On the hardness of losing weight[⋆]

Andrei Krokhin[1] and Dániel Marx[2]

[1] Department of Computer Science, Durham University, Durham, DH1 3LE, UK
`andrei.krokhin@durham.ac.uk`
[2] Department of Computer Science and Information Theory, Budapest University of Technology and Economics, H-1521 Budapest, Hungary
`dmarx@cs.bme.hu`

Abstract. We study the complexity of local search for the Boolean constraint satisfaction problem (CSP), in the following form: given a CSP instance, that is, a collection of constraints, and a solution to it, the question is whether there is a better (lighter, i.e., having strictly less Hamming weight) solution within a given distance from the initial solution. We classify the complexity, both classical and parameterized, of such problems by a Schaefer-style dichotomy result, that is, with a restricted set of allowed types of constraints. Our results show that there is a considerable amount of such problems that are NP-hard, but fixed-parameter tractable when parameterized by the distance.

1 Introduction

Local search is one of the most widely used approaches to solving hard optimization problems. The basic idea of local search is that one tries to iteratively improve a current solution by searching for better solutions in its (k-)neighborhood (i.e., within distance k from it). Any optimization algorithm can be followed by a local search phase, thus the problem of finding a better solution locally is of practical interest. As a brute force search of a k-neighborhood is not feasible for large k, thus it is natural to study the complexity of searching the k-neighborhood.

The constraint satisfaction problem (CSP) provides a framework in which it is possible to express, in a natural way, many combinatorial problems encountered in artificial intelligence and computer science. A CSP instance is represented by a set of variables, a domain of values for each variable, and a set of constraints on the variables. The basic aim is then to find an assignment of values to the variables that satisfies the constraints. Boolean CSP (when all variables have domain $\{0,1\}$) is a natural generalization of Sat where constraints are given by arbitrary relations, not necessarily by clauses. Local search methods for Sat and CSP are very extensively studied (see, e.g., [5,9,10,11]).

Complexity classifications for various versions of (Boolean) CSP have recently attracted massive attention from researchers, and one of the most popular directions here is to characterise restrictions on the type of constraints that lead to

[⋆] The first author is supported by UK EPSRC grants EP/C543831/1 and EP/C54384X/1; the second author is supported by the Magyary Zoltán Felsőoktatási Közalapítvány and the Hungarian National Research Fund (OTKA grant 67651).

problems with lower complexity in comparison with the general case (see [2,3]). Such classifications are sometimes called Schaefer-style because the first classification of this type was obtained by T.J. Schaefer in his seminal work [15]. A local-search related Schaefer-style classification for Boolean MAX CSP was obtained in [1], in the context of local search complexity classes such as PLS.

The hardness of searching the k-neighborhood (for any optimisation problem) can be studied very naturally in the framework of parameterized complexity [6,8], as suggested in [7]; such a study for the traveling salesman problem (TSP) was recently performed in [14]. Parameterized complexity studies hardness in finer detail than classical complexity. Consider, for example, two standard NP-complete problems MINIMUM VERTEX COVER and MAXIMUM CLIQUE. Both have the natural parameter k: the size of the required vertex cover/clique. Both problems can be solved in $n^{O(k)}$ time by complete enumeration. Notice that the degree of the polynomial grows with k, so the algorithm becomes useless for large graphs, even if k is as small as 10. However, MINIMUM VERTEX COVER can be solved in time $O(2^k \cdot n^2)$ [6,8]. In other words, for every fixed cover size there is a polynomial-time (in this case, quadratic) algorithm solving the problem where the degree of the polynomial is independent of the parameter. Problems with this property are called fixed-parameter tractable. The notion of W[1]-hardness in parameterized complexity is analogous to NP-completeness in classical complexity. Problems that are shown to be W[1]-hard, such as MAXIMUM CLIQUE [6,8], are very unlikely to be fixed-parameter tractable. A Schaefer-style classification of the basic Boolean CSP with respect to parameterized complexity (where the parameter is the required Hamming weight of the solution) was obtained in [13].

In this paper, we give a Schaefer-style complexity classification for the following problem: given a collection of Boolean constraints, and a solution to it, the question is whether there is a better (i.e., with smaller Hamming weight) solution within a given (Hamming) distance k from the initial solution. We obtain classification results both for classical (Theorem 9) and for parameterized complexity (Theorem 3). However, we would like to point out that it makes much more sense to study this problem in the parameterized setting. Intuitively, if we are able to decide in polynomial time whether there is a better solution within distance k, then this seems to be almost as powerful as finding the best solution (although there are technicalities such as whether there is a feasible solution at all). Our classification confirms this intuition: searching the k-neighborhood is polynomial-time solvable only in cases where finding the optimum is also polynomial-time solvable. On the other hand, there are cases (for example, 1-IN-3 SAT or affine constraints of fixed arity) where the problem of finding the optimum is NP-hard, but searching the k-neighborhood is fixed-parameter tractable. This suggests evidence that parameterized complexity is the right setting for studying local search.

The paper is organized as follows. Section 2 reviews basic notions of parameterized complexity and Boolean CSP. Section 3 presents the classificiation with respect to fixed-parameter tractability, while Section 4 deals with polynomial-time solvability. The proofs omitted from Section 4 will appear in the full version.

2 Preliminaries

Boolean CSP. A *formula* ϕ is a pair (V, C) consisting of a set V of *variables* and a set C of *constraints.* Each constraint $c_i \in C$ is a pair $\langle \overline{s}_i, R_i \rangle$, where $\overline{s}_i = (x_{i,1}, \ldots, x_{i,r_i})$ is an r_i-tuple of variables (the *constraint scope*) and $R_i \subseteq \{0,1\}^{r_i}$ is an r_i-ary Boolean relation (the *constraint relation*). A function $f : V \to \{0,1\}$ is a *satisfying assignment* of ϕ if $(f(x_{i,1}), \ldots, f(x_{i,r_i}))$ is in R_i for every $c_i \in C$. Let Γ be a set of Boolean relations. A formula is a *Γ-formula* if every constraint relation R_i is in Γ. In this paper, Γ is always a finite set. The *(Hamming) weight* $w(f)$ of assignment f is the number of variables x with $f(x) = 1$. The *distance* $\mathrm{dist}(f_1, f_2)$ of assignments f_1, f_2 is the number of variables x with $f_1(x) \neq f_2(x)$.

We recall various standard definitions concerning Boolean constraints (cf. [3]):

– R is *0-valid* if $(0, \ldots, 0) \in R$.
– R is *1-valid* if $(1, \ldots, 1) \in R$.
– R is *Horn* or *weakly negative* if it can be expressed as a conjunction of clauses such that each clause contains at most one positive literal. It is known that R is Horn if and only if it is *min-closed:* if $(a_1, \ldots, a_r) \in R$ and $(b_1, \ldots, b_r) \in R$, then $(\min(a_1, b_1), \ldots, \min(a_r, b_r)) \in R$.
– R is *affine* if it can be expressed as a conjunction of constraints of the form $x_1 + x_2 + \cdots + x_t = b$, where $b \in \{0,1\}$ and addition is modulo 2. The number of tuples in an affine relation is always an integer power of 2.
– R is *width-2 affine* if it can be expressed as a conjunction of constraints of the form $x = y$ and $x \neq y$.
– R is *IHS-B−* (or *implicative hitting set bounded*) if it can be represented by a conjunction of clauses of the form (x), $(x \to y)$ and $(\neg x_1 \vee \ldots \neg x_n)$, $n \geq 1$.
– The relation $R_{p\text{-IN-}q}$ (for $1 \leq p \leq q$) has arity q and $R_{p\text{-IN-}q}(x_1, \ldots, x_q)$ is true if and only if exactly p of the variables $x_1, \ldots, x_q$ have value 1.

The following definition is new in this paper. It plays a crucial role in characterizing the fixed-parameter tractable cases for local search.

Definition 1. *Let R be a Boolean relation and $(a_1, \ldots, a_r) \in R$. A set $S \subseteq \{1, \ldots, r\}$ is a* flip set *of $(a_1, \ldots, a_r)$ (with respect to R) if $(b_1, \ldots, b_r) \in R$ where $b_i = 1 - a_i$ for $i \in S$ and $b_i = a_i$ for $i \notin S$. We say that R is* flip separable *if whenever some $(a_1, \ldots, a_r) \in R$ has two flip sets S_1, S_2 with $S_1 \subset S_2$, then $S_2 \setminus S_1$ is also a flip set for $(a_1, \ldots, a_r)$.*

It is easy to see that $R_{1\text{-IN-}3}$ is flip separable: every flip set has size exactly 2, hence $S_1 \subset S_2$ is not possible. Moreover, $R_{p\text{-IN-}q}$ is also flip separable for every $p \leq q$. Affine constraints are also flip separable: to see this, it is sufficient to verify the definition only for the constraint $x_1 + \cdots + x_r = 0$.

The basic problem in CSP is to decide if a formula has a satisfying assignment:

CSP(Γ)

Input: A Γ-formula ϕ.

Question: Does ϕ have a satisfying assignment?

Schaefer completely characterized the complexity of CSP(Γ) for every finite set Γ of Boolean relations [15]. In particular, every such problem is either in PTIME or NP-complete, and there is a very clear description of the boundary between the two cases.

Optimization versions of Boolean CSP were investigated in [3,4]. A straightforward way to obtain an optimization problem is to relax the requirement that every constraint is satisfied, and ask for an assignment maximizing the number of satisfied constraints. Another possibility is to ask for a solution with minimum/maximum weight. In this paper, we investigate the problem of minimizing the weight. As we do not consider the approximability of the problem, we define here only the decision version:

Min-Ones(Γ)	
Input:	A Γ-formula ϕ and an integer W.
Question:	Does ϕ have a satisfying assignment f with $w(f) \leq W$?

The characterization of the approximability of finding a minimum weight satisfying assignment for a Γ-formula can be found in [3]. Here we state only the classification of polynomial-time solvable and NP-hard cases:

Theorem 2 ([3]). *Let Γ be a finite set of Boolean relations.* Min-Ones(Γ) *is solvable in polynomial time if one the following holds, and* NP*-complete otherwise:*

- *Every $R \in \Gamma$ is 0-valid.*
- *Every $R \in \Gamma$ is Horn.*
- *Every $R \in \Gamma$ is width-2 affine.*

A Schaefer-style characterization of the approximability of finding two satisfying assignments to a formula with a largest distance between them was obtained in [4], motivated by the blocks world problem from KR, while a Schaefer-style classification of the problem of deciding whether a given satisfying assignment to a given CSP instance is component-wise minimal was presented in [12], motivated by the circumscription formalism from AI.

The main focus of the paper is the local search version of minimizing weight:

LS-CSP(Γ)	
Input:	A Γ-formula ϕ, a satisfying assignment f, and an integer k.
Question:	Does ϕ have a satisfying assignment f' with $w(f') < w(f)$ and $\text{dist}(f, f') \leq k$?

LS in the above problem stands for both "local search" and "lighter solution."

Observe that the satisfying assignments of an $(x \vee y)$-formula correspond to the vertex covers of the graph where the variables are the vertices and the edges are the constraints. Thus LS-CSP($\{x \vee y\}$) is the problem of reducing the size of a (given) vertex cover by including and excluding a total of at most k vertices.

As we shall see (Lemma 7), this problem is W[1]-hard, even for bipartite graphs. Since the complement of an independent set is a vertex cover and vice versa, a similar W[1]-hardness result follows for increasing an independent set.

Parameterized complexity. In a *parmeterized problem,* each instance contains an integer k called the *parameter.* A parameterized problem is *fixed-parameter tractable (FPT)* if it can be solved by an algorithm with running time $f(k) \cdot n^c$, where n is the length of the input, f is an arbitrary (computable) function depending only on k, and c is a constant independent of k.

A large fraction of NP-complete problems is known to be FPT. On the other hand, analogously to NP-completeness in classical complexity, the theory of W[1]-hardness can be used to give strong evidence that certain problems are unlikely to be fixed-parameter tractable. We omit the somewhat technical definition of the complexity class W[1], see [6,8] for details. Here it will be sufficient to know that there are many problems, including Maximum Clique, that were proved to be W[1]-hard. To prove that a parameterized problem is W[1]-hard, we have to present a parameterized reduction from a known W[1]-hard problem. A *parameterized reduction* from problem L_1 to problem L_2 is a function that transforms a problem instance x of L_1 with parameter k into a problem instance x' of L_2 with parameter k' in such a way that

- x' is a yes-instance of L_2 if and only if x is a yes-instance of L_1,
- k' can be bounded by a function of k, and
- the transformation can be computed in time $f(k) \cdot |x|^c$ for some constant c and function $f(k)$.

It is easy to see that if there is a parameterized reduction from L_1 to L_2, and L_2 is FPT, then it follows that L_1 is FPT as well.

3 Characterizing fixed-parameter tractability

In this section, we completely characterize those finite sets Γ of Boolean relations for which LS-CSP(Γ) is fixed-parameter tractable.

Theorem 3. *Let Γ be a finite set of Boolean relations. The problem* LS-CSP(Γ) *is in* FPT *if every relation in Γ is Horn or every relation in Γ is flip separable. In all other cases,* LS-CSP(Γ) *is* W[1]*-hard.*

First we handle the fixed-parameter tractable cases (Lemmas 4 and 6)

Lemma 4. *If Γ is finite and every $R \in \Gamma$ is Horn, then* LS-CSP(Γ) *is* FPT.

Proof. If there is a solution f' for the LS-CSP(Γ) instance (ϕ, f, k), then we can assume $f'(x) \leq f(x)$ for every variable x: by defining $f''(x) := \min\{f(x), f'(x)\}$, we get that f'' is also satisfying (as every $R \in \Gamma$ is min-closed) and $\mathrm{dist}(f'', f) \leq \mathrm{dist}(f', f)$. Thus we can restrict our search to solutions that can be obtained from f by changing some 1's to 0's, but every 0 remains unchanged.

Since $w(f') < w(f)$, there is a variable x with $f(x) = 1$ and $f'(x) = 0$. For every variable x with $f(x) = 1$, we try to find a solution f' with $f'(x) = 0$ using a simple bounded-height search tree algorithm. For a particular x, we proceed as follows. We start with initial assignment f. Change the value of x to 0. If there is a constraint $\langle (x_1, \ldots, x_r), R \rangle$ that is not satisfied by the new assignment, then we select one of the variables $x_1, \ldots, x_r$ that has value 1, and change it to 0. Thus at this point we branch into at most $r-1$ directions. If the assignment is still not satisfying, the we branch again on the variables of some unsatisfied constraint. The branching factor of the resulting search tree is at most $r_{\max} - 1$, where $r_{\max}$ is the maximum arity of the relations in Γ. By the observation above, if there is a solution, then we find a solution on the first k levels of the search tree. Therefore, we can stop the search on the k-th level, implying that we visit at most $(r_{\max} - 1)^{k+1}$ nodes of the search tree. The work to be done at each node is polynomial in the size n of the input, hence the total running time is $(r_{\max} - 1)^{k+1} \cdot n^{O(1)}$. □

If every $R \in \Gamma$ is not only Horn, but IHS-B− (which is a subset of Horn), then the algorithm of Lemma 4 actually runs in polynomial time:

Corollary 5. *If every $R \in \Gamma$ is IHS-B−, then* LS-CSP(Γ) *is in* PTIME.

Proof. We can assume that every constraint is either (x), $(x \to y)$, or $(\bar{x}_1 \vee \cdots \vee \bar{x}_r)$. If a constraint $(\bar{x}_1 \vee \cdots \vee \bar{x}_r)$ is satisfied in the initial assignment f, then it remains satisfied after changing some 1's to 0. Observe that if a constraint (x) or $(x \to y)$ is not satisfied, then at most one variable has the value 1. Thus there is no branching involved in the algorithm of Lemma 4, making it a polynomial-time algorithm. □

For flip separable relations, we give a very similar branching algorithm. However, in this case the correctness of the algorithm requires a nontrivial argument.

Lemma 6. *If Γ is finite and every $R \in \Gamma$ is flip separable, then* LS-CSP(Γ) *is FPT.*

Proof. Let (ϕ, f, k) be an instance of LS-CSP(Γ). If $w(f') < w(f)$ for some assignment f', there there is a variable x with $f(x) = 1$ and $f'(x) = 0$. For every variable x with $f(x) = 1$, we try to find a solution f' with $f'(x) = 0$ using a simple bounded-height search tree algorithm. For each such x, we proceed as follows. We start with the initial assignment f and set the value of x to 0. Iteratively do the following: (a) if there is a constraint in ϕ that is not satisfied by the current assignment and such that the value of some variable in it has not been flipped yet (on this branch), then we select one of such variables, and flip its value; (b) if there is no such constraint, but the current assignment is not satisfying then we move to the next branch; (c) if every constraint is satisfied, then either we found a required solution or else we move to the next branch. If a required solution is not found on the first k levels of the search tree then the algorithm reports that there is no required solution.

Assume that (ϕ, f, k) is a yes-instance. We claim that if f' is a required solution with minimal distance from f, then some branch of the algorithm finds it. Let X be the set of variables on which f and f' differ, so $|X| \leq k$. We now show that on the first k levels of the search tree, the algorithm finds some satisfying assignment f_0 (possibly heavier than f) that differs from f only on a subset $X_0 \subseteq X$ of variables. To see this, assume that at some node of the search tree, the current assignment differs from the initial assignment only on a subset of X; we show that this remains true for at least one child of the node. If we branch on the variables $(x_1, \ldots, x_r)$ of an unsatisfied constraint, then at least one of its variables, say x_i, has a value different from f' (as f' is a satisfying assignment). It follows that $x_i \in X$: otherwise the current value of x_i is $f(x_i)$ (since so far we changed variables only in X) and $f(x_i) = f'(x_i)$ (by the definition of X), contradicting the fact that current value of x_i is different from $f(x_i)$. Thus if we change variable x_i, it remains true that only variables from X are changed. Since $|X| \le k$, this branch of the algorithm has to find some satisfying assignment f_0.

If $w(f_0) < w(f)$, then, by the choice of f', we must have $f_0 = f'$. Otherwise, let $X_0 \subseteq X$ be the set of variables where f and f_0 differ and let f'' be the assignment that differs from f exactly on the variables $X \setminus X_0$. From the fact that every constraint is flip separable, it follows that f'' is a satisfying assignment. We claim that $w(f'') < w(f)$. Indeed, if changing the values of the variables in X decreases the weight and changing the values in X_0 does not decrease the weight, then the set $X \setminus X_0$ has to decrease the weight. This contradicts the assumption that f' is a solution whose distance from f is minimal: f'' is a solution with distance $|X \setminus X_0| < |X|$. Thus it is sufficient to investigate only the first k levels of the search tree. As in the proof of Lemma 4, the branching factor of the tree is at most $r_{\max} - 1$, and the algorithm runs in time $(r_{\max} - 1)^{k+1} \cdot n^{O(1)}$. □

All the hardness proofs in this section are based on the following lemma:

Lemma 7. LS-CSP($\{x \vee y\}$) *is* W[1]-*hard.*

Proof. The proof is by reduction from a variant of Maximum Clique: given a graph $G(V, E)$ with a distinguished vertex x and an integer t, we have to decide whether G has a clique of size t that contains x. It is easy to see that this problem is W[1]-hard. Furthermore, it can be assumed that t is odd. Let n be the number of vertices of G and let m be the number of edges. We construct a formula ϕ on $m + n(t-1)/2 - 1$ variables and a satisfying assignment f such that G has a clique of size t containing x if and only if ϕ has a satisfying assignment f' with $w(f') < w(f)$ and distance at most $k := t(t-1) - 1$ from f.

Let $d := (t-1)/2$ (note that t is odd). The formula ϕ has d variables v_1, $\ldots$, v_d for each vertex $v \neq x$ of G and a variable u_e for each edge e of G. The distinguished vertex x has only $d-1$ variables x_1, $\ldots$, x_{d-1}. If a vertex v is the endpoint of an edge e, then for every $1 \le i \le d$ (or $1 \le i \le d-1$, if $v = x$), we add the constraint $u_e \vee v_i$. Thus each variable u_e is in $2d-1$ or $2d$ constraints (depending on whether x is the endpoint of e or not). Set $f(u_e) = 1$ for every $e \in E$ and $f(v_i) = 0$ for every $v \in V$, $1 \le i \le d$. Clearly, f is a satisfying assignment.

Assume that G has a clique K of size t that includes x. Set $f'(v_i) = 1$ for every $v \in K$ $(1 \le i \le d)$ and set $f'(u_e) = 0$ for every edge e in K; let f' be the same as f on every other variable. Observe that f' is also a satisfying assignment:

if a variable u_e was changed to 0 and there is a constraint $u_e \vee v_i$, then $v \in K$ and hence $f'(v_i) = 1$. We have $w(f') < w(f)$: $dt - 1$ variables were changed to 1 (note that $x \in K$) and $t(t-1)/2 = dk$ variables were changed to 0. Moreover, the distance of f and f' is exactly $dt - 1 + t(t-1)/2 = t(t-1) - 1 = k$.

Assume now that f' satisfies the requirements. Let K be the set of those vertices v in G for which $f'(v_i) = 1$ for every i. We claim that K is a clique of size t in G. Observe that there are at least $d|K| - 1$ variables v_i with $f'(v_i) > f(v_i)$ and $f'(u_e) < f(u_e)$ is possible only if both endpoints of e are in K, i.e., e is in the set $E(K)$ of edges in K. Thus $w(f') < w(f)$ implies $d|K| - 1 < |E(K)| \leq |K|(|K| - 1)/2$, which is only possible if $|K| \geq 2d + 1 = t$. If $|K| > t$, then $f'(v_i) > f(v_i)$ for at least $(t+1)d - 1$ variables, hence there must be at least that many variables u_e with $f'(u_e) < f(u_e)$. Thus the distance of f and f' is at least $2(t+1)d - 2 > t(t-1) - 1$. Therefore, we can assume $|K| = t$. Now $dt - 1 < |E(K)| \leq |K|(|K| - 1)/2 = t(t-1)/2$ is only possible if $|E(K)| = t(t-1)/2$ (i.e., K is a clique) and it follows that there are exactly $dt - 1$ variables v_i with $f'(v_i) > f(v_i)$ (i.e., $x \in K$). $\square$

Now we are ready to present the main hardness proof of the section:

Lemma 8. *If Γ contains a relation R_1 that is not Horn and a relation R_2 that is not flip separable, then* LS-CSP(Γ) *is* W[1]*-hard.*

Proof. The proof is by reduction from LS-CSP($\{x \vee y\}$). Let (ϕ_1, f_1, k) be an instance of LS-CSP($\{x \vee y\}$), i.e., every constraint relation in formula $\phi_1 = (V, C)$ is $(x \vee y)$. Since R_1 is not min-closed, we can assume (by permuting the variables) that for some $r_1, r_2 \geq 1$, $r_3, r_4 \geq 0$, if we define

$$R_1'(x, y, w_0, w_1) = R_1(\overbrace{x, \ldots, x}^{r_1}, \overbrace{y, \ldots, y}^{r_2}, \overbrace{w_0, \ldots, w_0}^{r_3}, \overbrace{w_1, \ldots, w_1}^{r_4}),$$

then $(0,1,0,1), (1,0,0,1) \in R_1'$, but $(0,0,0,1) \notin R_1'$. Since R_1' is obtained from R_1 by identifying variables, we can use the relation R_1' when specifying instances of LS-CSP(Γ). We consider two cases:

Case 1: $(1,1,0,1) \in R_1'$. In this case $R_1'(x, y, 0, 1) = x \vee y$, hence it is easy to simulate LS-CSP($\{x \vee y\}$). The only difficulty is how to simulate the constants 0 and 1. We do this as follows. Let us construct a formula ϕ_2 that has every variable of V and new variables q_0^j, q_1^j for every $1 \leq j \leq k+1$ (these new variables will play the role of the constants). We define assignment f_2 of ϕ_2 by setting $f_2(x) = f_1(x)$ for $x \in V$ and $f_2(q_0^j) = 0$ and $f_2(q_1^j) = 1$ for $1 \leq j \leq k+1$. For $1 \leq a, b, c \leq k+1$, we add constraint $c_{a,b,c}^1 = R_1'(q_1^a, q_0^b, q_0^b, q_1^c)$; it is clearly satisfied by assignment f_2. To simulate a constraint $x \vee y$, we add $c_{x,y,j}^2 = R_1'(x, y, q_0^j, q_1^1)$ for every $1 \leq j \leq k+1$.

It is easy to see that if there is a solution f_1' for the original instance (ϕ_1, f_1, k), then by setting $f_2'(x) = f_1'(x)$ for every $x \in V$ and $f_2'(q_0^j) = 0$, $f_2'(q_1^j) = 1$ for every $1 \leq j \leq k+1$ gives a solution f_2' for the constructed instance (ϕ_2, f_2, k). We claim the converse is also true: if f_2' is a solution for the instance (ϕ_2, f_2, k), then the restriction f_1' of f_2' to V gives a solution for (ϕ_1, f_1, k). Since the distance of f_2 and f_2' is at most k, there are $1 \leq b, c \leq k+1$ with $f_2'(q_0^b) = 0$ and

$f_2'(q_1^c) = 1$. Because of the constraint $c^1_{a,b,c}$, we have that $f_2'(q_1^a) = 1$ for every $1 \le a \le k+1$. It follows that f_2' restricted to V is a satisfying assignment of ϕ_1: for every constraint $x \vee y \in C$, the constraint $c^2_{x,y,b}$ prevents the possibility $f_2'(x) = f_2'(y) = 0$. We have seen that $f_2'(q_0^j) \ge f_2(q_0^j)$ and $f_2'(q_1^j) \ge f_2(q_1^j)$ for every $1 \le j \le k+1$. Now $w(f_2') < w(f_2)$ implies that the weight of f_2' on V has to be less than the weight of f_2 on V. Thus $w(f_1') < w(f_1)$.

Case 2: $(1,1,0,1) \notin R_1'$, which means that $R_1'(x,y,0,1)$ is $x \neq y$. In this case we have to rely on the fact that R_2 is not flip separable to simulate the constraint $x \vee y$. We construct formula ϕ_2 and its satisfying assignment f_2 as follows. Each variable x is replaced by 3 variables x_1, x_2, x_3. We set $f_2(x_1) = f_2(x_2) = f_1(x)$ and $f_2(x_3) = 1 - f_1(x)$. Furthermore, for $1 \le j \le 3k+1$, we add the variables q_0^j and q_1^j and set $f_2(q_0^j) = 0$ and $f_2(q_1^j) = 1$.

For every $1 \le a,b,c \le 3k+1$, we add the constraint $c^1_{a,b,c} = R_1'(q_1^a, q_0^b, q_0^b, q_1^c)$, as in the previous case. For every $x \in V$, $1 \le j \le 3k+1$, and $\ell = 1,2$, we add $c^2_{x,\ell,j} = R_1'(x_\ell, x_3, q_0^j, q_1^1)$, as we shall see, the role of these constraints is to ensure $f_2'(x_1) = f_2'(x_2) \neq f_2'(x_3)$.

Since R_2 is not flip separable, there is a tuple $(s_1,\dots,s_r) \in R_2$ that has flip sets $S_1 \subset S_2$, but $S_2 \setminus S_1$ is not a flip set. For every constraint $x \vee y$ of ϕ_1, we add $3k+1$ constraints to ϕ_2 as follows. First, for $1 \le i \le r$ and $1 \le j \le 3k+1$, we define variable v_i^j as

$$v_i^j = \begin{cases} x_1 & \text{if } i \in S_1 \text{ and } s_i = 0, \\ x_3 & \text{if } i \in S_1 \text{ and } s_i = 1, \\ y_1 & \text{if } i \in S_2 \setminus S_1 \text{ and } s_i = 1, \\ y_3 & \text{if } i \in S_2 \setminus S_1 \text{ and } s_i = 0, \\ q_1^1 & \text{if } i \notin S_2 \text{ and } s_i = 1, \\ q_0^j & \text{if } i \notin S_2 \text{ and } s_i = 0. \end{cases}$$

For every $1 \le j \le 3k+1$, we add the constraint $c^3_{x,y,j} = R_2(v_1^j,\dots,v_r^j)$. For example, assume that $(0,1,0,1) \in R_2$ and this tuple has flip sets $S_1 = \{1,2\}$ and $S_2 = \{1,2,3,4\}$, but $S_2 \setminus S_1 = \{3,4\}$ is not a flip set. This means that $(0,1,0,1), (1,0,1,0), (1,0,0,1) \in R_2$ and $(0,1,1,0) \notin R_2$. In this case, constraint $c^3_{x,y,j}$ is $R_2(x_1,x_3,y_3,y_1)$. Assuming $f(x_1) \neq f(x_3)$ and $f(y_1) \neq f(y_3)$, any combination of values on x_1 and y_1 satisfies the constraint, except if $f(x_1) = f(y_1) = 0$. Thus the constraint effectively acts as a constraint $x_1 \vee y_1$.

Finally, we set the maximum allowed distance to $k' := 3k$. This completes the description of the constructed instance (ϕ_2, f_2, k').

Assume first that f_1' is a solution for the instance (ϕ_1, f_1, k). Define $f_2'(x_1) = f_2'(x_2) = f_1'(x)$ and $f_2'(x_3) = 1 - f_1'(x)$ for every $x \in V$, and define $f_2'(q_0^j) = 0$, $f_2'(q_1^j) = 1$ for every $1 \le j \le 3k+1$. The fact $w(f_1') < w(f_1)$ implies $w(f_2') < w(f_2)$. Furthermore, the distance of f_2 and f_2' is exactly three times the distance of f_1 and f_1', i.e., at most $3k$. We claim that f_2' satisfies the constraints of ϕ_2. This is easy to see for $c^1_{a,b,c}$ and $c^2_{x,\ell,j}$. For $c^3_{x,y,j}$, this can be seen as follows:

– If $f_2'(x) = 0$, $f_2'(y) = 1$, then this holds because $(s_1,\dots,s_r) \in R_2$.

– If $f'_2(x) = 1$, $f'_2(y) = 0$, then this holds because S_2 is a flip set.
– If $f'_2(x) = 1$, $f'_2(y) = 1$, then this holds because S_1 is a flip set.

For the other direction, assume that f'_2 is a solution for instance (ϕ_2, f_2, k'). Define $f'_1(x) = f'_2(x_1)$ for every $x \in V$; we claim that f'_1 is a solution for instance (ϕ_1, f_1, k). Since the distance of f_2 and f'_2 is at most $3k$, there are $1 \leq b, c \leq 3k+1$ with $f'_2(q^b_0) = 0$ and $f'_2(q^c_1) = 1$. Because of the constraint $c^1_{a,b,c}$, we have that $f'_2(q^a_1) = 1$ for every $1 \leq a \leq 3k + 1$. The constraints $c^2_{x,1,b}$ and $c^2_{x,2,b}$ ensure that $f'_2(x_1) = f'_2(x_2) = 1 - f'_2(x_3)$ (since $(0, 0, 0, 1) \notin R'_1$ and $(1, 1, 0, 1) \notin R'_1$). It follows that the distance of f_1 and f'_1 is at most k: $f_1(x) \neq f'_1(x)$ implies $f_2(x_\ell) \neq f'_2(x_\ell)$ for $\ell = 1, 2, 3$, hence this can hold for at most k different $x \in V$. Moreover, $w(f'_1) < w(f_1)$: this follows from the facts $w(f'_2) < w(f_2)$ and $f'_2(q^j_0) \geq f_2(q^k_0)$, $f'_2(q^j_1) \geq f_2(q^k_1)$ $(1 \leq j \leq 3k + 1)$.

We claim that every constraint $x \vee y$ of ϕ_1 is satisfied. Assume that $f'_1(x) = f'_1(y) = f'_2(x_1) = f'_2(y_1) = 0$. Now $c^3_{x,y,b}$ is not satisfied: this follows from the fact that $S_2 \setminus S_1$ is not a flip set for $(s_1, \ldots, s_r)$ (with respect to R_2). □

4 Characterizing polynomial-time solvability

In this section, we completely characterize those finite sets Γ of Boolean relations for which LS-CSP(Γ) is polynomial-time solvable.

Theorem 9. *Let Γ be a finite set of Boolean relations. The problem* LS-CSP(Γ) *is in* PTIME *if every relation in Γ is IHS-B− or every relation in Γ is width-2 affine. In all other cases,* LS-CSP(Γ) *is* NP*-hard.*

Proof. If every relation in Γ is IHS-B−, then Corollary 5 gives a polynomial-time algorithm. If every relation in Γ is width-2 affine then the following simple algorithm solves LS-CSP(Γ): for a given instance (ϕ, f, k), compute the graph whose vertices are the variables in ϕ and two vertices are connected if there is a constraint $=$ or $\neq$ in ϕ imposed on them. If there is a connected component of this graph which has at most k vertices and such that f assigns more 1's in this component than it does 0's, then flipping the values in this component gives a required lighter solution. If such a component does not exists, then there is no lighter solution within distance k from f.

By Lemma 8, if Γ contains a relation that is not Horn and a relation that is not flip separable then LS-CSP(Γ) is NP-hard. (Note that Lemma 8 is a polynomial-time reduction from an NP-hard problem.) Thus we can assume that every relation in Γ is Horn or every relation in Γ is flip separable. We prove only the former case; the proof for the latter case will appear in the full version.

Assume now that Γ is Horn, and there is a relation $R \in \Gamma$ that is not IHS-B−. We prove that LS-CSP($\{R\}$) is NP-hard. It is shown in the proof of Lemma 5.27 of [3] that then R is at least ternary and one can permute the coordinates in R and then substitute 0 and 1 in R in such a way that the ternary relation $R'(x, y, z) = R(x, y, z, 0, \ldots, 0, 1, \ldots, 1)$ has the following properties:

1. R' contains tuples $(1, 1, 1), (0, 1, 0), (1, 0, 0), (0, 0, 0)$, and
2. R' does not contain the tuple $(1, 1, 0)$.

Note that if $(0,0,1) \in R'$ then $R'(x,x,y)$ is $x \to y$. If $(0,0,1) \notin R'$ then, since R (and hence R') is Horn (i.e., min-closed), at least one of of the tuples $(1,0,1)$ and $(0,1,1)$ is not in R'. Then it is easy to check that at least one of the relations $R'(x,y,x)$ and $R'(y,x,x)$ is $x \to y$. Hence, we can use constraints of the form $x \to y$ when specifying instances of LS-CSP($\{R'\}$).

We reduce Minimum Dominating Set to LS-CSP($\{R'\}$). Let $G(V,E)$ be a graph with n vertices and m edges where a dominating set of size at most t has to be found. Let $v_1, \ldots, v_n$ be the vertices of G. Let $S = 3m$. We construct a formula with $nS + 2m + 1$ variables as follows:

- There is a special variable x.
- For every $1 \leq i \leq n$, there are S variables $x_{i,1}, \ldots, x_{i,S}$. There is a constraint $x_{i,j} \to x_{i,j'}$ for every $1 \leq j, j' \leq n$.
- For every $1 \leq i \leq n$, if $v_{s_1}, \ldots, v_{s_d}$ are the neighbors of v_i, then there are d variables $y_{i,1}, \ldots, y_{i,d}$ and the following constraints: $x_{s_1,1} \to y_{i,1}$, $R'(x_{s_2,1}, y_{i,1}, y_{i,2})$, $R'(x_{s_3,1}, y_{i,2}, y_{i,3})$, $\ldots$, $R'(x_{s_d,1}, y_{i,d-1}, y_{i,d})$, $R'(x_{i,1}, y_{i,d}, x)$.
- For every variable z, there is a constraint $x \to z$.

Observe that the number of variables of type $y_{i,j}$ is exactly $2m$. Setting every variable to 1 is a satisfying assignment. Set $k := St + S - 1$.

Assume that there is a satisfying assignment where the number of 0's is at most k (but positive). Variable x has to be 0, otherwise every other variable is 1. If $x_{i,1}$ is 0, then $x_{i,j}$ is 0 for every $1 \leq j \leq S$. Thus $k < S(t+1)$ implies that there are at most t values of i such that $x_{i,1}$ is 0. Let D consist of all vertices v_i such that $x_{i,1}$ is 0. We claim that D is a dominating set. Suppose that some vertex v_i is not dominated. This means that if $v_{s_1}, \ldots, v_{s_d}$ are the neighbors of v_i, then the variables $x_{s_1,1}, \ldots, x_{s_d,1}, x_{i,1}$ all have the value 1. However, this means that these variables force variables $y_{i,1}, \ldots, y_{i,d}$ and variable x to value 1, a contradiction. Thus D is a dominating set of size at most t.

The reverse direction is also easy to see. Assume that G has a dominating set D of size at most t. For every $1 \leq i \leq n$ and $1 \leq j \leq S$, set variable $x_{i,j}$ to 1 if and only v_i is not contained in D. Set x to 0. It is easy to see that this assignment can be extended to the variables $y_{i,j}$ to obtain a satisfying assignment: indeed, if $v_{s_1}, \ldots, v_{s_d}$ are the neighbors of v_i and none of them is in D then $v_i \in D$, and we set $y_{i,1} = \ldots = y_{i,d} = 1$. Otherwise, if j is minimal such that $v_{s_j} \in D$, we set $y_{i,1} = \ldots = y_{i,j-1} = 1$ and $y_{i,q} = 0$ for $q \geq j$. This satisfying assignment contains at most $St + 2m + 1 \leq k$ variables with value 0, as required.

Finally, we reduce LS-CSP($\{R'\}$) to LS-CSP($\{R\}$) (and so to LS-CSP(Γ)). Take an instance (ϕ, f, k) of LS-CSP($\{R'\}$), let V be the variables of ϕ and $c_1, \ldots, c_p$ the constraints of ϕ. We build an instance ϕ' of LS-CSP($\{R\}$) as follows.

1. For each $1 \leq i \leq \max(p, k+1)$, introduce new variables x_0^i, x_1^i.
2. For each constraint $c_i = R'(x,y,z)$ in formula ϕ, replace it by the constraint $R(x, y, z, x_0^i, \ldots, x_0^i, x_1^i, \ldots, x_1^i)$.
3. For each ordered pair (i,j) where $1 \leq i, j \leq \max(p, k+1)$, add the constraints $R(x_0^i, x_0^i, x_0^j, x_0^j, \ldots, x_0^j, x_1^j, \ldots, x_1^j)$ and $R(x_1^j, x_1^j, x_1^i, x_0^j, \ldots, x_0^j, x_1^j, \ldots, x_1^j)$.

Finally, extend f so that, for all i, we have $x_0^i = 0$ and $x_1^i = 1$. It is clear that the obtained mapping f' is a solution to the new instance. Note that, by the choice of R', the tuple $(1, 1, 0, 0, \ldots, 0, 1, \ldots, 1)$ does not belong to R. Hence, the constraints added in step (3) above ensure that if a variable of the form x_0^i or x_1^i in f' is flipped then, in order to get a solution to ϕ' different from f', one must flip at least one of x_0^i or x_1^i for each $1 \leq i \leq \max(p, k+1)$. Consequently, all solutions to ϕ' that lie within distance k from f' must agree with f' on all such variables. In other words, searching for such a solution, it makes sense to flip only variables from V. Thus, clearly, the instances (ϕ, f, k) and (ϕ', f', k) are equivalent. $\Box$

References

1. P. Chapdelaine and N. Creignou. The complexity of Boolean constraint satisfaction local search problems. *Annals of Mathematics and Artificial Intelligence*, 43:51–63, 2005.
2. D. Cohen and P. Jeavons. The complexity of constraint languages. In F. Rossi, P. van Beek, and T. Walsh, editors, *Handbook of Constraint Programming*, chapter 8. Elsevier, 2006.
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*. 2001.
4. P. Crescenzi and G. Rossi. On the Hamming distance of constraint satisfaction problems. *Theoretical Computer Science*, 288(1):85–100, 2002.
5. E. Dantsin, A. Goerdt, E. Hirsch, R. Kannan, J. Kleinberg, C. Papadimitriou, P. Raghavan, and U. Schöning. A deterministic $(2 - \frac{2}{k+1})^n$ algorithm for k-SAT based on local search. *Theoretical Computer Science*, 289:69–83, 2002.
6. R.G. Downey and M.R. Fellows. *Parameterized Complexity*. Springer, 1999.
7. M.R. Fellows. Parameterized complexity: new developments and research frontiers. In *Aspects of Complexity (Kaikura, 2000)*, volume 4 of *de Gruyter Series in Logic and Applications*, pages 51–72. 2001.
8. J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006.
9. J. Gu, P. Purdom, J. Franko, and B.W. Wah. *Algorithms for the Satisfiability Problem*. Cambridge University Press, Cambridge, MA, 2000.
10. E. Hirsch. SAT local search algorithms: worst-case study. *Journal of Automated Reasoning*, 24:127–143, 2000.
11. H. Hoos and E. Tsang. Local search methods. In F. Rossi, P. van Beek, and T. Walsh, editors, *Handbook of Constraint Programming*, chapter 5. Elsevier, 2006.
12. L. Kirousis and Ph. Kolaitis. The complexity of minimal satisfiability problems. *Information and Computation*, 187:20–39, 2003.
13. D. Marx. Parameterized complexity of constraint satisfaction problems. *Computational Complexity*, 14:153–183, 2005.
14. D. Marx. Searching the k-change neighborhood for TSP is W[1]-hard. *Operations Research Letters*, 36:31–36, 2008.
15. T.J. Schaefer. The complexity of satisfiability problems. In *STOC'78*, pages 216–226, 1978.