On the hardness of losing weight

ANDREI KROKHIN
Durham University, UK
and
DÁNIEL MARX
Tel Aviv University, Israel

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.

Categories and Subject Descriptors: F.2.0 [**Analysis of Algorithms and Problem Complexity**]: General

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Constraint satisfaction problem, local search, complexity, fixed-parameter tractability

1. INTRODUCTION

Local search is one of the most widely used approaches to solving hard optimization problems [Aarts and Lenstra 2003; Michiels et al. 2007]. 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). Eventually, the iteration gets stuck in a local optimum, and our hope is that this local optimum is close to the global optimum. Metaheurestic techniques such as simulated annealing are more elaborate variants of this simple scheme, but iteration of

Preliminary version of parts of this paper was published in the Proceedings of ICALP'08. The first author is supported by UK EPSRC grants EP/C543831/1 and EP/C54384X/1; the second author is partially supported by the Magyary Zoltán Felsőoktatási Közalapítvány, Hungarian National Research Fund grant OTKA 67651, and ERC Advanced Grant DMMCA. Part of the work was done during Dagstuhl Seminar 07281 and while the second author was affiliated with Budapest University of Technology and Economics, Hungary.

Authors' addresses: A. Krokhin, School of Engineering and Computing Sciences, Durham University, Durham, DH1 3LE, UK, email: andrei.krokhin@durham.ac.uk; D. Marx, School of Computer Science, Tel Aviv University, Tel Aviv, Israel. email: dmarx@cs.bme.hu.

local improvements is an essential feature of these algorithms. Furthermore, any optimization algorithm can be followed by a local search phase, thus the problem of finding a better solution locally is of significant practical interest. As a brute force search of a k-neighborhood is not feasible for large k, it is natural to study the complexity of searching the k-neighborhood.

There exists complexity theory for local search, with specialized complexity classes such as PLS (see [Johnson et al. 1988; Michiels et al. 2007]). However, it was recently suggested in [Fellows 2001; Marx 2008a] that the hardness of searching the k-neighborhood (for any optimization problem) can be studied very naturally in the framework of parameterized complexity [Downey and Fellows 1999; Flüm and Grohe 2006]; such a study was very recently performed for the traveling salesperson problem (TSP) [Marx 2008b], for two variants of the stable marriage problem [Marx and Schlotter 2009a; 2009b], for some graph-theoretic problems [Fellows et al. 2009], and for the Max Sat problem [Szeider 2009]. The primary goal of this paper is to contribute towards this line of research. 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 time $n^{O(k)}$ on n-vertex graphs 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 $O(2^k \cdot n^2)$ [Downey and Fellows 1999; Flüm and Grohe 2006]. In other words, for every fixed cover size there is a polynomial-time (in this case, quadratic in the number of vertices) 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 [Downey and Fellows 1999; Flüm and Grohe 2006], are very unlikely to be fixed-parameter tractable.

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 values that certain collections of variables can simultaneously take. 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 k-Sat allowing that constraints are given by arbitrary relations, not necessarily by clauses. Local search methods for Sat and CSP are very extensively studied (see, e.g., [Dantsin et al. 2002; Gu et al. 2000; Hirsch 2000; Hoos and Tsang 2006]).

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 problems with lower complexity in comparison with the general case (see [Cohen and Jeavons 2006; Creignou et al. 2001]). Such classifications are sometimes called Schaefer-style because the first classification of this type was obtained by T.J. Schaefer in his seminal work [Schaefer 1978]. A local-search related Schaefer-style classification for Boolean MAX CSP was obtained in [Chapdelaine and Creignou 2005], in the context of complexity classes such as PLS. 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 [Marx 2005].

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 5.1) and for parameterized complexity (Theorem 4.1). 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 contains auxiliary technical results. Section 4 presents the classificiation with respect to fixed-parameter tractability, while Section 5 deals with polynomial-time solvability.

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 containing only non-empty relations. For a fixed finite Γ, every Γ-formula $\phi = (V, C)$ can be represented with length polynomial in $|V|$ and $|C|$: each constraint relation can be represented by constant number of bits (depending only on Γ). 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 where the two assignments differ, i.e., $f_1(x) \neq f_2(x)$.

Given an r-ary boolean relation R and a set of indices $J = \{i_1, \ldots, i_q\} \subseteq \{1, \ldots, r\}$, the *projection* of R onto J, denoted $\mathrm{pr}_J(R)$, is the relation defined as follows: $\{(y_{i_1}, \ldots, y_{i_q}) \mid \text{ there is } (x_1, \ldots, x_r) \in R \text{ with } y_{i_j} = x_{i_j}, 1 \leq j \leq q\}$.

We recall various standard definitions concerning Boolean constraints (cf. [Creignou et al. 2001]):

—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. We denote by EVEN_r the r-ary relation $x_1 + x_2 + \cdots + x_r = 0$ and by ODD_r the r-ary relation $x_1 + x_2 + \cdots + x_r = 1$.

—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 $(\neg x_1 \vee \ldots \vee \neg x_n)$, $n \geq 1$.

—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 2.1. *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 $p \leq q$. To see this, assume that S_1, S_2 are flip sets of $(a_1, \ldots, a_q)$. Define $X \subseteq \{1, \ldots, q\}$ such that $i \in X$ if and only if $a_i = 1$. From the fact that $(a_1, \ldots, a_q) \in R_{p\text{-IN-}q}$ and S_1, S_2 are flip sets, we have that $|X| = |X \bigtriangleup S_1| = |X \bigtriangleup S_2| = p$ (where $\bigtriangleup$ denotes the symmetric difference). With a straightforward calculation, it follows that $|X \bigtriangleup (S_2 \setminus S_1)| = p$, i.e., $S_2 \setminus S_1$ is also a flip set. Affine constraints are also flip separable: to see this, it is sufficient to verify the definition only for the constraints EVEN_r and ODD_r, since a conjunction of flip separable relations is again such a relation.

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 [Schaefer 1978]. 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 [Creignou et al. 2001; Crescenzi and Rossi 2002; Khanna et al. 2001]. 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 [Creignou et al. 2001; Khanna et al. 2001]. Here we state only the classification of polynomial-time solvable and NP-hard cases:

THEOREM 2.2 [CREIGNOU ET AL. 2001; KHANNA ET AL. 2001]. *Let Γ be a set of Boolean relations.* MIN-ONES(Γ) *is solvable in polynomial time if one of 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 [Crescenzi and Rossi 2002], motivated by the blocks world problem from knowledge representation, 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 [Kirousis and Kolaitis 2003], motivated by the circumscription formalism from artificial intelligence.

The main focus of the paper is the local search version of minimizing weight. Following [Marx and Schlotter 2009a; 2009b], we consider two variants of the problem: "strict" and "permissive," defined as follows:

sLS-CSP(Γ)	
Input:	A Γ-formula ϕ, a satisfying assignment f, and an integer k.
Goal:	Find a satisfying assignment f' to ϕ with $w(f') < w(f)$ and $\mathrm{dist}(f, f') \leq k$ or report that no such assignment exists.

pLS-CSP(Γ)	
Input:	A Γ-formula ϕ, a satisfying assignment f, and an integer k.
Goal:	Find a satisfying assignment f' to ϕ with $w(f') < w(f)$ or report that there is no such assignment with $\mathrm{dist}(f, f') \leq k$.

LS in the above problems stands for both "local search" and "lighter solution." The difference between the variants is that strict local search is strictly restricted to the neighborhood of the given satisfying assignment, while permissive local search allows one to produce an *arbitrary* better solution (even if there is no better solution in the given neighborhood). Observe that any algorithm for the strict version is a valid algorithm for the permissive version as well, thus it might happen that the strict version is hard while the permissive version is easy. However, from the viewpoint of local-search based optimization techniques, having a permissive local search algorithm is at least as useful as the strict version.

We distinguish between the two variants mainly for the following reason: there are situations when it easy to find an optimal solution to an instance despite strict local search being hard. Thus the study of strict local search can give counterintuitive results by showing hardness for problems that are actually easy. On the other hand, an algorithm that finds an optimum solution for MIN-ONES(Γ) would also solve pLS-CSP(Γ), i.e., permissive local search is always easy if the optimum can be found easily.

Note that sLS-CSP(Γ) and pLS-CSP(Γ) are defined as search problems, not as decision problems. Recall that a search problem is NP-hard if there is a polynomial-time Turing reduction from some NP-complete problem to it, i.e., given a polynomial-time subroutine for solving the search problem, we can solve an NP-hard decision problem in polynomial time. W[1]-hardness is interpreted analogously for search problems. To prove hardness of pLS-CSP(Γ), we will use the following decision problem. We say that a satisfying assignment f to a formula is *suboptimal* if it is not optimal, i.e., the formula has a satisfying assignment with less weight than f.

LIMP(Γ)	
Input:	A Γ-formula ϕ, a satisfying assignment f, and an integer k such that either f is optimal or else there is a satisfying assignment f' to ϕ with $w(f') < w(f)$ and $\text{dist}(f, f') \leq k$.
Question:	Is f suboptimal?

In other words, this (promise) problem is to distinguish between optimal satisfying assignments and those that can be improved locally. It is easy to see that an algorithm for pLS-CSP(Γ) would solve LIMP(Γ). Hence, NP-hardness or W[1]-hardness of LIMP(Γ) implies hardness in the same sense of pLS-CSP(Γ) and of sLS-CSP(Γ). When reducing a decision problem P to LIMP(Γ), we have to provide a mapping with the following two properties: (1) every yes-instance of P is mapped to an instance (ϕ, f, k) where f can be improved locally, and (2) every no-instance of P is mapped to an instance (ϕ, f, k) where f cannot be improved at all. Usually we will prove the contrapositive of (2): if an instance x of P is mapped to (ϕ, f, k) such that f is suboptimal, then x is a yes-instance. Note that (1) and (2) together imply that every constructed instance (ϕ, f, k) satisfies the requirement that if f is suboptimal, then it can be improved locally.

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 sLS-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 (Proposition 4.7), 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. This might be of independent interest.

Parameterized complexity. In a *parameterized 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 [Downey and Fellows 1999; Flüm and Grohe 2006] 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 some computable 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.

The most important difference between parameterized reductions and classical polynomial-time many-to-one reductions is the second requirement: in most NP-completeness proofs the new parameter is not a function of the old parameter. Therefore, finding parameterized reductions is usually more difficult, and the constructions have somewhat different flavor than classical reductions. In general, a parameterized reduction is not necessarily a polynomial-time reduction (since the third requirement is weaker than polynomial-time). However, the reductions presented in Section 3 are both parameterized and polynomial-time. Hence they will be used in both Section 4 (to prove W[1]-hardness) and Section 5 (to prove NP-hardness).

3. SOME BASIC REDUCTIONS

This section contains auxiliary technical results that will be used in subsequent sections.

Let C_0 and C_1 denote the unary relations $\{0\}$ and $\{1\}$, respectively.

Lemma 3.1. *For any* Γ, sLS-CSP(Γ) *and* sLS-CSP($\Gamma \cup \{C_0\}$) *are equivalent via polynomial-time parameterized reductions. The same holds for problems* LImp(Γ) *and* LImp($\Gamma \cup \{C_0\}$).

Proof. Let us describe a reduction from sLS-CSP($\Gamma \cup \{C_0\}$) to sLS-CSP(Γ). Let (ϕ, f, k) be an instance of the former problem. Order the variables in ϕ so that $x_1, \ldots, x_\ell$ are all variables x_i such that ϕ contains the constraint $C_0(x_i)$. Now produce a new instance ϕ' of sLS-CSP(Γ) as follows: remove all constraints involving

C_0, introduce a new variable y and replace all occurrences of $x_1, \ldots, x_\ell$ in ϕ by y. Then, introduce $k+1$ new variables $y_1, \ldots, y_{k+1}$, and replace, in ϕ', every constraint involving y by $k+1$ copies of the constraint such that the i-th copy contains y_i instead of y and all other variables unchanged. Call the resulting instance ϕ''. Consider the solution f'' to ϕ'' that maps each y_i to 0 and coincides with f everywhere else. It is not hard to see that the instance (ϕ'', f'', k) of sLS-CSP(Γ) is equivalent to (ϕ, f, k) because any solution g to ϕ'' with $w(g) < w(f'')$ and $\text{dist}(g, f'') \leq k$ maps at least one y_i, $1 \leq i \leq k+1$, to 0.

Furthermore, it is easy to see that this reduction is also a reduction from LImp$(\Gamma \cup \{C_0\})$ to LImp(Γ). □

Lemma 3.2. *For any Γ containing a relation that is not 0-valid,* LImp(Γ) *and* LImp$(\Gamma \cup \{C_0, C_1\})$ *are equivalent via polynomial-time parameterized reductions.*

Proof. By Lemma 3.1, we can assume that $C_0 \in \Gamma$. If $R \in \Gamma$ is non-0-valid then we can without loss of generality assume that $R(x, \ldots, x, 0, \ldots, 0)$ holds if and only if $x = 1$. Now the reduction from LImp$(\Gamma \cup \{C_0, C_1\})$ to LImp(Γ) is obvious: for a given instance, introduce a new variable z together with constraint $C_0(z)$ and replace each constraint of the form $C_1(x_i)$ by $R(x_i, , \ldots, x_i, z, \ldots, z)$. The given solution is transformed in the obvious way, and the parameter stays the same. □

Lemma 3.3. *For any Γ containing a relation that is not Horn,* sLS-CSP(Γ) *and* sLS-CSP$(\Gamma \cup \{C_0, C_1\})$ *are equivalent via polynomial-time parameterized reductions.*

Proof. By Lemma 3.1, we can assume that $C_0 \in \Gamma$. Let $R \in \Gamma$ be non-Horn. Since R 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

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

then $(0,1,0,1), (1,0,0,1) \in R'$, but $(0,0,0,1) \notin R'$. Since R' is obtained from R by identifying variables, we can use the relation R' when specifying instances of sLS-CSP(Γ).

Let (ϕ, f, k) be an instance of sLS-CSP$(\Gamma \cup \{C_0, C_1\})$. Let us construct a formula ϕ' that has every variable of V and new variables q_0, q_1^j for $1 \leq j \leq k+1$ (these new variables will play the role of the constants). First, for every variable $x \in V$ such that ϕ contains constraint $C_1(x)$, we remove the constraint from ϕ and replace all other occurrences of x in ϕ by q_1^1. Then we add constraints $C_0(q_0)$ and $R'(q_1^a, q_0, q_0, q_1^b)$ for $1 \leq a, b \leq k+1$. Call the obtained formula ϕ'. We define assignment f' for ϕ' by setting $f'(x) = f(x)$ for $x \in V$, $f'(q_0) = 0$ and $f_2(q_1^j) = 1$ for $1 \leq j \leq k+1$. Clearly, f' satisfies ϕ'. Moreover, by the choice of R', any satisfying assignment to ϕ' that maps one of the variables q_1^j to 0 would have to map all of such variables to 0. Therefore, any solution to ϕ' within distance k from f' must coincide with f' on all variables not in V. Moreover, it is easy to see that such solutions are in one-to-one correspondence with solutions to ϕ within distance k from f. Thus, the instances (ϕ, f, k) and (ϕ', f', k) are equivalent. □

Lemma 3.4. *If R is a non-Horn relation then, by identifying coordinates and substituting constants in R, it is possible to express at least one of the relations*

$x \vee y$ and $x \neq y$.

PROOF. Consider the relation R' obtained as in the previous proof. It is easy to see that the relation $R'(x, y, 0, 1)$ is one of the required relations. $\square$

LEMMA 3.5. *Let R be the set of solutions to a Γ-formula ϕ, and let $R' = \mathrm{pr}_J(R)$ where $J \supseteq \{j \mid \mathrm{pr}_j(R) = \{0,1\}\}$. Then the problems* sLS-CSP(Γ) *and* sLS-CSP($\Gamma \cup \{R'\}$) *are equivalent via polynomial-time parameterized reductions. The same holds for* LImp(Γ) *and* LImp($\Gamma \cup \{R'\}$).

PROOF. Note that, for any fixed $j \notin J$, each solution to ϕ takes the same value on the corresponding variable. In every instance of sLS-CSP($\Gamma \cup \{R'\}$), every constraint of the form $c = R'(\overline{s})$ can be replaced by the constraints from ϕ where variables from $\overline{s}$ keep their places, while all other variables are new and do not appear elsewhere. This transformation (with the distance k unchanged) is the required polynomial-time parameterized reduction. $\square$

The following corollary is a special case of Lemma 3.5:

COROLLARY 3.6. *If $C_0, C_1 \in \Gamma$ and R' can be obtained from some $R_0 \in \Gamma$ by substitution of constants, then the problems* sLS-CSP(Γ) *and* sLS-CSP($\Gamma \cup \{R'\}$) *are equivalent via polynomial-time parameterized reductions. The same holds for* LImp(Γ) *and* LImp($\Gamma \cup \{R'\}$).

For an n-ary tuple $\overline{s} = (a_1, \ldots, a_n)$ and $1 \leq i \leq n$, we define the $(n+1)$-ary tuple $\alpha_i(\overline{s}) = (a_1, \ldots, a_n, 1 - a_i)$ and the n-ary tuple $\beta_i(\overline{s}) = (a_1, \ldots, a_{i-1}, 1 - a_i, a_{i+1}, \ldots, a_n)$. For an n-ary relation R, let $\alpha_i(R)$ denote the $(n+1)$-ary relation defined by $\overline{s} \in R \Leftrightarrow \alpha_i(\overline{s}) \in \alpha_i(R)$ and let $\beta_i(R)$ denote the n-ary relation defined by $\overline{s} \in R \Leftrightarrow \beta_i(\overline{s}) \in \beta_i(R)$. Note that a constraint $\alpha(R)(x_1, \ldots, x_n, x_n + 1)$ is equivalent to two constraints: $R(x_1, \ldots, x_n)$, $x_i \neq x_{n+1}$.

LEMMA 3.7. *For any R, the following pairs of problems are equivalent via polynomial-time parameterized reductions:*

(*1*) *the problems* sLS-CSP($\{R, \neq\}$) *and* sLS-CSP($\{\alpha_i(R), \neq\}$), *and*

(*2*) *the problems* sLS-CSP($\{R, \neq\}$) *and* sLS-CSP($\{\beta_i(R), \neq\}$).

The same holds for the problem LImp.

PROOF. It is clear that sLS-CSP($\{\alpha_i(R), \neq\}$) reduces to sLS-CSP($\{R, \neq\}$), by simply replacing each constraint involving $\alpha_i(R)$ by its definition via R and $\neq$. Since $R = \beta_i(\beta_i(R))$, the two directions of the second statement are equivalent. Thus all we have to show is that sLS-CSP($\{R, \neq\}$) can be reduced to both sLS-CSP($\{\alpha_i(R), \neq\}$) and sLS-CSP($\{\beta_i(R), \neq\}$).

Let (ϕ, f, k) be an instance of sLS-CSP($\{R, \neq\}$) where ϕ is over a set V of variables. For each variable $x \in V$, introduce two new variables x', x'' along with constraints $x \neq x'$, $x' \neq x''$. Replace every constraint of the form $R(x_{j_1}, \ldots, x_{j_n})$ in ϕ by $\alpha_i(R)(x_{j_1}, \ldots, x_{j_n}, x'_{j_i})$ or $\beta_i(R)(x_{j_1}, \ldots, x_{j_{i-1}}, x'_{j_i}, x_{j_{i+1}}, \ldots, x_{j_n})$, and leave all other constraints in ϕ unchanged. Let ϕ' be the obtained instance of CSP($\{\alpha_i(R), \neq\}$) or CSP($\{\beta_i(R), \neq\}$). Clearly, f has a unique extension to a solution f' to ϕ' and if f'_1 and f'_2 are the extensions of f_1 and f_2, respectively, then $\mathrm{dist}(f'_1, f'_2) = 3\mathrm{dist}(f_1, f_2)$. It is clear that the instance $(\phi', f', 3k)$ of sLS-CSP($\{\alpha_i(R), \neq\}$) or pLS-CSP($\{\beta_i(R), \neq\}$) is equivalent to (ϕ, f, k). The same reduction works for LImp. $\square$

4. CHARACTERIZING FIXED-PARAMETER TRACTABILITY

In this section, we completely characterize those finite sets Γ of Boolean relations for which problems sLS-CSP(Γ) and pLS-CSP(Γ) are fixed-parameter tractable.

Theorem 4.1. *Let Γ be a set of Boolean relations. The problem* sLS-CSP(Γ) *is in* FPT *if one of the following holds, and* W[1]*-hard otherwise:*

—*Every $R \in \Gamma$ is Horn.*

—*Every $R \in \Gamma$ is flip separable.*

Theorem 4.2. *Let Γ be a set of Boolean relations. The problem* pLS-CSP(Γ) *is in* FPT *if one of the following holds, and* W[1]*-hard otherwise:*

—*Every $R \in \Gamma$ is 0-valid.*

—*Every $R \in \Gamma$ is Horn.*

—*Every $R \in \Gamma$ is flip separable.*

First we handle the fixed-parameter tractable cases (Lemmas 4.3 and 4.5) in Theorem 4.1. It is easy to see that the FPT part of Theorem 4.2 follows from the FPT part of Theorem 4.1 and (for the 0-valid case) from Theorem 2.2.

Lemma 4.3. *If every $R \in \Gamma$ is Horn, then* sLS-CSP(Γ) *is* FPT.

Proof. If there is a solution f' for the sLS-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 $\text{dist}(f'', f) \leq \text{dist}(f', f)$. 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 $\langle (x_1, \dots, x_r), R\rangle$ that is not satisfied by the new assignment, then we select one of the variables $x_1, \dots, x_r$ 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 $(r_{\max} - 1)^{k+1} \cdot n^{O(1)}$. $\square$

If every $R \in \Gamma$ is not only Horn, but IHS-B− (which is a subset of Horn), then the algorithm of Lemma 4.3 actually runs in polynomial time:

Corollary 4.4. *If every $R \in \Gamma$ is IHS-B−, then* sLS-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 of its variables has the value 1. Thus there is no branching involved in the algorithm of Lemma 4.3, making it a polynomial-time algorithm. $\square$

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 4.5. *If every $R \in \Gamma$ is flip separable, then* sLS-CSP(Γ) *is* FPT.

PROOF. Let (ϕ, f, k) be an instance of sLS-CSP(Γ). If $w(f') < w(f)$ for some assignment f', then 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 (if the weight of the assignment is strictly less than $w(f)$) 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 $|X| \le k$. 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 $f'(x_i)$ (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 4.3, the branching factor of the tree is at most $r_{\max} - 1$, and the algorithm runs in time $(r_{\max} - 1)^{k+1} \cdot n^{O(1)}$. $\square$

All the hardness proofs in this section are based on the fact that LIMP($\{x \vee y\}$) is W[1]-hard , which we show in the following lemma. Note that MIN-ONES($\{x \vee y\}$) corresponds to the MINIMUM VERTEX COVER problem: the variables represent the vertices and constraint $(x \vee y)$ corresponds to an edge xy, representing the requirement that at least one of x and y has to be in the vertex cover. Thus the following hardness proof shows also that it is W[1]-hard to find a smaller vertex cover in the k-neighborhood of a given vertex cover.

LEMMA 4.6. *The problem* LIMP($\{x \vee y\}$) *is* W[1]*-hard.*

PROOF. The proof is by reduction from MAXIMUM INDEPENDENT SET: given a graph $G(V,E)$ and an integer t, we have to decide whether G has an independent set of size t. Let n be the number of vertices of G and let m be the number of edges. We construct a formula as follows. The variables $x_1, \dots, x_n$ correspond to the vertices of G and there are $t-1$ additional variables $y_1, \dots, y_{t-1}$. For every edge $v_{i_1}v_{i_2}$ of G, we add the constraint $x_{i_1} \vee x_{i_2}$ on the corresponding two variables. Furthermore, we add all the constraints $x_i \vee y_j$ for $1 \le i \le n$, $1 \le j \le t-1$. Let us define the assignment f such that $f(x_i) = 1$ for every $1 \le i \le n$ and $f(y_j) = 0$ for every $1 \le j \le t-1$.

Set $k := 2t-1$. Suppose first that G has an independent set of size t. Set the corresponding t variables x_i to 0 and set the variables $y_1, \dots, y_{t-1}$ to 1. This gives a satisfying assignment of weight $w(f) - 1$: if some constraint $x_{i_1} \vee x_{i_2}$ is not satisfied, then this would mean the there is an edge $v_{i_1}v_{i_2}$. Thus f is suboptimal and there is a lighter solution at distance at most k from f.

Suppose now that there is a solution f' with $w(f') < w(f)$ and $\text{dist}(f, f') \le k$. If some variable x_i is 0 in f', then every variable y_j has value 1. Thus the only way to decrease the weight of f is to set all the variables y_j to 1 and set at least t of the variables $x_1, \dots, x_n$ to 0. The at least t variables that were set to 0 correspond to an independent set of size at least t in G: if there were an edge $v_{i_1}v_{i_2}$ between two such vertices, then the constraint $x_{i_1} \vee x_{i_2}$ would not be satisfied. Thus if f is suboptimal, then G has an independent set of size t. $\square$

Lemma 4.6 shows that (both strict and permissive) local search for MINIMUM VERTEX COVER is W[1]-hard. Since the complement of a vertex cover is an independent set (and vice versa), this result implies that local search for MAXIMUM INDEPENDENT SET is also W[1]-hard. For bipartite graphs, both problems are polynomial-time solvable, thus (trivially) permissive local search can be done in polynomial time. However, as the following proposition shows, strict local search for these problems is W[1]-hard in case of bipartite graphs. Although we do not use this result in the rest of the paper, it might be of independent interest, as it gives a natural example where the complexity of strict and permissive local search differs.

PROPOSITION 4.7. *The problem* sLS-CSP($\{x \vee y\}$) *is* W[1]*-hard, even if the instance is bipartite, i.e., the variables can be partitioned into two sets X, Y such that every constraint $x \vee y$ satisfies $x \in X$ and $y \in Y$.*

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 $v_1, \dots, v_d$ 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 $x_1, \dots, x_{d-1}$. 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 $t(t-1)/2 = dt$ 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 and $x \in K$. 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 $d|K| - 1 < |E(K)| \le |K|(|K|-1)/2$, which is only possible if $|K| \ge 2d+1 = t$. If $|K| > t$, then $f'(v_i) > f(v_i)$ for at least $(t+1)d - 1$ variables, hence there must be more than that many variables u_e with $f'(u_e) < f(u_e)$. Thus the distance of f and f' is at least $2(t+1)d - 1 > t(t-1) - 1$. Therefore, we can assume $|K| = t$. Now $dt - 1 < |E(K)| \le |K|(|K|-1)/2 = t(t-1)/2$ 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., $x \in K$). □

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

LEMMA 4.8. (*1*) *If* Γ *contains a relation that is not Horn and a relation that is not flip separable, then* sLS-CSP(Γ) *is* W[1]*-hard.*

(*2*) *If* Γ *contains a relation that is not Horn, a relation that is not flip separable and a relation that is not 0-valid, then* LIMP(Γ) *is* W[1]*-hard.*

PROOF. We prove the second item first, by reduction from LIMP($\{x \vee y\}$). By Lemmas 3.1 and 3.2, we can assume that Γ contains both C_0 and C_1. Moreover, Lemma 3.4 implies that it is possible to identify variables and substitute constants in the non-Horn relation to obtain $x \vee y$ or $x \neq y$. Since $C_0, C_1 \in \Gamma$, we may by Corollary 3.6 assume that Γ contains $x \vee y$ or $x \neq y$. In the former case, we are done by Lemma 4.6, so assume that the disequality relation is in Γ.

Let $R \in \Gamma$ be an r-ary relation that is not flip separable. This means that there is a tuple $\overline{s} = (s_1, \ldots, s_r) \in R$ that has flip sets $S_1 \subset S_2$, but $S_2 \setminus S_1$ is not a flip set. We can assume that $S_2 = \{1, \ldots, r\}$: otherwise, for every coordinate $i \notin S_2$, let us substitute into R the constant s_i. Since $C_0, C_1 \in \Gamma$, by Corollary 3.6 we can assume that the resulting relation is in Γ and it is clear that now we can choose $\overline{s}$, S_1, S_2 such that S_2 contains all the coordinates of the relation.

We show that LIMP($\{R, \neq\}$) is W[1]-hard. Note that if a set S is flip set of $\overline{s}$ with respect to R, then S is a flip set of $\beta_i(\overline{s})$ with respect to $\beta_i(R)$ (where β_i is as defined before Lemma 3.7). By repeated applications of the operations β_i, we can obtain a relation R' such that there is a tuple $\overline{s}' \in R'$ having the same flip sets with respect to R' as $\overline{s}$ has with respect to R and $\overline{s}'$ is 0 at every coordinate in S_1 and 1 at every coordinate in $S_2 \setminus S_1$. Let $R''(x, y)$ be the binary relation obtained by substituting x into R' at every coordinate in S_1 and y at every coordinate in $S_2 \setminus S_1$. It is easy to verify that R'' is exactly the relation $x \vee y$. Indeed, $(0, 1) \in R''$ since $\overline{s}' \in R'$, $(1, 1) \in R''$ since S_1 is a flip set of $\overline{s}'$, $(1, 0) \in R''$ since S_2 is a flip set of $\overline{s}'$, and $(0, 0) \notin R''$ since $S_2 \setminus S_1$ is not a flip set of $\overline{s}'$. Thus by Lemma 4.6, LIMP($\{R', \neq\}$) is W[1]-hard, and, by Lemma 3.7, LIMP($\{R, \neq\}$) is also W[1]-hard.

To prove the first claim of the lemma, simply notice that, by Lemma 3.3, we can again assume that Γ contains C_0 and C_1, in which case the first claim follows from the second claim. $\square$

5. CHARACTERIZING POLYNOMIAL-TIME SOLVABILITY

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

THEOREM 5.1. *Let Γ be a set of Boolean relations. The problem* sLS-CSP(Γ) *is in* PTIME *if one of the following holds, and* NP*-hard otherwise:*

—*Every $R \in \Gamma$ is IHS-B−.*
—*Every $R \in \Gamma$ is width-2 affine.*

THEOREM 5.2. *Let Γ be a set of Boolean relations. The problem* pLS-CSP(Γ) *is in* PTIME *if one of the following holds, and* NP*-hard otherwise:*

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

Note that the tractability part of Theorem 5.2 trivially follows from that of Theorem 2.2. We now prove the tractability part of Theorem 5.1.

LEMMA 5.3. *If every relation in Γ is IHS-B− or every relation in Γ is width-2 affine then* sLS-CSP(Γ) *is in* PTIME.

PROOF. If every relation in Γ is IHS-B−, then Corollary 4.4 gives a polynomial-time algorithm. If every relation in Γ is width-2 affine then the following simple algorithm solves sLS-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 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. $\square$

We begin our hardness proofs in this section by showing (Lemma 5.4) that sLS-CSP($\{R\}$) is NP-hard whenever R is Horn, but not IHS-B−. Then we reason as follows. We can now assume that Γ contains a relation that is not Horn and a relation that is not width-2 affine (and in case of Theorem 5.2, also a relation that is not 0-valid). By Lemmas 3.1, 3.2, 3.3, we may assume that $C_0, C_1 \in \Gamma$, and we can also assume that the disequality relation is in Γ, by Lemmas 3.5 and 4.6. Note that Lemma 4.8 actually gives a polynomial-time reduction from an NP-hard problem. Therefore, we will prove both Theorem 5.1 and Theorem 5.2 if we show that LImp($\{R, \neq, C_0, C_1\}$) is NP-hard whenever R is flip separable, but not width-2 affine. We do this in Proposition 5.5.

Lemma 5.4. *The problem* sLS-CSP($\{R\}$) *is* NP-*hard whenever R is Horn, but not IHS-B−.*

Proof. It is shown in the proof of Lemma 5.27 of [Creignou et al. 2001] that 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 $R'(x,y,z) = R(x,y,z,0,\dots,0,1,\dots,1)$ 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 sLS-CSP($\{R'\}$).

We reduce Minimum Dominating Set to sLS-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 (a dominating set is a subset D of the vertices such that every vertex is either in D or has a neighbor in D). Let $v_1, \dots, v_n$ 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 $1 \le i \le n$, there are S variables $x_{i,1}, \dots, x_{i,S}$. There is a constraint $x_{i,j} \to x_{i,j'}$ for every $1 \le j, j' \le n$.
—For every $1 \le i \le n$, if $v_{s_1}, \dots, v_{s_d}$ are the neighbors of v_i, then there are d variables $y_{i,1}, \dots, y_{i,d}$ and the following constraints: $x_{s_1,1} \to y_{i,1}$, $R'(x_{s_2,1}, y_{i,1}, y_{i,2})$, $R'(x_{s_3,1}, y_{i,2}, y_{i,3})$, $\dots$, $R'(x_{s_d,1}, y_{i,d-1}, y_{i,d})$, $R'(x_{i,1}, y_{i,d}, x)$.
—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 $1 \le j \le S$. 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 sLS-CSP($\{R'\}$) to sLS-CSP($\{R\}$) (and so to sLS-CSP(Γ)). Take an instance (ϕ, f, k) of sLS-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 sLS-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 and 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. □

The following proposition completes the proofs of Theorems 5.1 and 5.2.

Proposition 5.5. *If R is a flip separable relation that is not width-2 affine then* LImp($\{R, \neq, C_0, C_1\}$) *is* NP-*hard.*

This proposition will be proved through a sequence of lemmas, the main lemmas being Lemma 5.7 and Lemma 5.9.

We consider affine constraints first.

Lemma 5.6. LImp($\{\mathrm{ODD}_3, \neq, C_0, C_1\}$) *is* NP-*hard.*

Proof. By using Lemma 3.5 with instance $\mathrm{ODD}_3(x, y, z), C_1(z)$, we may assume that the equality relation is also available.

The hardness proof is by reduction from CSP($R_{\text{1-IN-3}}$), which is known to be NP-hard even if every variable appears in exactly 3 constraints and each constraint contains 3 distinct variables [Moore and Robson 2001]. (This implies that the number of variables equals the number of constraints and the weight of every solution is exactly $n/3$, where n is the number of variables.) Given a CSP($R_{\text{1-IN-3}}$) formula ϕ with n variables $x_1, \ldots, x_n$, we construct a Γ-formula ϕ' with variables

—$x_{i,j}$ for $1 \leq i \leq n$, $1 \leq j \leq 2n$,
—v_t for every $1 \leq t \leq n$,
—y_j for every $0 \leq j \leq (2/3)n^2 + 2$.

For every $1 \leq i \leq n$, $1 \leq j, j' \leq 2n$, we add the constraint $x_{i,j} = x_{i,j'}$. For every $1 \leq j, j' \leq (2/3)n^2 + 2$, we add the constraint $y_j = y_{j'}$. For every $1 \leq t \leq n$, if the t-th constraint in ϕ is on variables x_a, x_b, x_c, then let us add the constraints $\text{ODD}_3(x_{a,1}, x_{b,1}, v_t)$ and $\text{ODD}_3(x_{c,1}, v_t, y_0)$. Finally, we add the constraint $y_0 \neq y_1$. Define assignment f such that $f(v_t) = 1$ for $1 \leq t \leq n$ and $f(y_j) = 1$ for $1 \leq i \leq (2/3)n^2 + 2$, and every other variable is 0. The weight of f is $(2/3)n^2 + n + 2$. Set k to be equal to the number of variables in ϕ' (i.e., we do not care about how many variables we flip in this case). This completes the description of the reduction.

Assume that ϕ has a solution f_0. Define f' such that $f'(x_{i,j}) = f_0(x_i)$ ($1 \leq i \leq n$, $1 \leq j \leq 2n$), $f'(y_0) = 1$, $f'(y_j) = 0$ ($1 \leq j \leq (2/3)n^2 + 2$). This assignment can be extended to each v_t in a unique way: if the t-th constraint in ϕ is on variables x_a, x_b, x_c, then exactly two of the variables $x_{a,1}, x_{b,1}, x_{c,1}, y_0$ have value 1, hence we can set v_t accordingly. Thus we can obtain a satisfying assignment f' this way. Observe that the weight of f_0 is exactly $n/3$: each variable with value 1 in f_0 appears in exactly 3 constraints and each constraint contains exactly one such variable. Thus the weight of f' is at most $2n \cdot n/3 + 1 + n$, strictly less than the weight of f.

Assume now that ϕ' has a satisfying assignment f' with $w(f') < w(f)$ and prove that ϕ is satisfiable. We claim that $f'(y_0) = 1$ and $f'(y_i) = 0$ ($1 \leq i \leq (2/3)n^2 + 2$) for any satisfying assignment f' with $w(f') < w(f)$. Indeed, otherwise, $w(f') < w(f)$ would imply that $f'(v_t) = 0$ for at least one $1 \leq t \leq n$. But this means that there is at least one $1 \leq i \leq n$ such that $f'(x_{i,j}) = 1$ for every $1 \leq j \leq 2n$. Thus the weight of f' would be at least $(2/3)n^2 + 2n > (2/3)n^2 + n + 2$, a contradiction (since $n \geq 3$).

Define $f_0(x_i) := f'(x_{i,1})$ for every $1 \leq i \leq n$. We cannot have $w(f_0) > n/3$ because, otherwise, we have at least $2n(n/3+1)$ variables $x_{i,j}$ that are equal to 1 in f', and $w(f') \geq 2n(n/3 + 1) \geq w(f)$ would follow. Hence $w(f_0) \leq n/3$. Let x_a, x_b, x_c be the variables in the t-th constraint in ϕ. The facts $f'(x_{a,1}) + f'(x_{b,1}) + f'(v_t) = 1$, $f'(x_{c,1}) + f'(v_t) + f'(y_0) = 1$ and $f'(y_0) = 1$ imply that at least one of $f'(x_{a,1})$, $f'(x_{b,1})$, $f'(x_{c,1})$ is 1. Thus if we denote by X the set of those variables of ϕ that have value 1 in f_0, then each constraint in ϕ contains at least one variable of X. Moreover, it is not possible that a constraint contains more than one variables of X. To see this, observe that each variable is contained in 3 constraints, thus $|X| \leq n/3$ implies that the variables of X appear in at most n constraints. However, we have seen that each constraint contains at least one variable of X, hence the variables of X appear in *exactly* n constraints. Equality is possible only if any two variables of X appear in disjoint constraints, that is, no constraint contains more than one variables from X. Therefore, f_0 is a solution for the instance ϕ. $\square$

LEMMA 5.7. $\text{LIMP}(\{R, \neq, C_0, C_1\})$ *is* NP*-hard if R is affine, but not of width 2.*

PROOF. Let R' be a minimium arity relation which can be obtained from R by substituting constants and identifying variables (note that such a relation is also affine) and which is not of width 2. We claim that R' can be obtained from ODD_3 using the operations α_i and β_i of Lemma 3.7. By Lemma 5.6 and Lemma 3.7, establishing this claim would finish the proof of the present lemma.

Note that by the minimality of (the arity of) R', none of the projections of R' onto a single coordinate can be one-element, and none of the projections of R' onto a pair of coordinates can be the equality relation. Thus every binary projection is either the disequality relation or $\{0,1\}^2$ (note that the binary projection cannot contain exactly 3 tuples, since R' is affine). Furthermore, if two different binary projections of R' are disequality relations then the corresponding pairs of coordinates are disjoint (since otherwise two coordinates would be always equal). Let R'' be a largest arity projection of R' such that every binary projection of R'' is $\{0,1\}^2$. Note that, to prove the lemma, it is sufficient to show that R'' is ternary and is either $\mathrm{EVEN}_3(x_1,x_2,x_3)$ or $\mathrm{ODD}_3(x_1,x_2,x_3)$. It is easy to see that then the relation R' can be obtained from ODD_3 using the operations α_i and β_i: every variable of R' not in $\{x_1,x_2,x_3\}$ forms a disequality relation with one of x_1,x_2,x_3.

Suppose that the arity of R'' is $n+1$. Consider the relations $R''_0 = R''(x_1,\ldots,x_n,0)$ and $R''_1 = R''(x_1,\ldots,x_n,1)$. By the choice of R', both of these relations are width-2 affine, that is each of them can be expressed by a system of equations of the form $x_i + x_j = a$, $a \in \{0,1\}$. Note that it is impossible that, for a pair of coordinates i,j, exactly one of the projections of R''_0 and R''_1 onto i,j is the full relation $\{0,1\}^2$. Indeed, this would imply that the size of the projection of R'' onto $\{i,j,n+1\}$ would not be a power of 2. Therefore, if, for a pair of indices i,j, one of the projections of R''_0 and R''_1 onto i,j is the equality relation then the other must be the disequality relation (to ensure that the projection of R'' onto $\{i,j\}$ is $\{0,1\}^2$). It follows that R'' can be described by the following system of equations: for each pair i,j such that $\mathrm{pr}_{i,j}(R''_0)$ is the equality relation the system contains the equation $x_i + x_j + x_{n+1} = 0$, and for each pair i,j such that $\mathrm{pr}_{i,j}(R''_0)$ is the disequality relation the system contains the equation $x_i + x_j + x_{n+1} = 1$, and there are no other equations in the system.

Note that if some x_i with $1 \le i \le n$ participates in at least two equations in this system (which also involve x_j and $x_{j'}$) then the projection of R'' onto j,j' would not be $\{0,1\}^2$, which is a contradiction. Hence, the only variable that may appear in more than one equation is x_{n+1}. Assume that the system contains at least two equations, say, $x_1 + x_2 + x_{n+1} = a$ and $x_3 + x_4 + x_{n+1} = b$. Then, by identifying x_4 with x_{n+1} in R', we would be able to obtain a relation which is affine but not width-2 (because of the equation $x_1 + x_2 + x_{n+1} = a$), and has arity smaller than the arity of R', which is a contradiction.

The lemma is proved. $\square$

It remains to consider the case when R is flip separable, but not affine. Again, we first consider one particular relation.

LEMMA 5.8. LIMP$(\{R_{\text{1-IN-3}},\neq\})$ *is* NP-*hard.*

PROOF. Let $R = \beta_1(\beta_2(R_{\text{1-IN-3}}))$. We show that LIMP$(\{R,\neq\})$ is NP-hard; by Lemma 3.7, this implies hardness for LIMP$(\{R_{\text{1-IN-3}},\neq\})$ as well. Observe that the projection of $R(x_1, x_2, x_3)$ to $\{x_1, x_2\}$ is $x_1 \vee x_2$.

The proof is by reduction from LIMP($\{x \vee y\}$). Let (ϕ_1, f_1, k_1) be an instance of LIMP($\{x \vee y\}$) with n variables and m constraints. Formula ϕ_2 contains each variable of ϕ_1 and, for each $i = 1 \dots m$, two new variables $y_i, \bar{y}_i$ corresponding to the i-th constraint of ϕ_1. If the i-th constraint of ϕ_1 is, say, $x_1 \vee x_2$, then we add the constraints $R(x_1, x_2, y_i)$ and $y_i \neq \bar{y}_i$.

There is a one-to-one mapping between the satisfying assignments of ϕ_1 and ϕ_2: a satisfying assignment of ϕ_1 can be extended to ϕ_2 in a unique way. Let f_2 be the extension of f_1 and set $k_2 := k_1 + 2m$. The total weight of the $2m$ variables y_i, $\bar{y}_i$ $(1 \leq i \leq m)$ is exactly m in every satisfying assignment of ϕ_2, thus extending a satisfying assignment of ϕ_1 to ϕ_2 increases the weight exactly by m. It follows that if f_1 has a better solution in its k_1-neighborhood, then f_2 has a better solution in its $(k_1 + 2m)$-neighborhood. Furthermore, if f_2 is suboptimal, then this is only possible if f_1 is suboptimal as well. This proves the correctness of the reduction. $\square$

LEMMA 5.9. LIMP($\{R, \neq, C_0, C_1\}$) *is* NP-*hard if* R *is flip separable but not affine.*

PROOF. It is shown in the proof of Lemma 5.30 of [Creignou et al. 2001] that there exist an instance of CSP($\{R, \neq, C_0, C_1\}$) such that if R' is the set of all solutions to this instance and $R'' = \mathrm{pr}_I(R')$ where $I = \{j \mid \mathrm{pr}_j(R') = \{0,1\}\}$ then R'' has the following property. Either R'' is a ternary relation such that

—the tuples $(0,0,0), (0,1,1), (1,0,1)$ belong to R'', and

—the tuple $(1,1,0)$ does not belong to R'',

or else R'' is obtained from such a relation by using operations α_i (see Lemma 3.7). By Lemmas 3.5 and 3.7, it suffices to show LIMP($\{R'', \neq\}$) is NP-hard when R'' is a ternary relation satisfying the two properties. Note that, since all relations in $\{R, \neq, C_0, C_1\}$ are flip separable, R'' is also a flip separable relation.

Since R'' is flip separable, it is easy to see that if one of the tuples from $R_{\text{1-IN-3}}$ belongs to R'' then all of them do (consider the tuple $(0,0,0)$ and its flip sets of size 1 and 2). However, in this case $(1,0,0) \in R''$ has flip sets $\{1\}$ and $\{1,2\}$, and therefore $\{2\}$ should also be a flip set for this tuple, which is impossible because $(1,1,0) \notin R''$. It follows that $R'' \cap R_{\text{1-IN-3}} = \emptyset$. If the tuple $(1,1,1)$ was in R'' then it would have flip sets $\{1,2,3\}$ and $\{1\}$, and hence $\{2,3\}$ as well, which is impossible. It follows that $R'' = \{(0,0,0), (0,1,1), (1,0,1)\} = \beta_1(\beta_2(R_{\text{1-IN-3}}))$. The lemma now follows from Lemmas 3.7 and 5.8. $\square$

6. CONCLUSIONS

We have completed a Schaefer-style study of Boolean CSP with respect to the complexity of decreasing the weight of a solution by doing at most k flips. We analyzed both the polynomial-time solvability and the fixed-parameter tractability of the problem. As expected, the problem is polynomial-time solvable only in trivial cases, when finding an optimum solution is easy anyway. On the other hand, we have discovered nontrivial classes of constraints for which the problem is fixed-parameter tractable, which shows that this line of investigation (parameterized complexity of local search) is worthwhile, as it can lead to positive results that are not obvious at first sight.

Following [Marx and Schlotter 2009b], we made a distinction between "strict" and "permisive" local search and obtained a classification for both variants. Investigating permissive local search leads to results that are more natural: strict local search can be hard even if the problem can be solved optimally, for example, in the case of 0-valid constraints. However, in our study, every problem that is hard for strict local search but easy for permissive local search actually turned out to be trivial to solve optimally. It would be interesting to see a problem (related to CSP or to some other problem domain) where finding the optimum is hard, strict local search is hard, but permissive local search is fixed-parameter tractable.

REFERENCES

Aarts, E. and Lenstra, J., Eds. 2003. *Local Search in Combinatorial Optimization.* Princeton University Press.

Chapdelaine, P. and Creignou, N. 2005. The complexity of Boolean constraint satisfaction local search problems. *Annals of Mathematics and Artificial Intelligence 43*, 51–63.

Cohen, D. and Jeavons, P. 2006. The complexity of constraint languages. In *Handbook of Constraint Programming*, F. Rossi, P. van Beek, and T. Walsh, Eds. Elsevier, Chapter 8.

Creignou, N., Khanna, S., and Sudan, M. 2001. *Complexity Classifications of Boolean Constraint Satisfaction Problems.* SIAM Monographs on Discrete Mathematics and Applications, vol. 7.

Crescenzi, P. and Rossi, G. 2002. On the Hamming distance of constraint satisfaction problems. *Theoretical Computer Science 288,* 1, 85–100.

Dantsin, E., Goerdt, A., Hirsch, E., Kannan, R., Kleinberg, J., Papadimitriou, C., Raghavan, P., and Schöning, U. 2002. A deterministic $(2 - \frac{2}{k+1})^n$ algorithm for k-SAT based on local search. *Theoretical Computer Science 289*, 69–83.

Downey, R. and Fellows, M. 1999. *Parameterized Complexity.* Springer.

Fellows, M. 2001. Parameterized complexity: new developments and research frontiers. In *Aspects of Complexity (Kaikura, 2000).* de Gruyter Series in Logic and Applications, vol. 4. 51–72.

Fellows, M., Fomin, F., Lokshtanov, D., Rosamond, F., Saurabh, S., and Villanger, Y. 2009. Local search: Is brute force avoidable? In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI).* 486–491.

Flüm, J. and Grohe, M. 2006. *Parameterized Complexity Theory.* Springer.

Gu, J., Purdom, P., Franko, J., and Wah, B. 2000. *Algorithms for the Satisfiability Problem.* Cambridge University Press, Cambridge, MA.

Hirsch, E. 2000. SAT local search algorithms: worst-case study. *Journal of Automated Reasoning 24*, 127–143.

Hoos, H. and Tsang, E. 2006. Local search methods. In *Handbook of Constraint Programming*, F. Rossi, P. van Beek, and T. Walsh, Eds. Elsevier, Chapter 5.

Johnson, D., Papadimitriou, C., and Yannakakis, M. 1988. How easy is local search? *Journal of Computer and Systems Sciences 37*, 79–100.

Khanna, S., Sudan, M., Trevisan, L., and Williamson, D. 2001. The approximability of constraint satisfaction problems. *SIAM Journal on Computing 30,* 6, 1863–1920.

Kirousis, L. and Kolaitis, P. 2003. The complexity of minimal satisfiability problems. *Information and Computation 187*, 20–39.

Marx, D. 2005. Parameterized complexity of constraint satisfaction problems. *Computational Complexity 14*, 153–183.

Marx, D. 2008a. Local search. Parameterized Complexity News, pages 7–8, vol. 3.

Marx, D. 2008b. Searching the k-change neighborhood for TSP is W[1]-hard. *Operations Research Letters 36*, 31–36.

Marx, D. and Schlotter, I. 2009a. Parameterized complexity and local search approaches for the stable marriage problem with ties. *Algorithmica.* To appear.

Marx, D. and Schlotter, I. 2009b. Stable assignment with couples: Parameterized complexity and local search. In *Proceedings of 4th International Workshop on Exact and Parameterized Computation (IWPEC)*. To appear.

Michiels, W., Aarts, E., and Korst, J. 2007. *Theoretical Aspects of Local Search.* EATCS Series: Monographs in Theoretical Compter Science. Springer.

Moore, C. and Robson, J. M. 2001. Hard tiling problems with simple tiles. *Discrete Comput. Geom. 26,* 4, 573–590.

Schaefer, T. 1978. The complexity of satisfiability problems. In *STOC'78*. 216–226.

Szeider, S. 2009. The parameterized complexity of k-flip local search for SAT and MAX SAT. In *Proceedings of 12th Conference on Theory and Applications of Satisfiability Testing, SAT'09*. 276–283.