knowledge can be modelled as follows.

We introduce three Boolean variables x_a, x_b, and x_c that specify whether the block a, b, and c, respectively, is on the table and three Boolean variable x_{ab}, x_{ac}, and x_{bc} that specify whether the first block is on the second one. Then, we define the following collection C of five constraint applications on these six Boolean variables:

- $f_1(x_{ab}, x_{ac}, x_{bc}) = x_{ac} \vee (x_{ab} \wedge x_{bc})$. This constraint states that block a is above block c: indeed, either a is on c or a is on b that, in turn, is on c.
- $f_2(x_a, x_b, x_c) = x_a \vee x_b \vee x_c$. This constraint states that at least one block must be on the table.
- $f_3(x_a, x_{ab}, x_{ac}) = (x_{ab} \vee x_{ac}) \rightarrow \neg x_a$. This constraint states that if block a is on another block, then it cannot be on the table. Constraints f_4 and f_5 are similarly defined for blocks b and c, respectively.

It is easy to see that any assignment to the variables x_a, x_b, x_c, x_{ab}, x_{ac}, and x_{bc} that satisfies C must assign the value 0 to variable x_a. Indeed, as a consequence of our knowledge (that is, block a is above block c), we can infer that block a is not on the table. However, there exist several distinct assignments that satisfy C which correspond to distinct configurations of our blocks world (see Fig. 1).

The question we address in this paper is how we can measure the difference between any pair of these four alternative worlds. In a certain sense, we would like to define a *measure of ignorance* in order to quantify how much we do not know about the real world. In the case of the blocks world, a natural distance between two configurations that are consistent with our knowledge can be defined as the number of legal move actions necessary to transform one configuration into the other (observe that a move action is legal only if both the block being moved and the target location are clear when the action is attempted). Referring to the previous example, the distances between

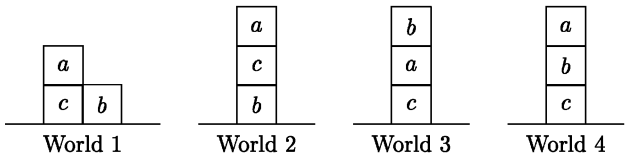

Fig. 1. Four different blocks worlds.

each pair of the four worlds are shown in the following table:

	World 2	World 3	World 4
World 1	3	1	3
World 2		4	4
World 3			4

The move action distance, however, is strictly dependent on the fact that we are dealing with the blocks world, that is, on the semantic of the collection C of constraint applications. The following definition introduces a measure of ignorance that, instead, can be used for *any* collection of constraint applications, independently of the interpretation of its satisfying assignments.

Definition 5 (*Hamming distance between assignments*). Given a set of Boolean variables X, the *Hamming distance* of two assignment τ_1 and τ_2 to X (in symbols, $d_H(\tau_1, \tau_2)$) is defined as the number of positions on which τ_1 and τ_2 disagree.

According to the above definition, we can define the Hamming distance of a collection of constraint applications as the maximum Hamming distance between any two satisfying assignments. In the case of the previous example, we have four satisfying assignments to $\{x_a, x_b, x_c, x_{ab}, x_{ac}, x_{bc}\}$, that is, $\tau_1 = (0,1,1,0,1,0)$, $\tau_2 = (0,1,0,0,1,0)$, $\tau_3 = (0,0,1,0,1,0)$, and $\tau_4 = (0,0,1,1,0,1)$. The following table shows the hamming distance between these assignments:

	τ_2	τ_3	τ_4
τ_1	1	1	4
τ_2		2	5
τ_3			3

Observe that, in this case, the maximum Hamming distance is reached by two assignments (that is, τ_2 and τ_4) that correspond to two blocks worlds (that is, world 2 and 4) that have maximum move action distance. This is not true in general: indeed, it is easy to find examples in which the maximum Hamming distance is reached by two assignments that correspond to two blocks worlds whose move action distance is not maximum.

We are now ready to define the optimization problems that are the subject of our study.

Definition 6 (MAX HAMMINGDISTANCE($\mathcal{F}$)). Let C be a collection of m constraint applications of the form $\{\langle f_j, \langle i_1(j), \ldots, i_{k_j}(j)\rangle\rangle\colon\ j = 1, \ldots, m\}$, on Boolean variables $X = \{x_1, \ldots, x_n\}$ where $f_j \in \mathcal{F}$ and k_j is the arity of f_j. The MAX HAMMINGDISTANCE($\mathcal{F}$) problem with input C consists in finding two assignments that satisfy all the constraint applications in C and have maximum Hamming distance.

5. The main result

The rest of the paper is devoted to give a complete classification of the approximability properties of MAX HAMMINGDISTANCE($\mathscr{F}$). To this aim, we first need to add two new classification criteria to those introduced in Sections 2 and 3.

Definition 7 (01-*valid constraint*). A constraint f is 01-*valid* if it is both 0-valid and 1-valid.

Definition 8 (*Strongly* 1-*valid constraint*). A constraint f is *strongly* 1-*valid* if it is satisfied by all assignments with at most one 0.

Definition 9 (*Even-affine constraint*). A constraint f is *even-affine* if it can be expressed as a conjunction of linear equalities in $\mathscr{Z}_2$ with an even number of terms.

A constraint family $\mathscr{F}$ is 01-valid (respectively, strongly 1-valid and even-affine) if all its constraints are 01-valid (respectively, strongly 1-valid and even-affine).

We are now ready to state our main theorem (see also Fig. 2).

Theorem 10 (MAX HAMMINGDISTANCE($\mathscr{F}$) classification). *Given a constraint family* $\mathscr{F}$, MAX HAMMINGDISTANCE($\mathscr{F}$) *is either in* PO *or* APX*-complete or* POLYAPX *-complete or in* NPO *but not approximable to within any factor. In particular, the following hold*:

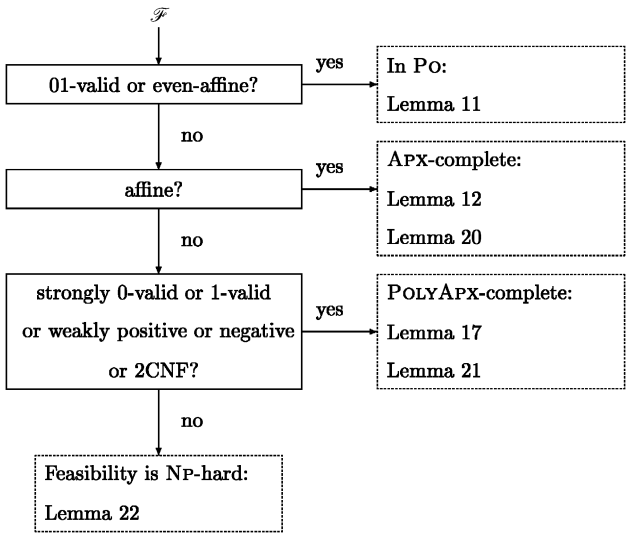

Fig. 2. The complexity of the maximum Hamming distance problem.

(i) *If* $\mathscr{F}$ *is* 01*-valid or even-affine, then* Max HammingDistance($\mathscr{F}$) *is in* Po.
(ii) *Else if* $\mathscr{F}$ *is affine, then* Max HammingDistance($\mathscr{F}$) *is* Apx*-complete.*
(iii) *Else if* $\mathscr{F}$ *is weakly positive or weakly negative or 2CNF or strongly* 0*-valid or strongly* 1*-valid, then* Max HammingDistance($\mathscr{F}$) *is* PolyApx*-complete.*
(iv) *Else finding a feasible solution of* Max HammingDistance($\mathscr{F}$) *is* Np*-hard.*

The proof of the above theorem is split into two parts. In the next section, we give the algorithms proving the upper bounds (e.g., we prove that Max HammingDistance($\mathscr{F}$) is in Apx whenever $\mathscr{F}$ is affine), while in Section 5.2 we provide the reductions showing the lower bounds (e.g., we prove that Max HammingDistance($\mathscr{F}$) is not in Ptas whenever $\mathscr{F}$ is affine).

5.1. Positive results

The first result of this section shows that 01-validity or even-affinity is sufficient to optimally solve the maximum Hamming distance problem in polynomial time.

Lemma 11 (Optimal algorithms). *Given a constraint family* $\mathscr{F}$*, if* $\mathscr{F}$ *is* 01*-valid or even-affine, then* Max HammingDistance($\mathscr{F}$) *is in* Po.

Proof. If $\mathscr{F}$ is 01-valid, then the solution formed by the two satisfying assignments $(0,\ldots,0)$ and $(1,\ldots,1)$ is, clearly, an optimal solution.

If $\mathscr{F}$ is even-affine, the, for any collection C of m constraint applications from $\mathscr{F}$ on n Boolean variables, we can find an assignment $\tau \in \{0,1\}^n$ that satisfies C in polynomial time by simply resolving the linear system corresponding to C. Observe now that, since $\mathscr{F}$ is even-affine, the assignment τ^c obtained from τ by complementing all its components, is still a satisfying assignment (indeed, this is due to the fact that, for any $x, y \in \mathscr{Z}_2$, $x+y \equiv (1-x)+(1-y) \bmod 2$). Hence, the solution formed by the two satisfying assignments τ and τ^c is an optimal solution for C. □

The next result shows that affinity is sufficient to solve the maximum Hamming distance problem in polynomial time within a constant factor of the optimal measure.

Lemma 12 (Approximation algorithm). *Given a constraint family* $\mathscr{F}$*, if* $\mathscr{F}$ *is affine, then* Max HammingDistance($\mathscr{F}$) *is in* Apx.

Proof. Let C be a collection of constraint applications from $\mathscr{F}$ on n Boolean variables. We will now compute two assignments $\tau', \tau'' \in \mathscr{Z}_2^n$ that satisfy the linear system $Ax = b$ corresponding to C, where $A \in \mathscr{Z}_2^{m,n}$, $x \in \mathscr{Z}_2^n$, and $b \in \mathscr{Z}_2^m$, and we will show that $d_H(\tau', \tau'')$ is at least half of the maximum Hamming distance between any two assignments satisfying C.

Assume without loss of generality, that $n \geqslant m$ and that the rows of A are independent. It is then easy to find in polynomial time a matrix $A' \in \mathscr{Z}_2^{m,n-m}$, a vector $b' \in \mathscr{Z}_2^m$,

and an ordering $x_1,\ldots,x_n$ of the n Boolean variables such that $x'' = A'x' + b'$, where $x'' = (x_1,\ldots,x_m)$ and $x' = (x_{m+1},\ldots,x_n)$.

The construction of τ' and τ'' is then obtained by the following randomized algorithm. For each variable x_i in x' with $m+1 \leqslant i \leqslant n$, we define $\tau'_i = 0$ with probability 1 and $\tau''_i = 1$ with probability 1/2. This, clearly, implies that, for each variable x_i in x'' with $1 \leqslant i \leqslant m$, $\tau'_i = b'_i$ with probability 1 and $\tau''_i = b'_i$ with probability 1/2. In summary, the expected number of variables on which τ' and τ'' differ is one half of the total number n of variables. Since n is an upper bound on the measure of an optimal solution, the expected performance ratio of the above algorithm is at most 2. The algorithm can be derandomized by means of the method of conditional probabilities (see, for example, [1, 11]). □

It, finally, remains to prove that if a constraint family $\mathscr{F}$ is strongly 0-valid or strongly 1-valid or weakly positive or weakly negative or 2CNF then MAX HAMMINGDISTANCE($\mathscr{F}$) is in POLYAPX. To this aim, let us first introduce the following property of a constraint family.

Definition 13 (*Strongly decidability*). A constraint family $\mathscr{F}$ is strongly decidable if, given a collection C of constraint applications from $\mathscr{F}$ on n Boolean variables, an index $i \in \{1,\ldots,n\}$, and a value $b \in \{0,1\}$, it is possible to compute in polynomial time an assignment $\tau \in \{0,1\}^n$ satisfying C and such that $\tau_i = b$ (if one such assignment exists).

The following result states that all the cases that remain to be dealt with satisfy the strongly decidability property.

Lemma 14 (Khanna et al. [9]). *Given a constraint family $\mathscr{F}$, if $\mathscr{F}$ is strongly* 0*-valid or strongly* 1*-valid or weakly positive or weakly negative or 2CNF, then $\mathscr{F}$ is strongly decidable.*

An immediate consequence of being strongly decidable is the fact that the corresponding satisfiability problem is solvable in polynomial time, as stated by the following lemma.

Lemma 15. *Given a constraint family $\mathscr{F}$, if $\mathscr{F}$ is strongly decidable, then* SATISFIABILITY($\mathscr{F}$) *is in* PO.

Proof. Let C be a collection of constraint applications from $\mathscr{F}$ on n Boolean variables. Since $\mathscr{F}$ is strongly decidable, we compute in polynomial time an assignment τ satisfying C and such that $\tau_1 = 0$, if one such assignment exists. Otherwise, we compute in polynomial time an assignment τ satisfying C and such that $\tau_1 = 1$, if one such assignment exists. Otherwise, C is not satisfiable. □

We are now ready to show that the above stronger form of decidability leads to a polynomial-time poly-approximation algorithm for the maximum hamming distance problem.

Lemma 16 (Poly-approximation algorithm). *Given a constraint family $\mathscr{F}$, if $\mathscr{F}$ is strongly decidable, then* Max HammingDistance($\mathscr{F}$) *is in* PolyApx.

Proof. Let C be a collection of constraint applications from $\mathscr{F}$ on n Boolean variables $\{x_1,\ldots,x_n\}$. From Lemma 15, it follows that we can compute in polynomial time an assignment τ' satisfying C, if one such assignment exists (otherwise, C is not satisfiable). We can then compute another assignment τ'' as follows. For any variable x_i with $1\leqslant i\leqslant n$, check whether there exists an assignment $\tau''[i]\in\{0,1\}^n$ satisfying C and such that $\tau''[i]_i\neq\tau'_i$ (this can be done in polynomial time because $\mathscr{F}$ is strongly decidable). If $\tau''[i]$ does not exists for any index i, then τ' is the only satisfying assignment and, hence, we have optimally solved the problem. Otherwise, let $\tau''=\tau''[j]$ where j is the first index for which $\tau''[j]$ exists. Clearly, $d_H(\tau',\tau'')\geqslant 1$. Since the optimal measure is at most n, we thus have that the performance ratio of the solution formed by τ' and τ'' is at most n. In conclusion, the above algorithm proves that Max HammingDistance($\mathscr{F}$) is in PolyApx. □

From Lemmas 14 and 16, it follows the last result of this section.

Corollary 17. *Given a constraint family $\mathscr{F}$, if $\mathscr{F}$ is strongly* 0*-valid or strongly* 1*-valid or weakly positive or weakly negative or 2CNF, then* Max HammingDistance($\mathscr{F}$) *is in* PolyApx.

5.2. Negative results

In order to complete the proof of our main theorem, we now prove in this section three hardness results. To this aim, we make use of both Theorem 4 and of an immediate corollary of that theorem. This corollary deals with the following CSOP.

Definition 18 (Max Zeros($\mathscr{F}$)). Let C be a collection of m constraint applications of the form $\{\langle f_j,\langle i_1(j),\ldots,i_{k_j}(j)\rangle\rangle\colon\ j=1,\ldots,m\}$, on Boolean variables $X=\{x_1,\ldots,x_n\}$ where $f_j\in\mathscr{F}$ and k_j is the arity of f_j. The Max Zeros($\mathscr{F}$) problem with input C consists in finding an assignment τ that satisfy all the constraint applications in C and such that $\mathtt{zeros}(\tau)$ is maximum, where $\mathtt{zeros}(\tau)=n-\mathtt{ones}(\tau)$.

Corollary 19 (Max Zeros($\mathscr{F}$) classification). *Given a constraint family $\mathscr{F}$,* Max Zeros($\mathscr{F}$) *is either in* Po *or* Apx*-complete or* PolyApx*-complete or in* Npo *but not approximable to within any factor. In particular, the following hold*:

(i) *If $\mathscr{F}$ is* 0*-valid or weakly negative or width-*2 *affine, then* Max Zeros($\mathscr{F}$) *is in* Po.

(ii) *Else if $\mathscr{F}$ is affine, then* Max Zeros($\mathscr{F}$) *is* Apx*-complete.*

(iii) *Else if $\mathscr{F}$ is strongly* 1*-valid or weakly positive or 2CNF, then* MAX ZEROS($\mathscr{F}$) *is* POLYAPX*-complete.*
(iv) *Else if $\mathscr{F}$* 1*-valid, then finding a solution of* MAX ZEROS($\mathscr{F}$) *with positive measure is* NP*-hard.*
(v) *Else finding a feasible solution of* MAX ZEROS($\mathscr{F}$) *is* NP*-hard.*

The first hardness result on the maximum Hamming distance problem states that 01-validity and even-affinity are necessary conditions in order to make the affine problem solvable in polynomial time (in the following, $\langle \tau_1, \tau_2 \rangle$ will denote the solution formed by the two satisfying assignments τ_1 and τ_2).

Lemma 20 (APX-completeness). *Let $\mathscr{F}$ be a constraint family. If $\mathscr{F}$ is affine but not* 01*-valid and not even-affine, then* MAX HAMMINGDISTANCE($\mathscr{F}$) *is* APX*-complete.*

Proof. Let $\mathscr{F}$ be the family consisting of the ternary constraint that can be written as $x+y+z \equiv 0 \bmod 2$ (that is, the Boolean ternary function that assume the value 1 only if an even number of its arguments is equal to 1). We now L-reduce MAX ONES($\mathscr{F}$) to MAX HAMMINGDISTANCE($\mathscr{F}$). Since MAX ONES($\mathscr{F}$) is APX-complete [9], the lemma will follow.

Given an instance of MAX ONES($\mathscr{F}$), that is, a collection $C = \{c_1, \ldots, c_n\}$ of constraint applications from $\mathscr{F}$ on n Boolean variables, we define $f(C) = C$. Observe that, given a solution of C (that is, an assignment τ satisfying C), the pair $\langle (0, \ldots, 0), \tau \rangle$ is a solution of $f(C)$ such that $\mathtt{ones}(\tau) = d_H((0, \ldots, 0), \tau)$. Hence, $\mathtt{opt}(C) \leqslant \mathtt{opt}(f(C))$.

Moreover, given a solution $\langle \tau', \tau'' \rangle$ of $f(C)$, we define a solution $\tau = g(C, \langle \tau', \tau'' \rangle)$ of C as

$$\tau_i = \begin{cases} 1 & \text{if } \tau'_i \neq \tau''_i, \\ 0 & \text{otherwise.} \end{cases}$$

The assignment τ satisfies C. On the contrary, assume that there exists a constraint application c_j such that τ assigns the value 1 to an odd number of variables in c_j. From the definition of τ, it follows that either τ' or τ'' assigns the value 1 to an odd number of variables in c_j which contradicts the fact that both τ' and τ'' satisfy c_j.

Clearly, $\mathtt{ones}(\tau) = d_H(\tau', \tau'')$. Hence,

$$\mathtt{opt}(f(C)) = \mathtt{opt}(C)$$

and

$$\mathtt{opt}(C) - \mathtt{m}(C, \tau) = \mathtt{opt}(f(C)) - \mathtt{m}(f(C), \langle \tau', \tau'' \rangle).$$

In conclusion, $(f, g, 1, 1)$ is an L-reduction from MAX ONES($\mathscr{F}$) to MAX HAMMING DISTANCE($\mathscr{F}$). □

The second hardness result states that all the poly-approximable cases shown in the previous section cannot be better approximated.

Lemma 21 (PolyApx-completeness). *Given a constraint family $\mathscr{F}$, if $\mathscr{F}$ is strongly* 0*-valid or strongly* 1*-valid or weakly positive or weakly negative or 2CNF but not* 01*-valid and not affine, then* Max HammingDistance($\mathscr{F}$) *is* PolyApx*-complete.*

Proof. Let $\mathscr{F}$ be a strongly 0-valid constraint family. We now A-reduce Max Ones($\mathscr{F}$) to Max HammingDistance($\mathscr{F}$). For any instance C of Max Ones($\mathscr{F}$), let $f(C)=C$ be the corresponding instance of Max HammingDistance($\mathscr{F}$). Given a solution $\langle\tau',\tau''\rangle$ of $f(C)$, let $\tau=g(C,\langle\tau',\tau''\rangle)$ be the assignment with the greatest number of ones between τ' and τ''. Clearly, $\mathtt{ones}(\tau)\geqslant d_H(\tau',\tau'')/2$. Moreover, given an assignment τ satisfying C, the pair $\langle(0,\ldots,0),\tau\rangle$ is a solution of $f(C)$ whose measure is equal to $\mathtt{ones}(\tau)$. It thus follows that $\mathtt{opt}(C)\leqslant\mathtt{opt}(f(C))$. Hence, $(f,g,2)$ is an A-reduction from Max Ones($\mathscr{F}$) to Max HammingDistance($\mathscr{F}$). Since Max Ones($\mathscr{F}$) is PolyApx-complete [9], this part of the lemma follows.

The above reduction can also be used to prove that if $\mathscr{F}$ is weakly negative, then Max HammingDistance($\mathscr{F}$) is PolyApx-complete. To this aim, it suffices to observe that, without loss of generality, we may assume that the arity of every constraint in $\mathscr{F}$ is at least 2 (indeed, any unary constraint application forces the value of its variable to be the same in any satisfying assignment). Since any weakly negative constraint whose arity is at least 2 is 0-valid, the above reduction is an A-reduction from Max Ones($\mathscr{F}$) to Max HammingDistance($\mathscr{F}$), where $\mathscr{F}$ is weakly negative.

A similar argument (starting from Max Zeros($\mathscr{F}$)) can be used to prove that if $\mathscr{F}$ is strongly 1-valid or weakly positive, then Max HammingDistance($\mathscr{F}$) is PolyApx-complete.

Finally, let $\mathscr{F}$ be 2CNF. We now A-reduce Max Ones($\mathscr{F}$) to Max Hamming Distance($\mathscr{F}$). Given an instance of Max Ones($\mathscr{F}$), that is, a collection C of constraint applications from $\mathscr{F}$ on n Boolean variables $\{x_1,\ldots,x_n\}$, the corresponding instance $f(C)$ of Max HammingDistance($\mathscr{F}$) is defined as $f(C)=C\cup\{x_1\vee x_{n+1},\ldots,x_n\vee x_{2n}\}$, where $\{x_{n+1},\ldots,x_{2n}\}$ are n new Boolean variables. Given an assignment τ satisfying C, let τ' and τ'' be two assignments for $f(C)$ defined as

$$\tau'_i=\begin{cases}\tau_i & \text{if } i\leqslant n,\\ 1 & \text{otherwise.}\end{cases}$$

and as

$$\tau''_i=\begin{cases}\tau_i & \text{if } i\leqslant n,\\ 0 & \text{if } n+1\leqslant i\leqslant 2n \quad\text{and}\quad \tau_{i-n}=1,\\ 1 & \text{otherwise.}\end{cases}$$

Clearly, $\langle\tau',\tau''\rangle$ is a solution of $f(C)$ and $\mathtt{ones}(\tau)=d_H(\tau',\tau'')$. Hence,

$$\mathtt{opt}(C)\leqslant\mathtt{opt}(f(C)).$$

Moreover, given a solution $\langle\tau',\tau''\rangle$ of $f(C)$, let $d^l_H(\tau',\tau'')$ (respectively, $d^r_H(\tau',\tau'')$) the Hamming distance between the leftmost (respectively, rightmost) n positions of τ' and τ''. Observe that we may always restrict ourselves to solutions such that $d^l_H(\tau',\tau'')\leqslant$

$d_H^r(\tau',\tau'')$. Indeed, if this is not the case, then there exists i with $1 \leqslant i \leqslant n$ such that $\tau'_i \neq \tau''_i$ and $\tau'_{n+i} = \tau''_{n+i}$. Assume that $\tau'_i = 0$ and $\tau''_i = 1$ (the other case is symmetric). This implies that $\tau'_{n+i} = \tau''_{n+i} = 1$. If we set $\tau''_{n+i} = 0$, then we obtain a new solution with a greater Hamming distance.

Since $\tau'_{n+i} \neq \tau''_{n+i}$ implies $(\tau'_i = 1 \vee \tau''_i = 1)$, we have that

$$|\{i\colon\ 1 \leqslant i \leqslant n \wedge (\tau'_i = 1 \vee \tau''_i = 1)\}| \geqslant d_H^r(\tau',\tau'').$$

Assume that the number of ones in the first n positions of τ' is greater than or equal to the number of ones in the first n positions of τ'' (we can prove the other case in a similar way). Let us then define $\tau = g(C, \langle \tau', \tau'' \rangle)$ as the first n positions of τ'. Since

$$\begin{aligned} \mathtt{ones}(\tau) &= |\{i\colon\ 1 \leqslant i \leqslant n \wedge \tau'_i = 1\}| \\ &\geqslant \tfrac{1}{2}|\{i\colon\ 1 \leqslant i \leqslant n \wedge (\tau'_i = 1 \vee \tau''_i = 1)\}|, \end{aligned}$$

we have that

$$\mathtt{ones}(\tau) \geqslant \frac{d_H^r(\tau',\tau'')}{2}.$$

Finally, from the fact that

$$d_H(\tau',\tau'') = d_H^l(\tau',\tau'') + d_H^r(\tau',\tau'') \leqslant 2d_H^r(\tau',\tau'')$$

it follows that

$$\mathtt{ones}(\tau) \geqslant \frac{d_H(\tau',\tau'')}{4}.$$

We have thus described an A-reduction from Max Ones($\mathscr{F}$) to Max Hamming Distance($\mathscr{F}$) with $\alpha = 4$. Since Max Ones($\mathscr{F}$) is PolyApx-complete if $\mathscr{F}$ is 2CNF [9], the last part of the lemma follows. □

Lemma 22 (The remaining cases). *If $\mathscr{F}$ is not* 01*-valid, not affine, not strongly* 0*-valid, not strongly* 1*-valid, not weakly positive, not weakly negative, and not 2CNF, then it is* Np*-hard to find a solution for* Max HammingDistance($\mathscr{F}$).

Proof. From Theorem 4 we have that if $\mathscr{F}$ is not 1-valid, not 0-valid, not strongly 0-valid, not 2CNF, not affine, not weakly positive, and not weakly negative then finding a feasible solution for Max Ones($\mathscr{F}$) (and, hence, for Max HammingDistance ($\mathscr{F}$)) is Np-hard. Observe that if $\mathscr{F}$ is not 0-valid and 1-valid, then $\mathscr{F}$ is neither 01-valid nor strongly 1-valid. Hence, the theorem follows. □

References

[1] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, M. Protasi, Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties, Springer, Berlin, 1999.

[2] J.L. Balcazar, J. Diaz, J. Gabarro, Structural Complexity I, Springer, Berlin, 1988.
[3] D.P. Bovet, P. Crescenzi, Introduction to the theory of complexity, Prentice-Hall, Englewood Cliffs, NJ, 1993.
[4] S.A. Cook, The complexity of theorem-proving procedures, Proc. 3rd ACM Symp. on Theory of Computing, 1971, pp. 151–158.
[5] P. Crescenzi, A. Panconesi, Completeness in approximation classes, Informat. and Comput. 93 (1991) 241–262.
[6] M.R. Genesereth, N.J. Nilsson, Logical Foundations of Artificial Intelligence, Morgan Kaufmann, Los Altos, CA, 1987.
[7] S. Khanna, M. Sudan, L. Trevisan, Constraint satisfaction: the approximability of minimization problems. Proc. 12th IEEE Conf. on Computational Complexity, 1997, pp. 282–296.
[8] S. Khanna, M. Sudan, L. Trevisan, D.P. Williamson. The approximability of constraint satisfaction problems, SIAM J. Comput. 30 (2001) 1863–1920.
[9] S. Khanna, M. Sudan, D.P. Williamson, A complete classification of the approximability of maximization problems derived from Boolean constraint satisfaction, Proc. 29th Annual ACM Symp. on Theory of Computing, 1997, pp. 11–20.
[10] V. Kumar, Algorithms for constraint-satisfaction problems: a survey, AI Mag. 13 (4) (1992) 32–44.
[11] R. Motwani, P. Raghavan, Randomized Algorithms, Cambridge University Press, Cambridge, MA, 1995.
[12] C.H. Papadimitriou, Computational complexity, Addison-Wesley, Reading, MA, 1993.
[13] C.H. Papadimitriou, M. Yannakakis, Optimization, approximation, and complexity classes, J. Comput. System Sci. 43 (1991) 425–440.
[14] T. Schaefer, The complexity of satisfiability problems, Proc. 10th Annu. ACM Symp. on Theory of Computing, 1978, pp. 216–226.