Journal of Artificial Intelligence Research 52 (2015) 203-234 Submitted 8/14; published 1/15

On the Subexponential-Time Complexity of CSP

Ronald de Haan DEHAAN@KR.TUWIEN.AC.AT
Vienna University of Technology
Vienna, Austria

Iyad Kanj IKANJ@CS.DEPAUL.EDU
School of Computing, DePaul University
Chicago, USA

Stefan Szeider STEFAN@SZEIDER.NET
Vienna University of Technology
Vienna, Austria

Abstract

Not all NP-complete problems share the same practical hardness with respect to exact computation. Whereas some NP-complete problems are amenable to efficient computational methods, others are yet to show any such sign. It becomes a major challenge to develop a theoretical framework that is more fine-grained than the theory of NP-completeness, and that can explain the distinction between the exact complexities of various NP-complete problems. This distinction is highly relevant for constraint satisfaction problems under natural restrictions, where various shades of hardness can be observed in practice.

Acknowledging the NP-hardness of such problems, one has to look beyond polynomial time computation. The theory of subexponential-time complexity provides such a framework, and has been enjoying increasing popularity in complexity theory. An instance of the constraint satisfaction problem with n variables over a domain of d values can be solved by brute-force in d^n steps (omitting a polynomial factor). In this paper we study the existence of subexponential-time algorithms, that is, algorithms running in $d^{o(n)}$ steps, for various natural restrictions of the constraint satisfaction problem. We consider both the constraint satisfaction problem in which all the constraints are given extensionally as tables, and that in which all the constraints are given intensionally in the form of global constraints. We provide tight characterizations of the subexponential-time complexity of the aforementioned problems with respect to several natural structural parameters, which allows us to draw a detailed landscape of the subexponential-time complexity of the constraint satisfaction problem. Our analysis provides fundamental results indicating *whether and when* one can significantly improve on the brute-force search approach for solving the constraint satisfaction problem.

1. Introduction

It has been observed in various practical contexts that some NP-hard problems are accessible to efficient exact computational methods, whereas for others such methods are futile. It is a central challenge for theoreticians to develop a framework, that is more fine-grained than the theory of NP-completeness, and that can explain the distinction between the exact complexities of NP-hard problems. Subexponential-time complexity is a framework of complexity theory that provides such a distinction (Lokshtanov, Marx, & Saurabh, 2011). It is based on the observation that for some NP-complete problems, one can improve the exponent in the exponential term of the upper bound on their running time indefinitely—such problems admit subexponential-time algorithms—whereas for others this is apparently not possible under commonly-believed hypotheses in complexity theory. In particular, subexponential-time algorithms were developed for many graph problems, including INDEPENDENT SET and DOMINATING SET, under natural structural restrictions; e.g., see the work of Alber, Fernau, and Niedermeier (2004), Chen, Kanj, Perkovic, Sedgwick, and Xia (2007) and Demaine, Fomin, Hajiaghayi, and Thilikos (2005). The benchmark problem for subexponential-time computation is the satisfiability problem for CNF formulas, where each clause contains at most three literals, denoted 3-CNF-SAT. The *Exponential Time Hypothesis* (ETH), proposed by Impagliazzo and Paturi (2001), states that 3-CNF-SAT with n variables is not decidable in subexponential time, i.e., not decidable in time $2^{o(n)}$ (omitting polynomial factors).

The Constraint Satisfaction Problem (CSP) provides a general and uniform framework for the representation and solution of hard combinatorial problems that arise in various areas of Artificial Intelligence and Computer Science (Rossi, van Beek, & Walsh, 2006). For instance, in database theory, CSP is equivalent to the evaluation problem of conjunctive queries on relational databases (Gottlob, Leone, & Scarcello, 2002). It is well known that CSP is NP-hard, as it entails fundamental NP-hard problems such as 3-COLORABILITY and 3-CNF-SAT. Hence, we cannot hope for a polynomial-time algorithm for CSP. On the other hand, CSP can obviously be solved in *exponential time*: by simply trying all possible instantiations of the variables, we can solve a CSP instance consisting of n variables that range over a domain of d values in time d^n (omitting a polynomial factor in the input size). Significant work has been concerned with improving this trivial upper bound for various restrictions of CSP (Beigel & Eppstein, 2005; Feder & Motwani, 2002; Grandoni & Italiano, 2006; Moser & Scheder, 2011; Schöning, 1999). For instance, Razgon (2006) showed that binary CSP with domain size d can be solved in time $(d-1)^n$ by a forward-checking algorithm employing a fail-first variable ordering heuristic; although there are faster algorithms known, this result indicates that the exponential running time for CSP can be improved by using heuristic methods that were designed for solving real-world CSP instances in practice. All these improvements over the trivial brute-force search give exponential running times in which the exponent is linear in n.

The aim of this paper is to investigate the theoretical limits of such improvements. More precisely, we explore whether the exponential factor d^n can be reduced to a *subexponential factor* $d^{o(n)}$ or not, considering various natural NP-hard restrictions of classical CSP in which all the constraints are given extensionally in the form of tables, and of CSP in which the constraints are specified intensionally using *global constraints*. For CSP with global constraints, we consider CSP in which the global constraints are all either

- *AllDifferent* constraints (denoted $\text{CSP}^{\neq}$),
- *NValue* constraints (denoted $\text{CSP}^{=}$),
- *AtLeastNValue* constraints (denoted $\text{CSP}^{\geq}$),
- *AtMostNValue* constraints (denoted $\text{CSP}^{\leq}$), or
- *cTable* constraints, i.e., constraints that are specified by tables with compressed tuples (denoted CSP^{c}).

This study of CSP with global constraints is highly relevant as it is central for the modeling and the solving of real-world problems to use various global constraints that come along with efficient propagation and filtering techniques (Régin, 2011; Van Hoeve & Katriel, 2006). Therefore, the study of the existence of subexponential-time algorithms for these generic problems under various restrictions is of prime interest.

In this paper, we obtain lower and upper bounds results, and in most cases draw a detailed complexity landscape of CSP with extensionally represented constraints and CSP with global constraints with respect to subexponential-time solvability. Our lower bounds are subject to the *Exponential Time Hypothesis* (ETH), even though some of our results are derived under "weaker" complexity-theoretic hypotheses (Proposition 2, Proposition 3, and Proposition 10). The structural parameters in a CSP instance that we focus on (when relevant) are: the (instance) size, the domain size, the number of constraints, the arity (i.e., maximum size of a constraint scope), the maximum degree (i.e., the maximum number of occurrences of a variable), and the treewidth of the primal or incidence graph. We highlight below some of the results that we obtain. As it turns out, for almost all restrictions under consideration, both CSP and its generalization, CSP with the (global) compressed table constraints (CSP^c), exhibit the same behavior with respect to their subexponential-time solvability. So unless explicitly indicated in the results below, *all results* about CSP (positive and negative) mentioned below hold as well for CSP^c.

It is easy to see that CSP with bounded domain size and bounded arity has a subexponential-time algorithm if and only if the ETH fails. Our first result provides evidence that when we drop the bound on the domain size or the bound on the arity, the problem becomes "harder"; we refer to the discussion preceding Proposition 2 (n below is the number of variables in the instance):

1. If BOOLEAN CSP is solvable in nonuniform subexponential time then so is (unrestricted) CNF-SAT. For BOOLEAN CSP^c, we show that if BOOLEAN CSP^c is solvable in subexponential time then the parameterized complexity hierarchy collapses at the second level, a consequence that implies that CNF-SAT is solvable in subexponential time.

2. If 2-CSP (all constraints have arity 2) is solvable in subexponential time then CLIQUE is solvable in time $N^{o(k)}$ (N is the number of vertices and k is the clique-size).

As it turns out, the number of tuples plays an important role in characterizing the subexponential-time complexity of CSP. We show the following tight result:

3. CSP is solvable in subexponential time for instances in which the number of tuples is $o(n)$, and unless the ETH fails, is not solvable in subexponential time if the number of tuples in the instances is $\Omega(n)$.

For BOOLEAN CSP of linear size we can even derive an equivalence to the ETH:

4. BOOLEAN CSP for instances of size $\Omega(n)$ is solvable in subexponential time if and only if the ETH fails.

Results 3 and 4 also hold if we consider the total number of tuples in the constraint relations instead of the input size.

5. CSP is solvable in subexponential time for instances whose primal treewidth is $o(n)$, but is not solvable in subexponential time for instances whose primal treewidth is $\Omega(n)$ unless the ETH fails.

6. CSP is solvable in polynomial time for instances whose incidence treewidth is $O(1)$, but is not solvable in subexponential time for instances whose incidence treewidth is $\omega(1)$ unless the ETH fails.

For CSP$^{\neq}$ we show the following results:

7. CSP$^{\neq}$ is solvable in subexponential time for instances whose domain size is lower bounded by a function that is $\omega(1)$, but is not solvable in subexponential time for any constant domain size that is at least 3 unless the ETH fails.

We note that the aforementioned result may sound strange because it implies that the problem is "easier" for larger domain size. This can be explained by the fact that when the domain size gets large, the allowable upper bound on the subexponential time for solving the problem (i.e., $d(n)^{o(n)}$) gets larger as well.

8. CSP$^{\neq}$ is solvable in subexponential time for instances whose primal treewidth is $o(n)$, but is not solvable in subexponential time for instances whose primal treewidth is $\Omega(n)$ unless the ETH fails.

9. CSP$^{\neq}$ is solvable in subexponential time for instances whose incidence treewidth is $o(n)$, but is not solvable in subexponential time for instances whose primal treewidth is $\Omega(n)$ unless the ETH fails. Contrast this result with the result in (6) above.

For CSP$^{=}$, CSP$^{\geq}$, and CSP$^{\leq}$, we show the following:

10. CSP$^{\geq}$ is solvable in subexponential time for instances whose number of constraints is constant and whose domain size is lower bounded by a function that is $\omega(1)$, but is not solvable in subexponential time if the number of constraints is linear and the domain size is constant unless the ETH fails.

11. CSP$^{=}$ and CSP$^{\leq}$ are not solvable in subexponential time for instances whose domain size is constant and whose number of constraints is $\Omega(n)$ unless the ETH fails.

12. CSP$^{=}$, CSP$^{\geq}$, and CSP$^{\leq}$ are solvable in subexponential time for instances whose primal treewidth is $o(n)$, but are not solvable in subexponential time for instances whose primal treewidth is $\Omega(n)$ unless the ETH fails.

The table below provides a map that, for each of the structural parameters considered in the paper, lists the results in the paper pertaining to that structural parameter. The structural parameters that we consider for an instance of CSP, or CSP with global constraints, or both are: The size (size), the maximum size of a constraint scope (arity), the cardinality of the domain (dom), the number of tuples (tuples), the number of constraints (cons), the treewidth of the incidence graph (tw*), the treewidth of the primal graph (tw), and the maximum number of occurrences of a variable (deg).

Parameter	Results
size	Theorem 3
arity	Propositions 1, 2, 12
dom	Theorems 1, 2, 3, 7; Propositions 1, 3, 12, 14, 17; Corollaries 1, 3
tuples	Theorem 2
cons	Theorems 4, 6, 7; Propositions 14, 17; Corollaries 2, 3
tw*	Theorem 5; Propositions 16, 19
tw	Theorem 5; Propositions 15, 18
deg	Proposition 11

The results in this paper shed some light on which instances of the aforementioned variants of CSP (with and without global constraints) may be feasible with respect to exact computation. Moreover, some of the results derived in the paper provide strong theoretical evidence that some of the natural restrictions of CSP may be "harder than" k-CNF-SAT—for which a subexponential-time algorithm would lead to the failure of the ETH. Hence, our results provide a new point of view of the relationship between CNF-SAT and CSP, an important topic of recent AI research (Jeavons & Petke, 2012; Dimopoulos & Stergiou, 2006; Benhamou, Paris, & Siegel, 2012; Bennaceur, 2004).

We close this section by mentioning some further work on the subexponential-time complexity of CSP and problems in AI. Already the pioneering work on the ETH (Impagliazzo & Paturi, 2001; Impagliazzo, Paturi, & Zane, 2001) considered the k-COLORABILITY problem, which constitutes an important special case of 2-CSP over a fixed domain of size k. There are several results on 2-CSP with bounds on tw, the treewidth of the primal graph (see Section 3.3 for definitions). Lokshtanov et al. (2011) showed the following lower bound, using a result about the LIST COLORING problem (Fellows et al., 2011a): 2-CSP cannot be solved in time $f(\mathsf{tw})n^{o(\mathsf{tw})}$ unless the ETH fails. Marx (2010) showed that if there is a recursively enumerable class $\mathcal{G}$ of graphs with unbounded treewidth and a function f such that 2-CSP can be solved in time $f(G)n^{o(\mathsf{tw}/\log \mathsf{tw})}$ for instances whose primal graph is in $\mathcal{G}$, then the ETH fails. Jonsson, Lagerkvist, and Nordh (2013) investigated BOOLEAN CSP over finite constraint languages and identify the "easiest" Boolean constraint language for which CSP is still NP-hard, and show that already this problem has no subexponential-time algorithm unless the ETH fails. Traxler (2008) studied the subexponential-time complexity of CSP where the constraints are represented by listing the forbidden tuples; this is in contrast to the standard representation that we use, where the allowed tuples are given, and which naturally captures database problems (Gottlob et al., 2002; Grohe, 2006; Papadimitriou & Yannakakis, 1999). This setting can be considered as a generalization of CNF-SAT; a single clause gives rise to a constraint with exactly one forbidden tuple. If the arity is bounded by a constant, then it is insignificant whether the constraints are represented by forbidden or allowed tuples, as one can translate between these two representations in polynomial time. Finally we would like to point out some recent use of the ETH for the complexity analysis of problems that are highly relevant for AI like Planning (Bäckström & Jonsson, 2011), Probabilistic Inference (Kwisthout, Bodlaender, & van der Gaag, 2010), and Text Analysis (Ge, 2013).

Parts of this paper have been published in preliminary form in the proceedings of AAAI'13 and CP'14 (Kanj & Szeider, 2013; De Haan, Kanj, & Szeider, 2014).

2. Preliminaries

In this section we introduce the terminologies and background material needed in the paper.

2.1 Constraint Satisfiability and CNF-Satisfiability

An instance I of the CONSTRAINT SATISFACTION PROBLEM (or CSP, for short) is a triple $(V, D, \mathcal{C})$, where V is a finite set of *variables*, D is a finite set of *domain values*, and $\mathcal{C}$ is a finite set of *constraints*. Each constraint in $\mathcal{C}$ is a pair (S, R), where S, the *constraint scope*, is a non-empty sequence of distinct variables of V, and R, the *constraint relation*, is a relation over D whose arity matches the length of S; a relation is considered as a set of tuples. Therefore, the *size* of a CSP instance $I = (V, D, \mathcal{C})$ is the sum $\sum_{(S,R)\in\mathcal{C}} |S| \cdot |R|$; the *total number of tuples* is $\sum_{(S,R)\in\mathcal{C}} |R|$. We assume, without loss of generality, that every variable occurs in at least one constraint scope and every domain element occurs in at least one constraint relation. Consequently, the size of an instance I is at least as large as the number of variables in I. We write $var(C)$ for the set of variables that occur in the scope of constraint C.

An *assignment* or *instantiation* is a mapping from the set V of variables to the domain D. An assignment τ *satisfies* a constraint $C = ((x_1, \ldots, x_n), R)$ if $(\tau(x_1), \ldots, \tau(x_n)) \in R$, and τ satisfies the CSP instance if it satisfies all its constraints. An instance I is *consistent* or *satisfiable* if it is satisfied by some assignment. CSP is the problem of deciding whether a given instance of CSP is consistent. BOOLEAN CSP denotes CSP with the *Boolean domain* $\{0, 1\}$. By r-CSP we denote the restriction of CSP to instances in which the arity of each constraint is at most r.

The *primal graph* of a CSP instance I has as vertices the variables of I, and two variables are joined by an edge if and only if the variables occur together in some constraint of I. The *incidence graph* of a CSP instance I is a bipartite graph, one side of which consists of the variables in I and the other side consists of the constraints in I; a variable and a constraint are joined by an edge if the variable occurs in the constraint.

A *tree decomposition* of a graph $G = (V, E)$ is a pair (T, χ) consisting of a tree T and a mapping χ that assigns to each node t of T a subset $\chi(t) \subseteq V$ such that the following conditions are satisfied: (i) for every edge $\{u, v\} \in E$ there is a node t of T such that $u, v \in \chi(t)$; and (ii) for any three nodes t_1, t_2, t_3 of T we have $\chi(t_2) \supseteq \chi(t_1) \cap \chi(t_3)$ if t_2 lies on a path between t_1 and t_3. The *width* of (T, χ) is the size of a largest set $\chi(t)$ minus 1. The *treewidth* of G is the smallest width over all its tree decompositions. Bounding the treewidth is a classical method for restricting the structure of CSP instances. The method dates back to Freuder (1982). The treewidth parameter can be applied to CSP in terms of its primal graphs or incidence graphs giving rise to the *primal treewidth* (also called *induced width* (Dechter, 2003)) and *incidence treewidth*, respectively (Samer & Szeider, 2010), of CSP instances.

For an instance $I = (V, D, \mathcal{C})$ of CSP we define the following basic parameters.

- vars: the number $|V|$ of variables, usually denoted by n.
- size: the size of the CSP instance defined as $\sum_{(S,R)\in\mathcal{C}} |S| \cdot |R|$.
- dom: the number $|D|$ of values; that is, the union of all the values that the variables can assume.
- cons: the number $|\mathcal{C}|$ of constraints.

- arity: the maximum size of a constraint scope.
- deg: the maximum number of occurrences of a variable.
- tw: the treewidth of the primal graph of I.
- tw*: the treewidth of the incidence graph of I.

A propositional formula F over a set of variable $\{x_1, \ldots, x_n\}$ is in the *conjunctive normal form* (CNF) if it is the conjunction of a set of clauses $\{C_1, \ldots, C_m\}$, where each clause C_i, $i = 1, \ldots, m$, is the disjunction of literals (i.e., variables or negations of variables). We say that a propositional formula F is *satisfiable* if there exists a truth assignment τ to the variables in F that assigns at least one literal in each clause of F the value 1 (TRUE); we also say in this case that τ *satisfies* F. The CNF-SATISFIABILITY problem, CNF-SAT for short, is given a formula F in the CNF form, decide whether or not F is satisfiable. The *width* of a clause in a CNF formula F is the number of literals in the clause. The k-CNF-SAT problem, where $k \geq 2$, is the restriction of the CNF-SAT problem to instances in which the width of each clause is at most k. It is well known that the k-CNF-SAT problem for $k \geq 3$ is NP-complete (Garey & Johnson, 1979), whereas the 2-CNF-SAT problem is solvable in polynomial time (Papadimitriou, 1994).

2.2 Global Constraints

It is often preferred to represent a constraint more succinctly than by listing all the tuples of the constraint relation. Such an intensionally represented constraint is called a *global constraint* (Régin, 2011; Van Hoeve & Katriel, 2006). The *Global Constraints Catalogue* (Beldiceanu, Carlsson, & Rampon, 2006) lists several hundred of global constraints. In this paper we focus on the following global constraints.

- The *AllDifferent* global constraint is probably the best-known, most influential, and most studied global constraint in constraint programming (Van Hoeve & Katriel, 2006). It admits efficient matching based filtering algorithms (Régin, 1994). An *AllDifferent* constraint over a set S of variables is satisfied if each variable in S is assigned a different value.
- The global constraints *NValue* (Pachet & Roy, 1999), *AtLeastNValue* (Régin, 1995), and *AtMostNValue* (Bessiere, Hebrard, Hnich, Kiziltan, & Walsh, 2006) are widely used in constraint programming (Beldiceanu et al., 2006). Each such constraint C is associated with an integer $n_C \in \mathbb{N}$; here we consider n_C as a given integer, not as the value of a variable of the CSP instance. The *NValue* constraint C over a set S_C of variables is satisfied if the number of distinct values assigned to the variables in S_C is exactly n_C. The *AtLeastNValue* and *AtMostNValue* constraints are satisfied if the number of distinct values is $\leq n_C$ or $\geq n_C$, respectively. The special case of an *NValue* or *AtLeastNValue* constraint C where n_C equals the arity of C is equivalent to an *AllDifferent* constraint.
- The global constraint *cTable* is a *table constraint with compressed tuples*. This global constraint admits a potentially exponential reduction in the space compared to an extensional table constraint and can be propagated using a variant of the GAC-schema algorithm (Katsirelos & Walsh, 2007). *cTable* constraints have also been studied under the name *generalized DNF constraints* (Chen & Grohe, 2010). A *cTable* constraint is a pair (S, U) where $S = (v_1, \ldots, v_r)$

is a non-empty sequence of distinct variables, and U is a set of *compressed tuples*, which are sequences of the form $(V_1, \ldots, V_r)$, where $V_i \subseteq D(v_i)$, $1 \leq i \leq r$. One compressed tuple $(V_1, \ldots, V_r)$ represents all the tuples $(d_1, \ldots, d_r)$ with $d_i \in V_i$. Thus, by "decompression" one can compute from (S, U) a (unique) equivalent table constraint (S, R) where R contains all the tuples that are represented by the compressed tuples in U.

CSP where all constraints are *AllDifferent* constraints is denoted CSP$^{\neq}$. This variant of CSP was studied by Fellows, Friedrich, Hermelin, Narodytska, and Rosamond (2011b) who called it MAD-CSP (multiple all different CSP). CSP where all constraints are *NValue*, *AtLeastNValue*, or *AtMostNValue* constraints, is denoted CSP$^{=}$, CSP$^{\geq}$, and CSP$^{\leq}$, respectively. CSP where all constraints are *cTable* constraints is denoted CSP^{c}.

We note that all CSP$^{\neq}$, CSP$^{=}$, CSP$^{\geq}$, CSP$^{\leq}$, CSP^{c}, are NP-complete. In fact, CSP$^{\neq}$ (and therefore the more general CSP$^{\geq}$) is even NP-hard for instances consisting of only two constraints (Kutz, Elbassioni, Katriel, & Mahajan, 2008), and CSP$^{\leq}$ and CSP$^{=}$ are even NP-hard for instances consisting of a single constraint (Bessiere et al., 2007). CSP^{c} is clearly NP-hard as it contains classical CSP (with table constraints) as a special case. Hence all the considered problems admit the representation of NP-hard combinatorial problems.

Consider a CSP instance that models some real-world problem and uses, among others, some of the global constraints considered above, say the *AllDifferent* constraint. Then, we can combine all the *AllDifferent* constraints in the instance into a new global constraint, a multi-AllDifferent constraint. Filtering this combined constraint is polynomial time equivalent to solving one instance of CSP$^{\neq}$. Such a combination of several global constraints into a new one has been considered for several different global constraints (see, e.g., Hnich et al., 2004; Régin & Rueher, 2000).

Guarantees and limits for polynomial-time preprocessing for single *NValue*, *AtLeastNValue*, and *AtMostNValue* constraints have been given by Gaspers and Szeider (2014).

The Boolean versions of the above global constraints problems, and the parameters vars, dom, cons, arity, deg, tw, and tw^*, are defined as for CSP. The size of an instance $I = (V, D, \mathcal{C})$ of CSP$^{\neq}$ is defined as $\sum_{C \in \mathcal{C}} |S_C|$. For CSP$^{=}$, CSP$^{\geq}$, and CSP$^{\leq}$, the size of an instance $I = (V, D, \mathcal{C})$ is defined as
$\sum_{C \in \mathcal{C}} (|S_C| + \log(n_C))$. For an instance $I = (V, D, \mathcal{C})$ of CSP^{c}, the size of I is defined as
$\sum_{(S,U) \in \mathcal{C}} \sum_{(V_1, \ldots, V_r) \in U} (|V_1| + \cdots + |V_r|)$. Note that the definition of the instance size for CSP^{c} encompasses that for CSP.

2.3 Subexponential Time

A *proper* complexity function in complexity theory stands for any nondecreasing function f that is computable in $O(n + f(n))$ time and $O(f(n))$ space, where n is the length of the input (see Papadimitriou, 1994). The time complexity functions used in this paper are assumed to be proper complexity function. The $o(\cdot)$ notation used denotes the $o^{\text{eff}}(\cdot)$ notation (Flum & Grohe, 2006). More formally, for any two proper complexity functions $f, g : \mathbb{N} \to \mathbb{N}$, by writing $f(n) = o(g(n))$ we mean that there exists a proper complexity function $\mu(n) : \mathbb{N} \to \mathbb{N}$, and $n_0 \in \mathbb{N}$, such that $f(n) \leq g(n)/\mu(n)$ for all $n \geq n_0$. The $\omega(\cdot)$ notation is defined similarly to the above.

It is clear that CSP and CNF-SAT are solvable in time $\mathsf{dom}^n |I|^{O(1)}$ and $2^n |I|^{O(1)}$, respectively, where I is the input instance and n is the number of variables in I. We say that CSP (resp. CNF-SAT) is solvable in *uniform subexponential time* if there exists an algorithm that solves the problem in time $\mathsf{dom}^{o(n)} |I|^{O(1)}$ (resp. $2^{o(n)} |I|^{O(1)}$). Using the results of Chen, Kanj, and Xia (2009) and Flum and

Grohe (2006), the above definition is equivalent to the following: CSP (respectively, CNF-SAT) is solvable in *uniform subexponential time* if there exists an algorithm that for all $\varepsilon = 1/\ell$, where ℓ is a positive integer, solves the problem in time $\mathsf{dom}^{\varepsilon n}|I|^{O(1)}$ (resp. $2^{\varepsilon n}|I|^{O(1)}$). CSP (resp. CNF-SAT) is solvable in *nonuniform subexponential time* if for each $\varepsilon = 1/\ell$, where ℓ is a positive integer, there exists an algorithm A_ε that solves the problem in time $\mathsf{dom}^{\varepsilon n}|I|^{O(n)}$ (resp. $2^{\varepsilon n}|I|^{O(1)}$) (that is, the algorithm depends on ε). We note that if a problem admits a subexponential-time algorithm (uniform or nonuniform) then this means that we can improve the exponent in the exponential-term of the running time of the algorithm indefinitely.

Let Q and Q' be two problems, and let μ and μ' be two functions defined on instances of Q and Q', respectively, each assigning with an instance of the corresponding problem a parameter value. In the case of CSP and CNF-SAT, μ and μ' will assign the number of variables in the instances of these problems. A *subexponential-time Turing reduction family* (Impagliazzo, Paturi & Zane, 2001; see also Flum & Grohe, 2006) a *serf-reduction* [1] for short, is an algorithm A with an oracle to Q' such that there are computable functions $f, g : \mathbb{N} \longrightarrow \mathbb{N}$ satisfying: (1) given a pair (I, ε) where $I \in Q$ and $\varepsilon = 1/\ell$ (ℓ is a positive integer), A decides I in time $f(1/\varepsilon)\mathsf{dom}^{\varepsilon\mu(I)}|I|^{O(1)}$ (for CNF-SAT $\mathsf{dom} = 2$); and (2) for all oracle queries of the form "$I' \in Q'$" posed by A on input (I, ε), we have $\mu'(I') \leq g(1/\varepsilon)(\mu(I) + \log|I|)$.

The optimization class SNP consists of all search problems expressible by second-order existential formulas whose first-order part is universal (Papadimitriou & Yannakakis, 1991). Impagliazzo, Paturi, and Zane (2001) introduced the notion of *completeness* for the class SNP under serf-reductions, and identified a class of problems which are complete for SNP under serf-reductions, such that the subexponential-time solvability for any of these problems implies the subexponential-time solvability of all problems in SNP. Many well-known NP-hard problems are proved to be SNP-complete under the serf-reduction, including 3-SAT, VERTEX COVER, and INDEPENDENT SET, for which extensive efforts have been made in the last three decades to develop subexponential-time algorithms with no success. This fact has led to the *exponential-time hypothesis*, ETH, which is equivalent to the statement that not all SNP problems are solvable in subexponential time:

Exponential-Time Hypothesis (ETH): The problem k-CNF-SAT, for any $k \geq 3$, cannot be solved in time $2^{o(n)}$, where n is the number of variables in the input formula. Therefore, there exists $c > 0$ such that k-CNF-SAT cannot be solved in time 2^{cn}.

The following result is implied, using the standard technique of renaming variables (Impagliazzo, Paturi & Zane, 2001, Corollary 1, 2) and from the proof of the Sparsification Lemma (Impagliazzo, Paruri & Zane, 2001; Flum & Grohe, 2006, Lemma 16.17). For the sake of completeness, we provide a sketch of how these aforementioned results in the literature are combined to give the statement of the lemma.

Lemma 1. *k-CNF-SAT ($k \geq 3$) is solvable in $2^{o(n)}$ time if and only if k-CNF-SAT with a linear number of clauses and in which the number of occurrences of each variable is at most 3 is solvable in time $2^{o(n)}$, where n is the number of variables in the formula (note that the size of an instance of k-CNF-SAT is polynomial in n). In particular, choosing $k = 3$ we get: 3-CNF-SAT in which every variable occurs at most 3 times, denoted 3-3-SAT, is not solvable in $2^{o(n)}$ time unless the ETH fails.*

1. Serf-reductions were introduced by Impagliazzo, Paturi, and Zane (2001). Here we use the definition given by Flum and Grohe (2006). There is a slight difference between the two definitions, and the latter definition is more flexible for our purposes.

Proof. It was shown by Impagliazzo et al. (2001, Corollary 1, 2) that, for any $k \geq 3$, there is a serf-reduction from k-CNF-SAT to k-CNF-SAT in which the number of clauses m is linear in the number of variables n. For an instance of k-CNF-SAT in which $m = O(n)$, the total number of occurrences of the variables is also linear in n (because the width of each clause is at most k). Now for each variable that appears more than $\ell > 3$ times, using the standard technique of renaming variables, we can replace (rename) each of the ℓ occurrences with a new variable, and add a cycle of ℓ implications (using ℓ new 2-CNF-SAT clauses) enforcing that all these ℓ new variables receive the same value in any satisfying assignment. The resulting formula is a k-CNF-SAT formula in which the number of occurrences of each variable is at most 3, and in which the number of new variables is linear in the original number of variables n. This gives a serf-reduction from k-CNF-SAT (for any $k \geq 3$) to k-CNF-SAT in which the number of occurrences of each variable is at most 3 (and hence also with a linear number of clauses). $\square$

The ETH has become a standard hypothesis in complexity theory (Lokshtanov et al., 2011).

Remark 1. *In this paper, when we consider* CSP *(with or without global constraints) restricted to instances in which a certain parameter is $\Omega(g(n))$ (resp. $\omega(g(n))$, $O(g(n))$, $o(g(n))$), for some proper complexity function $g(n)$ of the number of variables n in the instance, we mean* CSP *restricted to all the instances in which the parameter is* upper bounded *by a prespecified function that is $\Omega(g(n))$ (resp. $\omega(g(n))$, $O(g(n))$, $o(g(n))$). For example, when we say "*CSP *restricted to instances whose primal treewidth is $o(n)$ is solvable in subexponential time" we mean the following: For any proper complexity function $g(n) = o(n)$, the problem consisting of the restriction of* CSP *to instances whose primal treewidth is at most $g(n)$ is solvable in subexponential time.*

3. CSP and CSP^c

In this section we investigate the subexponential-time complexity of CSP and CSP^c with respect to restrictions on various structural parameters. We start in Subsection 3.1 by establishing relations among the subexponential-time complexity of CNF-SAT, CSP, and CSP^c; some of these results will be the corner stones that the results in the subsequent (sub)sections rely upon.

3.1 Relations Among CSP, CSP^c, and CNF-SAT

We start with the following simple observation:

Observation 1. *For any positive integer constant r, there is a serf-reduction from r-*CSP *to r-*CSP^c *and vice versa. Moreover, each of the reductions produces an instance having the same set of variables and the domain values as those of the original instance.*

The fact that there is a serf-reduction from r-CSP to r-CSP^c trivially follows from the fact that r-CSP is a special case of r-CSP^c. For the opposite direction, observe that each *cTable* constraint of bounded arity can be decompressed to a table constraint, over the same set of variables, in polynomial time by enumerating all tuples that satisfy the *cTable* constraint. This is a polynomial time serf-reduction from r-CSP^c to r-CSP.

Proposition 1. BOOLEAN r-CSP*, where $r \geq 3$, is solvable in subexponential time if and only if the ETH fails.*

Proof. To prove the first part of the statement, we give a serf-reduction from the r-CNF-SAT to BOOLEAN r-CSP. Given an instance F of r-CNF-SAT, it is easy to see that we can correspond with every clause in F a constraint with arity at most r (over the same variables) containing at most 2^r tuples such that the clause is satisfied if and only if the corresponding constraint is. Clearly, this is a polynomial-time reduction that results in an instance of BOOLEAN r-CSP with the same variable-set as F, and hence is a serf-reduction.

To prove the converse, we give a serf-reduction from BOOLEAN r-CSP to r-CNF-SAT. Let I be an instance of BOOLEAN r-CSP. We construct an instance F of r-CNF-SAT as follows. Let C be a constraint in I. Since the arity of I is at most r, C contains at most r variables and 2^r tuples. We can associate with C a set of clauses in F, each of width at most r, such that C is satisfied if and only if all the associated clauses are. This can be easily done by considering each tuple over the variable-set of C that is not contained in C, and adding a clause to F consisting of the disjunction of the negation of the set of literals that the tuple represents. Since each tuple represents a conjunction of a set of at most r literals, this results in at most 2^r clauses, with each being the disjunction of at most r literals. Clearly, F is an instance of r-CNF-SAT over the same variable-set as I, and F is computable in polynomial time. The proof follows. □

The following proposition suggests that Proposition 1 may not extend to r-CSP with unbounded domain size. Chen, Chor, Fellows, Huang, Juedes, Kanj, and Xia (2005) showed that if CLIQUE (decide whether a given a graph on N vertices contains a complete subgraph of k vertices) is solvable in time $N^{o(k)}$ then the ETH fails. The converse, however, is generally believed not to be true. The idea behind the proof of the following proposition goes back to the paper by Papadimitriou and Yannakakis (1999), where they used it in the context of studying the complexity of database queries. We provide the proof for completeness.

Proposition 2. *If* 2-CSP *is solvable in subexponential time then* CLIQUE *is solvable in time* $N^{o(k)}$.

Proof. Assume that 2-CSP is solvable in time $\mathsf{dom}^{o(n)}$, and let (G, k) be an instance of CLIQUE, where G has N vertices. Assume that the vertices in G are labeled $\{1, \ldots, N\}$. We construct an instance I of 2-CSP as follows. The variable-set of I is $\{x_1, \ldots, x_k\}$, and the variables range over the domain $\{1, \ldots, N\}$; that is, the variables will be used to select the vertices of G that form the clique (if it exists). For every pair of distinct variables x_i, x_j, where $i < j$, we add a constraint C_{ij} containing all pairs/tuples of the form (u, v) such that uv is an edge in G and $u < v$.

It is not difficult to verify that G has a clique of k vertices if and only if I is consistent. Since I has k variables and $\mathsf{dom} = N$, it follows that I, and hence (G, k), can be decided in time $N^{o(k)}$. The proof follows. □

By Observation 1, the statement of Proposition 1 holds true for BOOLEAN r-CSP^c, and the statement of Proposition 2 holds true for 2-CSP^c.

We explore next the relation between BOOLEAN CSP with unbounded arity and CNF-SAT. We show that if BOOLEAN CSP is solvable in nonuniform subexponential time then so is CNF-SAT. To do so, we exhibit a nonuniform subexponential-time Turing reduction from CNF-SAT to BOOLEAN CSP.

Intuitively, one would try to reduce an instance F of CNF-SAT to an instance I of CSP by associating with every clause in F a constraint in I whose variables are the variables in the clause, and whose relation consists of all tuples that satisfy the clause. There is a slight complication in such an attempted reduction because the number of tuples in a constraint could be exponential if the number of variables in the corresponding clause is linear (in the total number of variables). To overcome this subtlety, the idea is to first apply a subexponential-time (Turing) reduction, which is originally due to Schuler (2005) and was also used and analyzed by Calabro, Impagliazzo, and Paturi (2006), that reduces the instance F to subexponentially many (in n) instances in which the width of each clause is at most some constant k; in our case, however, we will reduce the width to a suitable nonconstant value. We follow this reduction with the reduction to BOOLEAN CSP described in the proof of Proposition 1.

Theorem 1. *If* BOOLEAN CSP *has a nonuniform subexponential-time algorithm then so does* CNF-SAT.

Proof. Suppose that BOOLEAN CSP is solvable in nonuniform subexponential time. Then for every $\delta > 0$, there exists an algorithm A'_δ that, given an instance I of BOOLEAN CSP with n' variables, A'_δ solves I in time $2^{\delta n'}|I|^{c'}$, for some constant $c' > 0$.

Let $0 < \varepsilon < 1$ be given. We describe an algorithm A_ε that solves CNF-SAT in time $2^{\varepsilon n}m^{O(1)}$. Set $k = \lfloor \frac{\varepsilon n}{2(1+c')} \rfloor$. Let F be an instance of CNF-SAT with n variables and m clauses. The algorithm A_ε is a search-tree algorithm, and works as follows. The algorithm picks a clause C in F of width more than k; if no such clause exists the algorithm stops. Let $l_1, \dots, l_k$ be any k literals in C. The algorithm branches on C into two branches. The first branch, referred to as a *left branch*, corresponds to one of these k literals being assigned the value 1 in the satisfying assignment sought, and in this case C is replaced in F by the clause $(l_1 \vee \ldots \vee l_k)$, thus reducing the number of clauses in F of width more than k by 1. The second branch, referred to as a *right branch*, corresponds to assigning all those k literals the value 0 in the satisfying assignment sought; in this case the values of the variables corresponding to those literals have been determined, and the variables can be removed from F and F gets updated accordingly. Therefore, in a right branch the number of variables in F is reduced by k. The execution of the part of the algorithm described so far can be depicted by a binary search tree whose leaves correspond to instances resulting from F at the end of the branching, and in which each clause has width at most k. The running time of this part of the algorithm is proportional to the number of leaves in the search tree, or equivalently, the number of root-leaf paths in the search tree.

Before we continue the description of the algorithm A_ε, we illustrate the above branching phase of the algorithm with the following concrete example. Suppose that F is an instance of CNF-SAT over the 6 variables $\{x_1, \dots, x_6\}$ consisting of the 3 clauses C_1, C_2, C_3, where $C_1 = \{x_1, x_2, \overline{x_3}, x_4, x_5\}$, $C_2 = \{\overline{x_2}, x_3, x_5, x_6\}$, and $C_3 = \{x_1, x_3, x_4, \overline{x_5}, \overline{x_6}\}$. Suppose that we want to reduce the formula-width to 3 (i.e., $k = 3$). We pick any clause of width more than 3, say C_1, and branch on any 3 literals in C_1, say $x_1, x_2, \overline{x_3}$. In the left branch (at least one of these 3 literals is 1) we obtain the (CNF) formula F_1 consisting of the 3 clauses $\{x_1, x_2, \overline{x_3}\}$, C_2, and C_3; in the right branch (each of these literals is assigned 0), we obtain the formula F_2 consisting of the clause $\{x_4, x_5\}$ (C_2 and C_3 are satisfied in this case). Note that we do not branch anymore on F_2 since its width is 2. Since F_1 still contains clauses of width more than 3, namely C_2 and C_3, we branch further on F_2 by picking a clause of width more than 3, say C_3, and branching on 3 literals in C_3, say x_1, x_3, x_4. In the left branch, we obtain the formula $F_{1,1}$ consisting of the 3 clauses $\{x_1, x_2, \overline{x_3}\}$, C_2, $\{x_1, x_3, x_4\}$; in the right branch we obtain the formula $F_{1,2}$ consisting of the 2 clauses $\{\overline{x_2}, x_5, x_6\}$ and $\{\overline{x_5}, \overline{x_6}\}$. We do not branch on $F_{1,2}$ since its width is 3. Since $F_{1,1}$ contains the clause C_2 of width more than 3, we branch on 3 literals in C_2, say $\overline{x_2}, x_3, x_5$. In the left branch we obtain the formula $F_{1,1,1}$ consisting of the 3 clauses $\{x_1, x_2, \overline{x_3}\}$, $\{\overline{x_2}, x_3, x_5\}$, and $\{x_1, x_3, x_4\}$; we do not branch on $F_{1,1,1}$ since its

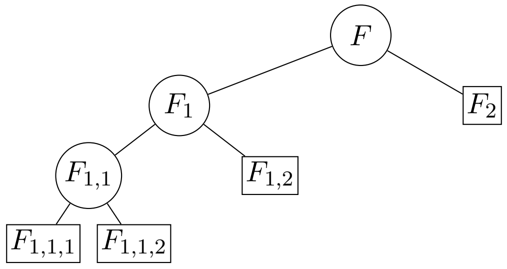

Figure 1: The search tree corresponding to the branching of the algorithm in the example.

width is 3. In the right branch we obtain the formula $F_{1,1,2}$ consisting of the two clauses $\{x_1, x_4\}$ and $\{x_6\}$; we do not branch on $F_{1,1,2}$. The algorithm does not branch anymore since all the leaves in the search tree are formulas of width at most 3. Figure 1 depicts the search tree corresponding to the branching in the above example.

We now continue the description of the algorithm A_ε. Let F' be an instance resulting from F at a leaf of the search tree. We reduce F' to an instance $I_{F'}$ of BOOLEAN CSP as follows. For each clause C' in F', we correspond to it a constraint whose variable-set is the set of variables in C', and whose tuples consist of at most $2^k - 1$ tuples corresponding to all assignments to the variables in C' that satisfy C'. Clearly, $I_{F'}$ can be constructed in time $2^k m^{O(1)}$ (note that the number of clauses in F' is at most m). To the instance $I_{F'}$, we apply the algorithm A'_δ with $\delta = \varepsilon/2$. The algorithm A_ε accepts F if and only if A'_δ accepts one of the instances $I_{F'}$, for some F' resulting from F at a leaf of the search tree.

To illustrate this phase of the algorithm using the example above, for each of the formulas F_2, $F_{1,2}$, $F_{1,1,1}$, and $F_{1,1,2}$, corresponding to the leaves of the search tree (see Figure 1), we associate an instance of BOOLEAN CSP. For example, the instance of BOOLEAN CSP associated with $F_{1,2}$ consists of two constraints. The first constraint corresponds to clause $\{\overline{x_2}, x_5, x_6\}$; it has (x_2, x_5, x_6) as its sequence of variables, and
$\{(0,0,0),(0,0,1),(0,1,0),(0,1,1),(1,1,0),(1,1,1),(1,0,1)\}$ as its relation. The second constraint corresponds to clause $\{\overline{x_5}, \overline{x_6}\}$ in $F_{1,2}$; it has (x_5, x_6) as its sequence of variables, and $\{(0,0),(0,1),(1,0)\}$ as its relation.

The running time of A_ε is upper bounded by the number of leaves in the search tree, multiplied by a polynomial in the length of F (polynomial in m) corresponding to the (maximum) total running time along a root-leaf path in the search tree, multiplied by the time to construct the instance $I_{F'}$ corresponding to F' at a leaf of the tree, and multiplied by the running time of the algorithm A'_δ applied to $I_{F'}$. Note that the binary search tree depicting the execution of the algorithm is not a complete binary tree. To upper bound the size of the search tree, let P be a root-leaf path in the search tree, and let ℓ be the number of right branches along P. Since each right branch removes k variables, $\ell \leq n/k$ and the number of variables left in the instance F' at the leaf endpoint of P is $n - \ell k$. Noting that the length of a path with ℓ right branches is at most $m + \ell$ (each left branch reduces m by 1 and hence there can be at most m such branches on P, and there are ℓ right branches), we conclude that the number of root-leaf paths, and hence the number of leaves, in the search tree is at most $\sum_{\ell=0}^{\lceil n/k \rceil} \binom{m+\ell}{\ell}$.

The reduction from F' to an instance of BOOLEAN CSP can be carried out in time $2^k m^{O(1)}$, and results in an instance $I_{F'}$ in which the number of variables is at most $n' = n - \ell k$, the number of constraints is at most m, and the total size is at most $2^k m^{O(1)}$. Summing over all possible paths in the search tree, the running time of A_ε is $2^{\varepsilon n} m^{O(1)}$. This is a consequence of the following estimation:

$$
\begin{aligned}
\sum_{\ell=0}^{\lceil n/k \rceil} \binom{m+\ell}{\ell} 2^k m^{O(1)} \cdot 2^{\delta(n-\ell k)} . (2^k m^{O(1)})^{c'} &\leq 2^{(1+c')k+\delta n} m^{O(1)} \sum_{\ell=0}^{\lceil n/k \rceil} \binom{m+\lceil n/k \rceil}{\ell} \\
&\leq 2^{(1+c')k+\delta n} m^{O(1)} \binom{2m}{\lceil n/k \rceil} \qquad (1) \\
&\leq 2^{(1+c')k+\delta n} m^{O(1)} \cdot (2m)^{n/k} \qquad (2) \\
&\leq 2^{(1+c')k+\delta n} m^{O(1)} \qquad (3) \\
&\leq 2^{\varepsilon n} m^{O(1)}.
\end{aligned}
$$

The first inequality follows after replacing ℓ by the larger value $\lceil n/k \rceil$ in the upper part of the binomial coefficient, and upper bounding the term $2^{-\ell \delta k}$ by 1. Inequality (1) follows from the fact that the largest binomial coefficient in the summation is $\binom{m+\lceil n/k \rceil}{\lceil n/k \rceil} \leq \binom{2m}{\lceil n/k \rceil}$ ($m \geq \lceil n/k \rceil$, otherwise m is a constant, and the instance of CNF-SAT can be solved in polynomial time from the beginning), and hence, the summation can be replaced by the largest binomial coefficient multiplied by the number of terms ($\lceil n/k \rceil$+1) in the summation, which gets absorbed by the term $m^{O(1)}$. Inequality (2) follows from the trivial upper bound on the binomial coefficient (the ceiling can be removed because polynomials in m get absorbed). Inequality (3) follows after noting that n/k is a constant (depends on ε), and after substituting k and δ by their values/bounds.

It follows that the algorithm A_ε solves CNF-SAT in time $2^{\varepsilon n} m^{O(1)}$. Therefore, if BOOLEAN CSP has a nonuniform subexponential-time algorithm, then so does CNF-SAT. The algorithm is nonuniform because the polynomial factor in the running time (exponent of m) depends on ε. □

Theorem 1 provides strong evidence that BOOLEAN CSP is not solvable in subexponential time. We show next that BOOLEAN CSP^c is not solvable in subexponential time under a weaker hypothesis than that assumed in Theorem 1. By SAT[3] we denote the satisfiability of normalized propositional formulas of depth 3 (Flum & Grohe, 2006), that is, propositional formulas that are the conjunction-of-disjunction-of-conjunction of literals. It is well known that if SAT[3] is solvable in subexponential time then the W-hierarchy in parameterized complexity collapses at the second level (Chen et al., 2006), that is, $W[2] =$ FPT, which is a consequence that is deemed very unlikely and would imply that CNF-SAT is solvable in subexponential time (Chen et al., 2006).

Proposition 3. *Unless* $W[2] =$ FPT, BOOLEAN CSP^c *is not solvable in subexponential time.*

Proof. It is easy to see that an instance of SAT[3] is polynomial-time reducible to an instance of BOOLEAN CSP^c on the same set of variables. In this reduction, every disjunction-of-conjunction of literals in the Boolean formula is associated with a *cTable* constraint, where each compressed tuple $(V_1, \ldots, V_r)$ of this constraint represents a conjunction of literals: a positive literal x_i is represented by $V_i = \{1\}$, a negative literal $\neg x_i$ is represented by $V_i = \{0\}$, and if a variable x_i does not occur in the conjunction, it is represented by $V_i = \{0, 1\}$. Therefore, there is a serf-reduction from SAT[3] to BOOLEAN CSP^c. The statement now follows from the result by Chen et al. (2006). □

3.2 Instance Size and Number of Tuples

In this section we give characterizations of the subexponential-time complexity of CSP and CSP^c with respect to the instance size and the number of tuples. We also show that the subexponential-time solvability of BOOLEAN CSP and BOOLEAN CSP^c with linear size, or linear number of tuples, is equivalent to the statement that the ETH fails.

Proposition 4. *Unless the ETH fails, the restriction of* BOOLEAN CSP *to instances whose size is* $\Omega(n)$ *is not solvable in subexponential time.*

Proof. Let $s(n) = \Omega(n) \geq cn$ be a proper complexity function, where $c > 0$ is a constant. Suppose that the restriction of BOOLEAN CSP to instances of size at most $s(n)$ is solvable in subexponential time, and we will show that 3-CNF-SAT is solvable in subexponential time. By Lemma 1, it is sufficient to show that 3-CNF-SAT with a linear number of clauses is solvable in $2^{o(n)}$ time. Using a padding argument[2], we can prove the preceding statement assuming any linear upper bound on the number of clauses; this is true because we can pad any instance of 3-CNF-SAT with a large number of new variables to obtain an equivalent instance in which the number of clauses satisfies the (smaller) desired upper bound. We pick this linear upper bound to be $cn/24$, where c is the constant in the upper bound on $s(n)$.

Let F be an instance of 3-CNF-SAT with n variables and at most $cn/24$ clauses. We reduce F to an instance I_F of BOOLEAN CSP using the same reduction described in the proof of Theorem 1: for each clause C of F we correspond a constraint whose variables are those in C and whose tuples are those corresponding to the satisfying assignments to C. Since the width of C is 3 and the number of clauses is at most $cn/24$, the instance I_F consists of at most $cn/24$ constraints, each containing at most 3 variables and 8 tuples. Therefore, the size of I_F is at most cn. We now apply the hypothetical subexponential-time algorithm to I_F. Since $|I|$ is linear in n, and since the reduction takes linear time in n, we conclude that 3-CNF-SAT is solvable in time $2^{o(n)}n^{O(1)} = 2^{o(n)}$. The proof follows. $\square$

Since BOOLEAN CSP is a special case of BOOLEAN CSP^c, the statement of Proposition 4 holds true for BOOLEAN CSP^c as well.

Proposition 5. *The restriction of* CSP^c *to instances with* $o(n)$ *tuples is solvable in subexponential time.*

Proof. Let $s(n) = o(n)$ be a proper complexity function, and consider the restriction of CSP^c to instances with at most $s(n)$ tuples. We will show that this problem is solvable in time $\mathsf{dom}^{s(n)}|I|^{O(1)}$. Let I be an instance of the problem under consideration. Consider the algorithm A that, for each compressed tuple in a constraint in I, branches on whether or not the compressed tuple is satisfied by the satisfying assignment sought. A branch in which more than one compressed tuple in any constraint is selected as satisfied is rejected, and likewise for a branch in which no compressed tuple in a constraint is selected. For each remaining branch, the algorithm checks if the branch is consistent, which would imply that there is an assignment to the variables that aligns with the branch and satisfies I. Checking if a branch is consistent is done as follows. Let x be a variable in I, and let $t_1, \ldots, t_p$ be the compressed tuples selected by the branch in the *cTables* that contain x as a variable.

2. A padding argument is a general tool that is used in complexity theory to extend a result to a larger class of problems. For our purpose in this paper, the padding argument works by adding/padding a "dummy" part to the instance to create an equivalent new instance in which a relation holds true between certain parameters in the new instance. We will use the padding argument several times in this paper, and skip the details once the argument is clear.

Let V_i^x, $i = 1, \ldots, p$, be the set of values admissible for x in the *cTable* from which t_i was selected by the branch. A branch is consistent with respect to x if $\bigcap_{i=1}^{p} V_i^x \neq \emptyset$, and a branch is consistent if it is consistent with respect to every variable in I. Clearly, for a given branch by the algorithm A, checking whether or not the branch is consistent can be done in polynomial time in $|I|$.

If a branch is consistent, the algorithm accepts; the algorithm rejects if no branch corresponds to a consistent assignment. Clearly, the algorithm A is correct, and runs in time $2^{s(n)}|I|^{O(1)} = \mathsf{dom}^{s(n)}|I|^{O(1)}$ (we assume that $\mathsf{dom} \geq 2$, otherwise the problem is trivial). $\square$

Noting that the number of tuples is a lower bound for the instance size, the following proposition follows from Proposition 4 and Proposition 5:

Proposition 6. *The restriction of* CSP *to instances in which the number of tuples is $o(n)$ is solvable in subexponential time, and unless the ETH fails, the restriction of* CSP *to instances in which the number of tuples is $\Omega(n)$ is not solvable in subexponential time. The same holds true for* CSP^c.

Next, we show that the subexponential-time solvability of BOOLEAN CSP with linear size, or with linear number of tuples, is equivalent to the statement that the ETH fails. We first need the following proposition:

Proposition 7. *If the ETH fails then the restriction of* BOOLEAN CSP^c *to instances with linear number of tuples is solvable in subexponential time.*

Proof. We give a polynomial-time serf-reduction from BOOLEAN CSP^c with linear number of tuples to CIRCUIT SATISFIABILITY with linear size circuits. The result will then follow from the fact that CIRCUIT SATISFIABILITY with linear size circuits is SNP-complete under serf-reductions (and hence is solvable in subexponential time if and only if the ETH fails) (Impagliazzo, Paturi & Zane, 2001). Let $s(n) \leq cn$ be a proper complexity function, where $c > 0$ is a constant. Consider the restriction of BOOLEAN CSP^c to instances in which the number of tuples is at most cn, and let I be an instance of this problem. We construct a Boolean circuit C_I as follows. The circuit C_I is a depth-3 circuit whose output gate is an AND-gate, and whose set of variables is the same as that of I. With each *cTable* constraint T in I we correspond an OR-gate g_T that is connected to the output gate of C. Let T be a *cTable* constraint in I over the Boolean variables $(v_1, \ldots, v_r)$, and let $t = (V_1, \ldots, V_r)$ be a compressed tuple in T. We correspond to t an AND-gate g_t in C_I that is connected to the OR-gate g_T corresponding to T in C_I; the input to g_t are literals in C_I that are determined as follows. For each v_i, $i = 1, \ldots, r$, if $V_i = \{1\}$ then connect the variable corresponding to v_i in C_I to g_t, and if $V_i = \{0\}$ then connect the negation of the variable corresponding to v_i in C_I to g_t (we do nothing if $V_i = \{0, 1\}$ because there is no constraint imposed by the tuple on the Boolean value of v_i). It is easy to see that t is satisfied if and only if the corresponding gate g_t in C_I evaluates to 1, and hence T is satisfied if and only if g_T evaluates to 1. It follows that I is satisfied if and only if C_I is. Moreover, the size of C_I is linear in the number of tuples in I, and subsequently in the number of variables in C_I. Since the construction of C_I can be done in polynomial time, the proof follows. $\square$

Clearly, the statement of the above proposition holds true for BOOLEAN CSP as well.

Proposition 4, combined with Proposition 7 after noting that the size is an upper bound on the number of tuples, gives the following results:

Theorem 2. *The restriction of* BOOLEAN CSP *to instances with linear number of tuples is solvable in subexponential time if and only if the ETH fails. The same result holds for* BOOLEAN CSP^c.

Theorem 3. *The restriction of* BOOLEAN CSP *to instances with linear size is solvable in subexponential time if and only if the ETH fails. The same result holds for* BOOLEAN CSP^c.

3.3 Number of Constraints and Treewidth

In this section we give characterizations of the subexponential-time complexity of CSP and CSP^c with respect to the number of constraints, and the treewidth of the primal and incidence graphs. We start withe following proposition:

Proposition 8. *Unless the ETH fails, the restriction of* CSP *to instances in which the number of constraints is $\omega(1)$ is not solvable in subexponential time.*

Proof. Let $\lambda(n) = \omega(1)$ be a proper complexity function. We show that, unless the ETH fails, the restriction of CSP to instances in which $\mathsf{cons} \leq \lambda(n)$, denoted CSP$_\lambda$ is not solvable in $\mathsf{dom}^{o(n)}$ time. By Proposition 1, it suffices to provide a serf-reduction from BOOLEAN 3-CSP with a linear number of constraints to BOOLEAN CSP$_\lambda$.

Let I be an instance of BOOLEAN CSP in which $\mathsf{cons} = n' \leq cn$, where $c > 0$ is a constant. Let $C_1, \ldots, C_{n'}$ be the constraints in I; we partition these constraints arbitrarily into $\lfloor \lambda(n) \rfloor$ many groups $\mathcal{C}_1, \ldots, \mathcal{C}_r$, where $r \leq \lfloor \lambda(n) \rfloor$, each containing at most $\lceil n'/\lambda(n) \rceil$ constraints. The serf-reduction A works as follows. A "merges" all the constraints in each group $\mathcal{C}_i$, $i = 1, \ldots, r$, into one constraint C'_i as follows. The variable-set of C'_i consists of the union of the variable-sets of the constraints in $\mathcal{C}_i$. For each constraint C in $\mathcal{C}_i$, iterate over all tuples in C. After selecting a tuple from each constraint in $\mathcal{C}_i$, check if all the selected tuples are consistent, and if so merge all these tuples into a single tuple and add it to C'_i. By merging the tuples we mean form a single tuple over the variables in these tuples, and in which the value of each variable is its value in the selected tuples (note that the values are consistent). Since each constraint in I has arity at most 3, and hence contains at most 8 tuples, and since each group contains at most $\lceil n'/\lambda(n) \rceil$ constraints, C'_i can be constructed in time $8^{\lceil n'/\lambda(n) \rceil} n'^{O(1)} = 2^{o(n)}$, and hence, all the constraints $C'_1, \ldots, C'_r$ can be constructed in time $2^{o(n)} n^{O(1)} = 2^{o(n)}$. We now form the instance I' whose variable-set is that of I, and whose constraints are $C'_1, \ldots, C'_r$. Since $r \leq \lfloor \lambda(n) \rfloor$, I' is an instance of CSP$_\lambda$. Moreover, it is easy to see that I is consistent if and only if I' is. Since I' can be constructed from I in subexponential time and the number of variables in I' is at most that of I, it follows that A is a serf-reduction from BOOLEAN 3-CSP with a linear number of constraints to CSP$_\lambda$. □

Proposition 9. *The restriction of* CSP^c *to instances in which* $\mathsf{cons} = O(1)$ *is solvable in polynomial time.*

Proof. If the number of constraints in an instance is $O(1)$, then in polynomial time we can enumerate each subset of tuples such that the subset contains exactly one compressed tuple from each constraint in the instance (because the size of such a subset is $O(1)$). We can then verify consistency (as described in the proof of Proposition 5), and deduce an instantiation of the set of variables if it exists in polynomial time. □

Clearly, Proposition 8 holds true for CSP^c, and Proposition 9 holds true for CSP. Therefore, combining Proposition 8 and Proposition 9 we have:

Theorem 4. *The restriction of* CSP *to instances with $O(1)$ constraints is solvable in polynomial time, and unless the ETH fails, the restriction of* CSP *to instances with $\omega(1)$ constraints is not solvable in subexponential time. The same holds true for* CSP^c.

When now turn our attention to treewidth. We have the following proposition:

Proposition 10. *Unless* CSP *(in general) is solvable in subexponential time (and hence the ETH fails), the restriction of* CSP *to instances whose* tw *is* $\Omega(n)$ *is not solvable in subexponential time.*

Proof. Let $s(n) = cn$, where $c > 0$ is a constant, and consider the restriction of CSP to instances whose tw is at most $s(n)$, denoted LINEAR-tw-CSP. Note that the number of vertices in the primal graph is n, and hence $\mathsf{tw} \le n$. Therefore, if $c \ge 1$, then the statement trivially follows. Suppose now that $c < 1$, and let I be an instance of CSP with n variables. By "padding" $\lceil 1/c \rceil$ disjoint copies of I we obtain an instance I' that is equivalent to I, whose number of variables is $N' = \lceil 1/c \rceil n$, and whose tw is the same as that of I. Since the tw of I is at most n, it follows that the tw of I' is at most cN', and hence I' is an instance of LINEAR-tw-CSP. This gives a serf-reduction from CSP to LINEAR-tw-CSP. □

We note that the hypothesis "CSP is solvable in subexponential time" in the above theorem implies that the "ETH fails" by Proposition 1, and implies that CNF-SAT has a nonuniform subexponential-time algorithm by Theorem 1.

The following theorem provides a tight characterization of the subexponential-time complexity of CSP^c (and CSP) with respect to the primal and incidence treewidth.

Theorem 5. *The following statements are true:*

(i) The restriction of CSP^c *to instances in which* $\mathsf{tw} = o(n)$ *is solvable in subexponential time, and unless the ETH fails, the restriction of* CSP^c *to instances in which* $\mathsf{tw} = \Omega(n)$ *is not solvable in subexponential time.*

(ii) The restriction of CSP^c *to instances in which* $\mathsf{tw}^* = O(1)$ *is solvable in subexponential time (even in* P*), and unless the ETH fails, the restriction of* CSP^c *to instances in which* $\mathsf{tw}^* = \omega(1)$ *is not solvable in subexponential time.*

Proof. (i) Note that an upper bound on the primal treewidth implies the same upper bound on the arity. Let $\mathcal{I}$ be an instance of CSP^c whose $\mathsf{tw} = o(n)$. Since $\mathsf{arity} = o(n)$, each constraint contains at most $d(n)^{o(n)}$ many satisfying tuples. By decompressing compressed tuples, i.e., by enumerating all the satisfying tuples in each constraint in time $O^*(d(n)^{o(n)})$ we can reduce the instance $\mathcal{I}$ to an instance of CSP on the same set of variables, domain, and primal tree width. Now we can compute a tree decomposition of width at most $4 \cdot \mathsf{tw}$ in time $2^{4.38\mathsf{tw}}|I|^{O(1)}$ (Amir, 2010). It is well known (Freuder, 1990) that CSP is solvable in time $O^*(d(n)^{\mathsf{tw}}) \subseteq O^*(d(n)^{o(n)})$, and hence $\mathcal{I}$ can be decided in subexponential time. The hardness result follows from the same hardness result for CSP in Proposition 10.

(ii) The hardness result is a direct consequence of the hardness result in Theorem 4, since cons is an upper bound on tw^*. Establishing the first statement requires some work. Consider an instance $\mathcal{I}$ of CSP^c whose incidence treewidth is a constant w.

We apply a construction of Samer and Szeider (2010) to transform $\mathcal{I}$ into an equivalent instance $\mathcal{I}'$ of CSP^c whose incidence treewidth is at most $w + 1$ and where each variable appears in the scope of at most 3 constraints. The construction keeps all constraints of $\mathcal{I}$ and adds binary equality constraints and copies of variables. The equality constraints enforce that a variable and all its copies get assigned the same value. The construction of Samer and Szeider is stated for table constraints but clearly works also for *cTable*, since the constraints of $\mathcal{I}$ are not changed at all, and the newly introduced constraints are binary.

Consider the *dual graph* G^d of $\mathcal{I}'$ which has as vertices the constraints of $\mathcal{I}'$, and where two constraints are joined by an edge if and only if they share at least one variable. Because each variable appears in the scope of at most 3 constraints, a further result of Samer and Szeider (2010, Lemma 2(5)) applies, which is based on a construction due to Kolaitis and Vardi (2000), and from which it follows that the treewidth of G^d is at most $2w + 2$.

Next we obtain the the CSP instance $\mathcal{I}''$ which is "dual" to the instance $\mathcal{I}'$. This construction is a straightforward generalization of a known construction for CSP with table constraints (see, e.g., Dechter, 2003, Definition 2.1). Each constraint $C = (S, U)$ of $\mathcal{I}'$ gives rise to a variable $x[C]$ of $\mathcal{I}''$; the domain $D(x[C])$ is U, a set of compressed tuples. Between any two variables $x[C_1], x[C_2]$ of $\mathcal{I}''$ corresponding to constraints $C_1 = (S_1, U_1)$ and $C_2 = (S_2, U_2)$, respectively, of $\mathcal{I}'$ that share at least one variable we add a binary table constraint $((x[C_1], x[C_2]), R)$. Here, the relation R contains all pairs $(t_1, t_2) \in U_1 \times U_2$ that are consistent in the sense that for all variables x that appear in the scopes of C_1 and C_2, the coordinate V_i^1 of t_1 corresponding to x and the coordinate V_j^2 of t_2 corresponding x have a nonempty intersection. It is straightforward to see that $\mathcal{I}'$ and $\mathcal{I}''$ are equivalent. It remains to observe that G^d is isomorphic to the primal graph of $\mathcal{I}''$, and hence the primal treewidth of $\mathcal{I}''$ is $2w + 2$, a constant. Hence we can solve $\mathcal{I}''$ in polynomial time (Freuder, 1990). ☐

Clearly the same results in Theorem 5 hold true for CSP since the positive results in the theorem were shown for the more general CSP^c, and the negative results were proved for CSP.

We note the difference between the subexponential-time complexity of CSP^c (and CSP) with respect to the two structural parameters tw and tw*: Whereas the threshold function for the subexponential-time solvability of CSP^c and CSP with respect to tw is $o(n)$, the threshold function with respect to tw* is $O(1)$.

3.4 Degree and Arity

In this section we give characterizations of the subexponential-time complexity of CSP and CSP^c with respect to the degree and the arity.

Proposition 11. *Unless the ETH fails, the restriction of* CSP *to instances whose* deg ≥ 2 *is not solvable in subexponential time.*

Proof. The statement follows from the proof of Theorem 1 after noting that, by Lemma 1, one can use 3-3-SAT in the reduction. This will result in instances of BOOLEAN 3-CSP with degree at most 3 as well. Now for each variable x of degree 3 in an instance of BOOLEAN 3-CSP, we introduce two new variables x', x'', and add a constraint whose variables are $\{x, x', x''\}$, and containing the two tuples $(0, 0, 0)$ and $(1, 1, 1)$; this constraint stipulates that the values of x, x', x'' be the same. We then substitute the variable x in one of the constraints it appears in with x', and in another constraint that it appears in with x''. Therefore, in the new instance, the degree of each of x, x', x'' becomes 2. After repeating this step to every variable of degree 3, we obtain an instance of BOOLEAN 3-CSP in which the degree of each variable is at most 2. Since the increase in the number of variables is linear, the above reduction is a serf-reduction from 3-3-SAT to BOOLEAN 3-CSP with degree at most 2, and gives the statement of the proposition. ☐

Proposition 12. *Unless the ETH fails, the restriction of* CSP *to instances whose* $\mathsf{arity} \geq 2$ *(and* $\mathsf{dom} \geq 3$*) is not solvable in subexponential time.*

Proof. We will give a serf-reduction from the 3-COLORABILITY problem to CSP with $\mathsf{arity} = 2$ and $\mathsf{dom} = 3$. Since the 3-COLORABILITY problem is SNP-complete under serf-reductions (Impagliazzo, Paturi & Zane, 2001), the statement of the theorem will follow. Recall that the 3-COLORABILITY problem asks if the vertices of a given graph can be properly colored (no two adjacent vertices are assigned the same color) with at most 3 colors.

The reduction is folklore. Given an instance of $G = (V, E)$ of 3-COLORABILITY, where G has n vertices, we construct an instance I of CSP as follows. The variables of I correspond to the vertices of G, and the domain of I corresponds to the color-set $\{1, 2, 3\}$. For every edge of the graph we construct a constraint of $\mathsf{arity} = 2$ over the two variables corresponding to the endpoint of the edge. The constraint contains all tuples corresponding to valid colorings of the endpoints of the edge. It is easy to see that G has a 3-coloring if and only if I is consistent. Since for the instance I we have $\mathsf{vars} = n$, which is the number of vertices in G, and since $\mathsf{arity} = 2$ and $\mathsf{dom} = 3$, this is a (polynomial-time) serf-reduction from the 3-COLORABILITY problem to CSP with $\mathsf{arity} = 2$ and $\mathsf{dom} = 3$. □

Clearly, Proposition 11 and Proposition 12 hold true for CSP^c as well.

We note that CSP^c and CSP with $\mathsf{dom} = 2$ and $\mathsf{arity} = 2$ are solvable in polynomial time via simple reductions to 2-CNF-SAT.

As it turns out, both CSP and CSP^c exhibit the same subexponential-time complexity behavior with respect to the same restrictions on the structural parameters considered above. On the other hand, the negative result proved in Proposition 3 for BOOLEAN CSP^c assumes a weaker hypothesis than the result about BOOLEAN CSP proved in Theorem 1.

4. CSP$^{\neq}$, CSP$^{=}$, CSP$^{\geq}$, and CSP$^{\leq}$

In this section we consider CSP$^{\neq}$, CSP$^{=}$, CSP$^{\geq}$, and CSP$^{\leq}$. Since our results for CSP$^{=}$, CSP$^{\geq}$, and CSP$^{\leq}$ are related, and rely on the results that we establish for CSP$^{\neq}$, we start by presenting our results for CSP$^{\neq}$.

4.1 CSP$^{\neq}$

Let $\mathcal{I}$ be an instance of CSP$^{\neq}$ with constraints $C_1, \ldots, C_c$ for some integer $c > 0$, over the set of variables $\{x_1, \ldots, x_n\}$. Denote by D_i, $i = 1, \ldots, n$, the domain of x_i.

Proposition 13. CSP$^{\neq}$ *can be solved in time* $O^*(2^n)$.

Proof. We reduce the instance $\mathcal{I}$ to an instance of the LIST COLORING problem. Recall that in the LIST COLORING problem we are given a graph, each of whose vertices is associated with a list of colors, and we are asked to decide if there exists a proper coloring of the graph such that each vertex is assigned a color from its list. To reduce $\mathcal{I}$ to an instance of LIST COLORING, we construct the graph G whose vertices are $x_1, \ldots, x_n$ (without loss of generality, we label the vertices in G with their corresponding variables' names in $\mathcal{I}$) and such that there is an edge between two vertices x_i and x_j, $1 \leq i < j \leq n$, if and only if x_i and x_j appear together in some constraint in $\mathcal{I}$. For each vertex x_i in G, associate with it a list of colors $L_i = D_i$. It is not difficult to see that $\mathcal{I}$ is a yes-instance of CSP$^{\neq}$ if and only if the graph G has a proper list coloring. It is known that the LIST COLORING problem is solvable in time $O^*(2^n)$ (Björklund, Husfeldt, & Koivisto, 2009), and hence so is CSP$^{\neq}$. $\square$

Corollary 1. *Let $d(n) = \omega(1)$ be a proper complexity function. The restriction of* CSP$^{\neq}$ *to instances in which $\mathsf{dom} \geq d(n)$ is solvable in subexponential time.*

Proof. Let $d(n) = \omega(1)$ be a proper complexity function, and consider the restriction of CSP$^{\neq}$ to instances in which $\mathsf{dom} \geq d(n)$. By Proposition 13, CSP$^{\neq}$ is solvable in time $O^*(2^n) = O^*(d(n)^{n/\log(d(n))}) \subseteq O^*(\mathsf{dom}^{o(n)})$. $\square$

We note that the above result may sound strange, especially when taken in conjunction with the next proposition, because it implies that the problem becomes "easier" for larger domain size. This can be explained by the fact that when the domain size gets large, the allowable upper bound on the subexponential time for solving the problem (i.e., $d(n)^{o(n)}$) gets larger as well.

By Corollary 1, we can focus our investigation of the subexponential-time complexity of CSP$^{\neq}$ on instances in which $\mathsf{dom} = O(1) = d$, for some integer constant d. Note that dom is an upper bound on arity because each constraint must have arity at most dom (otherwise it cannot be satisfied). If $d \leq 2$, then each constraint can have arity at most 2, and CSP$^{\neq}$ in this case reduces to 2-CNF-SAT, which is in P. Therefore, we can assume in the remainder of this section that $d \geq 3$.

Proposition 14. *Unless the ETH fails, the restriction of* CSP$^{\neq}$ *to instances in which $\mathsf{dom} = d \geq 3$ and $\mathsf{cons} = \Omega(n)$ is not solvable in subexponential time.*

Proof. It suffices to prove the result for $\mathsf{cons} = s(n)$, where $s(n)$ is any *specific* function such that $s(n)$ is linear in n ($\Theta(n)$), as the result would extend using a padding argument to any function that is linear in n (we can add new "dummy" variables and new "dummy" constraints on those new variables to make the relation between the constraints and the variables satisfy the desired function $s(\cdot)$).

By Lemma 1, 3-3-SAT is not solvable in subexponential time unless the ETH fails. The standard polynomial-time reduction from 3-SAT to 3-COLORABILITY (see Cormen et al., 2009), establishing the NP-hardness of 3-COLORABILITY, reduces an instance of 3-SAT on n variables and m clauses to an instance of 3-COLORABILITY with $O(n + m)$ vertices and $O(n + m)$ edges. Therefore, if we use the same reduction but start from 3-3-SAT instead of 3-SAT, we end up with an instance of 3-COLORABILITY in which the number of vertices is $O(n)$ and the number of edges is $O(n)$ as well. Hence, we have a serf-reduction from 3-3-SAT to the restriction of 3-COLORABILITY to instances whose size is linear in the number of vertices, denoted LINEAR-3-COLORABILITY. Now if we use the standard reduction from 3-COLORABILITY to CSP$^{\neq}$ (in which each vertex becomes a variable, each edge becomes a constraint of arity 2, and the domain is the set of 3 colors), but instead we start from an instance of LINEAR-3-COLORABILITY, we obtain an instance of CSP$^{\neq}$ on n variables (the same as the number of vertices in the graph), linear number of constraints, and domain size $\mathsf{dom} = 3$. Therefore, the previous reduction is a serf-reduction from LINEAR-3-COLORABILITY to the restriction of CSP$^{\neq}$ to instances in which the number of constraints is linear, and $\mathsf{dom} = 3$. Composing the two serf-reductions above gives a serf reduction from 3-3-SAT to the problem under consideration, and thus proves the proposition. $\square$

Remark 2. *We note that we did not phrase the statement of Corollary 1 to consider the restriction of* CSP$^{\neq}$ *to instances in which $\mathsf{dom} = \omega(1)$ (as we had been been doing in the paper) because such* *restriction will encompass a slice of the problem that is hard (instances whose domain size is upper bounded by a constant), as shown in Proposition 14. So we had to explicitly consider only instances whose domain size is lower-bounded by a function that is $\omega(1)$. Proposition 17, in the next section, is handled similarly.*

Remark 3. *We do not consider the restriction of* CSP$^{\neq}$ *to instances in which* $\mathsf{cons} = o(n)$ *and* $\mathsf{dom} = O(1)$. *This is because each constraint must have* $\mathsf{arity} \leq \mathsf{dom}$, *and hence, if* $\mathsf{cons} = o(n)$ *then it would follow that the total number of variables is* $o(n)$. *It follows that Proposition 14 and Corollary 1 provide tight characterizations of the subexponential-time complexity of* CSP$^{\neq}$ *with respect to each of* cons *and* dom.

The following proposition provides a tight characterization of the subexponential-time complexity of CSP$^{\neq}$ with respect to the treewidth of the primal graph:

Proposition 15. *The restriction of* CSP$^{\neq}$ *to instances in which* $\mathsf{tw} = o(n)$ *is solvable in subexponential time, and unless the ETH fails, the restriction of* CSP$^{\neq}$ *to instances in which* $\mathsf{tw} = \Omega(n)$ *is not solvable in subexponential time.*

Proof. To derive the subexponential-time result, we can assume that the domain size is d, for some constant $d \geq 3$, because in the other case we get that CSP$^{\neq}$ is solvable in subexponential time by Corollary 1. Let $\mathcal{I}$ be an instance of CSP$^{\neq}$ such that the treewidth of its primal graph is $o(n)$. Since the arity of each constraint in $\mathcal{I}$ is at most d and the domain size is d, in polynomial time we can reduce $\mathcal{I}$ to an instance of CSP on the same set of variables, and with the same domain, constraints, and primal treewidth. By part (i) of Theorem 5, the restriction of CSP to instances whose $\mathsf{tw} = o(n)$ is solvable in subexponential time, and hence $\mathcal{I}$ can be decided in subexponential time.

The hardness result follows from a general observation about the primal treewidth of CSP instances. First note that the number of variables n is an upper bound on the primal treewidth; that is, $\mathsf{tw} \leq n$. Therefore, for any upper bound $s(n) = \Omega(n)$ on tw, using a padding argument (adding a linear number of dummy new variables and singleton constraints that do not increase the primal treewidth) we can reduce a general instance of CSP$^{\neq}$ to an instance in which $\mathsf{tw} \leq s(n)$ at the cost of a linear increase in the number of variables and the instance size. This provides a serf-reduction from a general instance of CSP$^{\neq}$ to an instance in which $\mathsf{tw} \leq s(n) = \Omega(n)$. The result now follows from the same result for CSP$^{\neq}$ on general instances (implied, e.g., from Proposition 14). $\square$

It is well known that $\mathsf{tw} \leq \mathsf{arity} \cdot (\mathsf{tw}^* - 1)$ and $\mathsf{tw}^* \leq \mathsf{tw} + 1$ (Kolaitis & Vardi, 2000). If $\mathsf{arity} = O(1)$, then tw and tw^* are within a multiplicative constant from one another. Therefore, from Proposition 15 we can infer the following tight result:

Proposition 16. *The restriction of* CSP$^{\neq}$ *to instances in which* $\mathsf{tw}^* = o(n)$ *is solvable in subexponential time, and unless the ETH fails, the restriction of* CSP$^{\neq}$ *to instances in which* $\mathsf{tw}^* = \Omega(n)$ *is not solvable in subexponential time.*

Remark 4. *There are several width parameters for* CSP *that are even more general than* tw^* *in the sense that any instances for which* tw^* *is small, the other width parameter is small as well; but there are instances for which the other width parameter is small but* tw^* *can be arbitrarily large. Prominent examples for such width parameters are* hypertree width *(Gottlob et al., 2002) and* submodular width *(Marx, 2013). The lower bound statement of Proposition 16 clearly carries over to the more general width parameters. The same holds true for the lower bound statements in Proposition 19 and Theorem 5.*

4.2 CSP$^=$, CSP$^\geq$, and CSP$^\leq$

We start by presenting an exact algorithm for CSP$^\geq$; we do so by reducing CSP$^\geq$ to CSP$^{\neq}$. We use the example illustrated in Figure 2 as a running example to explain the idea behind this reduction. In this example, the instance of CSP$^\geq$ consists of three constraints C_1, C_2, C_3, where the variables in C_1 are x_1, x_2, x_3, x_4, the variables in C_2 are x_4, x_5, and the variables in C_3 are x_1, x_5, x_6, x_7. The domain of x_1 is $\{a, b\}$, the domain of both x_2 and x_3 is $\{b\}$, the domain of x_4 is $\{b, c\}$, the domain of x_5 is $\{a\}$, and the domain of both x_6 and x_7 is $\{d, e\}$. The number of distinct values that need to be assigned to the variables of C_1 is at least 3, to the variables of C_2 is at least 2, and to the variables of C_3 is at least 3.

In a solution $\mathcal{S}$ (i.e., an assignment of variables to domain values) to an instance $\mathcal{I}$ of CSP$^\geq$, and for a constraint C in $\mathcal{I}$, it is possible for several variables in C to be assigned the same value by the solution $\mathcal{S}$ (in the running example we are forced to assign both x_2 and x_3 the value b). Therefore, if we attempt a straightforward reduction from CSP$^\geq$ to CSP$^{\neq}$ that produces the same instance $\mathcal{I}$, the solution $\mathcal{S}$ to $\mathcal{I}$ as an instance of CSP$^\geq$ may not be a solution to $\mathcal{I}$ as an instance of CSP$^{\neq}$. It is possible that the above happens due to the fact that there are variables in $\mathcal{I}$ that can be removed without affecting the satisfiability of $\mathcal{I}$, because there is a solution to $\mathcal{I}$ in which each constraint will still be satisfied without considering the values assigned to those variables.

The algorithm starts by trying each subset of the variables as a subset for which there exists a solution in which each of those variables is "essential" for this solution; the algorithm then removes all the other (nonessential) variables, updates the instance, and works toward finding a solution under this assumption in the resulting instance. (In the running example, we remove x_3 from C_1; see the Venn diagram on the left in Figure 2.) Even with the above assumption, it is still possible that in a solution to the resulting instance, two variables in a constraint C are assigned the same value. One cannot simply ignore (remove) one of these variables on the basis that removing it will not affect the satisfiability of C, because the removed variable may contribute to the satisfiability of a constraint other than C, in which this variable appears as well. (In the running example, we are forced to assign both x_1 and x_5 the same value, which would violate constraint C_3 of CSP$^{\neq}$.) Therefore, the resulting instance, even though it may be a satisfiable instance of CSP$^\geq$, it may not be a satisfiable instance of CSP$^{\neq}$. However, as it will be shown in Lemma 2, it is possible in such an instance to "reassign" each variable to a subset of the constraints that it appears in, so that after this reassignment/repartitioning each variable contributes to the satisfiability of each constraint that it appears in. After such a reassignment, the resulting instance of CSP$^\geq$ becomes an equivalent instance of CSP$^{\neq}$. (In the running example, variable x_5 is not contributing to C_3, and can be safely reassigned to C_2; see the Venn diagram on the right in Figure 2.) We now proceed to the formal proofs.

Let $\mathcal{I}$ be an instance of CSP$^\geq$ with constraints $C_1, \ldots, C_c$ for some integer value $c > 0$, over the variables $x_1, \ldots, x_n$. Let n_i, $i = 1, \ldots, c$, be the nonnegative integer associated with constraint C_i. Denote by D_i, $i = 1, \ldots, n$, the domain of variable x_i, and let $D = \bigcup_{i=1}^{n} D_i$. Set $k = |D|$. If we consider each C_i, $i = 1, \ldots, c$, as a set consisting of all the variables in C_i, and we draw the Venn diagram for the C_i's, then this Venn diagram consists of at most $s \leq 2^c$ many nonempty *regions*, where each region R_j, $j = 1, \ldots, s$, is defined as the intersection of all the sets containing the variables that lie in R_j in the Venn diagram. For a solution $\mathcal{S}$ to the instance $\mathcal{I}$, we call a variable x_i *essential* (to $\mathcal{S}$) if discounting the value assigned to x_i by $\mathcal{S}$ violates at least one of the constraints (containing x_i), and hence no longer gives a solution to $\mathcal{I}$. It is clear that by enumerating every

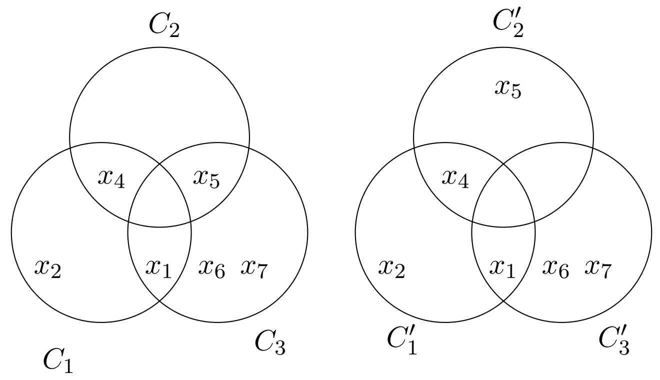

Figure 2: Illustration of the example of the reduction from CSP$^{\geq}$ to CSP$^{\neq}$.

subset of the variables in $\mathcal{I}$, which takes $O(2^n)$ time, we can work under the assumption that we are looking for a solution such that every variable is essential to $\mathcal{S}$. Since we are working on an instance of CSP$^{\geq}$, adding the nonessential variables to the solution afterwards (and assigning them values from their domains) will not hurt the solution. Therefore, without loss of generality, we will assume that each of the variables $x_1, \ldots, x_n$ is essential to the solution sought (if any exists). We start with the following lemma.

Lemma 2 (The Repartitioning Lemma). *Let $\mathcal{I}$ be an instance of* CSP$^{\geq}$. *There is a solution to $\mathcal{I}$ if and only if there is an instance $\mathcal{I}'$ on the same set of variables as $\mathcal{I}$, and whose constraints are $C_1', \ldots, C_c'$, such that:*

(1) the variables in C_i' are a subset of those in C_i, for $i = 1, \ldots, c$;

(2) the numbers $n_1, \ldots, n_c$ are the same in both $\mathcal{I}$ and $\mathcal{I}'$; and

(3) there is a solution to $\mathcal{I}'$ satisfying that for every value v, and for any two distinct variables x_i, x_j that are assigned the value v in the solution for $\mathcal{I}'$, the set of constraints that x_i belongs to in $\mathcal{I}'$ is disjoint from that that x_j belongs to in $\mathcal{I}'$.

Proof. Suppose that $\mathcal{I}$ has a solution $\mathcal{S}$; by the discussion preceding this lemma, we can assume that every variable is essential to $\mathcal{S}$. We define the instance $\mathcal{I}'$ on the same set of variables as $\mathcal{I}$ as follows. The constants $n_1, \ldots, n_c$ remain the same in $\mathcal{I}'$. We define the constraints in $\mathcal{I}'$ by a sequence of changes performed to the constraints in $\mathcal{I}$; initially the constraints of $\mathcal{I}'$ are identical to those of $\mathcal{I}$. For every value $v \in D$ assigned to some variable by the solution $\mathcal{S}$, let $x_v^1, \ldots, x_v^\ell$ be the variables assigned the value v by $\mathcal{S}$. For each x_v^j, $j = 1, \ldots, \ell - 1$, considered in the listed order, let $\mathcal{C}_v^j$ be the set of constraints containing x_v^j in $\mathcal{I}'$, and let $\mathcal{C}_{v,\cup}^j$ be the union of all constraints containing any of the variables $x_v^{j+1}, \ldots, x_v^\ell$. Remove x_v^j from each constraint in $\mathcal{C}_v^j \cap \mathcal{C}_{v,\cup}^j$.

We claim that the same solution to $\mathcal{I}$ is a solution to $\mathcal{I}'$ that satisfies all the conditions in the statement of the lemma. First, from the construction of the constraints in $\mathcal{I}'$, for any value v in the solution, the set of constraints containing each variable assigned the value v are mutually disjoint because each variable x_v^i ($i < \ell$) assigned a value v is removed from each constraint that some subsequent variable in $x_v^{i+1}, \ldots, x_v^\ell$ is contained in. Moreover, because each constraint C_i' is obtained from C_i only by (possibly) removing variables from C_i, we have $C_i' \subseteq C_i$, for $i = 1, \ldots, c$. Finally, when a variable x_v^i that is assigned a value v is removed from a constraint C'_j, this removal will not affect the number of different values assigned to the variables in C'_j by $\mathcal{S}$; this is because we know for sure that there will be a subsequent variable x_v^p, $p \in \{i+1, \ldots, \ell\}$, that is assigned value v and that will remain in C'_j, namely the variable x_v^p with the maximum index p that appears in C'_j.

Conversely, because each C'_i is a subset of C_i, for $i = 1, \ldots, c$, it is easy to see that any solution to $\mathcal{I}'$ is also a solution to $\mathcal{I}$. □

Theorem 6. CSP$^{\geq}$ *can be solved in time* $O^*((2^{(\mathsf{cons}+1)}+1)^n)$.

Proof. Let $\mathcal{I}$ be an instance of CSP$^{\geq}$ with constraints $C_1, \ldots, C_c$ for some integer $c > 0$, over the variables $x_1, \ldots, x_n$. Let n_i, $i = 1, \ldots, c$, be the nonnegative integer associated with constraint C_i.

We first enumerate each subset of the variables $\{x_1, \ldots, x_n\}$ as the subset of essential variables for the solution $\mathcal{S}$ sought. Fix such an enumerated subset X, remove the other variables from $\mathcal{I}$, and update the instance accordingly (i.e., update the constraints); without loss of generality, we will still refer to the resulting instance as $\mathcal{I}$.

By Lemma 2, there is a solution to $\mathcal{I}$ if and only if there is an instance $\mathcal{I}'$ on the same set of variables as $\mathcal{I}$, and whose constraints are $C'_1, \ldots, C'_c$, such that: (1) the variables in C'_i form a subset of those in C_i, for $i = 1, \ldots, c$, (2) the numbers $n_1, \ldots, n_c$ are the same in both $\mathcal{I}$ and $\mathcal{I}'$, and (3) there is a solution to $\mathcal{I}'$ satisfying that for every value v, and for any two distinct variables x_i, x_j that are assigned the value v in the solution for $\mathcal{I}'$, the set of constraints that x_i belongs to in $\mathcal{I}'$ is disjoint from that that x_j belongs to in $\mathcal{I}'$.

To find the instance $\mathcal{I}'$, we will try every possible partitioning of the variables in X into c constraints to determine the new constraints $C'_1, \ldots, C'_c$ in $\mathcal{I}'$. For each such partitioning π in which $C'_i \subseteq C_i$ and at least n_i variables are in C'_i, for $i = 1, \ldots, c$, we form the instance of CSP$^{\neq}$ on the set of variables X and the set of constraints $C'_1, \ldots, C'_c$, and invoke the algorithm for CSP$^{\neq}$ described in Proposition 13 on this instance; if the algorithm returns a solution then we return the same solution as a solution to $\mathcal{I}$. If for each enumerated subset X and each enumerated partitioning π the algorithm for CSP$^{\neq}$ rejects, then we reject the instance $\mathcal{I}$.

It is easy to see the correctness of the above algorithm. Clearly, if there is a solution to the CSP$^{\neq}$ instance then there is a solution to $\mathcal{I}'$, and hence to $\mathcal{I}$. This is because each constraint contains at least n_i variables, which must receive n_i distinct values in the solution to the CSP$^{\neq}$ instance, hence satisfying each constraint C_i and satisfying $\mathcal{I}$. On the other hand, if $\mathcal{I}$ has a solution, then there is an enumerated partitioning of the variables in X that will correspond to the constraints in $\mathcal{I}'$. Now because there is a solution to $\mathcal{I}'$ that satisfies properties (1)-(3) in Lemma 2, no two variables in the same constraint of $\mathcal{I}'$ receive the same value v in this solution (by property (3)). Therefore, this solution will also be a solution to the constructed instance of CSP$^{\neq}$. This shows the correctness of the above algorithm.

The running time of the algorithm is the time taken to enumerate all subsets of the variables, and for each subset X, the time to enumerate all partitions of X into c constraints, and finally for each such partition the time taken to invoke the CSP$^{\neq}$ algorithm on the resulting instance. The number of subsets of variables of $\{x_1, \ldots, x_n\}$ is $\sum_{i=0}^{n} \binom{n}{i}$. For each subset of cardinality i, there are at most 2^{ci} many ways of partitioning it into c constraints. Finally, for each instance on i variables, the CSP$^{\neq}$ algorithm takes $O^*(2^i)$ time. Putting everything together, the overall running time of the algorithm is a polynomial factor multiplied by:

$$\sum_{i=0}^{n} \binom{n}{i} \cdot 2^{ci} \cdot 2^{i} = \sum_{i=0}^{n} \binom{n}{i} \cdot 2^{(c+1)i} = (2^{(c+1)} + 1)^n.$$

Therefore, the running time of the algorithm is $O^*((2^{(\mathsf{cons}+1)} + 1)^n)$ as claimed. □

Corollary 2. *The restriction of* CSP$^{\geq}$ *to instances in which* $\mathsf{cons} = O(1)$ *is solvable in* $O^*(2^{O(n)})$ *time.*

Corollary 3. *The restriction of* CSP$^{\geq}$ *to instances in which* $\mathsf{cons} = o(\log \mathsf{dom})$ *is solvable in subexponential time.*

Proof. The result follows from Theorem 6 after noticing that if $\mathsf{cons} = o(\log \mathsf{dom})$ then $2^{\mathsf{cons}} = \mathsf{dom}^{o(1)}$. □

Proposition 17. *Let* $d(n) = \omega(1)$ *be a proper complexity function. The restriction of* CSP$^{\geq}$ *to instances in which* $\mathsf{cons} = O(1)$ *and* $\mathsf{dom} \geq d(n)$ *is solvable in subexponential time, and unless the ETH fails, the restriction of* CSP$^{\geq}$ *to instances in which* $\mathsf{cons} = \Omega(n)$ *(even when* $\mathsf{dom} = O(1)$*) is not solvable in subexponential time.*

Proof. The positive result follows from Corollary 3. The hardness result follows from the hardness result for CSP$^{\neq}$ in Proposition 14 (CSP$^{\neq}$ is a special case of CSP$^{\geq}$). □

The NP-hardness reduction for CSP$^{=}$ with a single constraint (and linear domain size), given by Bessiere et al. (2007), which also works for CSP$^{\leq}$, is actually a serf-reduction from 3-CNF-SAT. This implies that, unless the ETH fails, neither CSP$^{=}$ nor CSP$^{\leq}$, restricted to instances with a single constraint and $\mathsf{dom} = O(n)$, is solvable in subexponential time. We show next the same result for the restrictions of CSP$^{\leq}$ and CSP$^{=}$ to instances with a constant domain size and a linear number of constraints:

Theorem 7. *The restrictions of* CSP$^{\leq}$ *and* CSP$^{=}$ *to instances where* $\mathsf{dom} = O(1)$ *and* $\mathsf{cons} = \Omega(n)$ *are not solvable in subexponential time, unless the ETH fails.*

Proof. We give a serf-reduction from 3-3-SAT to CSP$^{\leq}$; the result will then follow by Lemma 1. The same serf-reduction also works for the case of CSP$^{=}$. Take an instance φ of 3-3-SAT with n variables. We construct in polynomial time an instance of CSP$^{\leq}$, with $\mathsf{cons} = O(n)$ and $\mathsf{dom} = O(1)$ that is a yes-instance if and only if $\varphi \in$ 3-3-SAT. We proceed in two steps: firstly, we modify the well-known polynomial-time reduction from 3-SAT to VERTEX COVER (Garey & Johnson, 1979) to a reduction from 3-3-SAT to CSP$^{\leq}$, resulting in an instance with $\mathsf{cons} = O(n)$ and $\mathsf{dom} = O(n)$; secondly, we transform this instance of CSP$^{\leq}$ to an equivalent instance of CSP$^{\leq}$ with $\mathsf{cons} = O(n)$ and $\mathsf{dom} = O(1)$.

We start with the first step. Let φ consist of the clauses $c_1, \dots, c_m$, where $c_i = l_1^i \vee l_2^i \vee l_3^i$ for each $1 \leq i \leq m$. The well-known reduction to VERTEX COVER produces a graph $G = (V, E)$, containing vertices $v_x, v_{\overline{x}}$ for each variable x occurring in φ, and a vertex v_j^i for each literal occurrence, where $1 \leq i \leq m$ and $1 \leq j \leq 3$. The variables v_x and $v_{\overline{x}}$ are adjacent, for each variable x, and the vertices v_1^i, v_2^i, v_3^i form a triangle, for each $1 \leq i \leq m$. Moreover, there is an edge between v_j^i and v_l, where $l = l_j^i$. Then φ is satisfiable if and only if G has a vertex cover consisting of $n + 2m$ vertices. More specifically, φ is satisfiable if and only if G has a vertex cover containing exactly one vertex from $v_x, v_{\overline{x}}$ for each variable x and exactly two vertices from v_1^i, v_2^i, v_3^i for each $1 \leq i \leq m$. We now construct an instance of CSP$^{\leq}$ as follows. For each edge $e = \{v_1, v_2\} \in E$, we introduce a variable u_e with domain $\{v_1, v_2\}$. Then, for each clause c_i, we define the set E_{c_i} to consist of all edges between v_1^i, v_2^i, v_3^i, between v_j^i and $v_{l_j^i}$ and between $v_{l_j^i}$ and $v_{\overline{l_j^i}}$, for each $1 \leq j \leq 3$. Then, we add a constraint ensuring that the variables u_e for all nine $e \in E_{c_i}$ take at most 5 different values. The assignments to the variables u_e that satisfy all these constraints exactly correspond to the vertex covers of G containing exactly one vertex from $v_x, v_{\overline{x}}$ for each variable x and exactly two vertices from v_1^i, v_2^i, v_3^i for each $1 \leq i \leq m$. These particular vertex covers, in turn, correspond exactly to truth assignments (which set one of $x, \overline{x}$ to true, for each variable x) satisfying φ. The construction of such a constraint is illustrated in Figure 3.

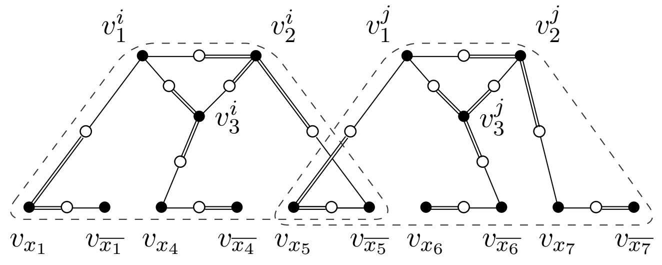

Figure 3: The CSP$^{\leq}$ constraints corresponding to example clauses $c_i = (x_1 \vee x_4 \vee \overline{x_5})$ and $c_j = (x_5 \vee \overline{x_6} \vee x_7)$. Variables are denoted by $\circ$, and values by $\bullet$. The constraints are indicated by dashed lines. The nine variables in each constraint must be assigned to at most 5 different values. The double lines indicate an assignment to the variables satisfying the constraint that corresponds to the truth assignment $\{x_1 \mapsto \top, x_4 \mapsto \bot, x_5 \mapsto \top, x_6 \mapsto \top, x_7 \mapsto \bot\}$.

In the second step, we transform the instance of CSP$^{\leq}$ in such a way that $\mathsf{dom} = O(1)$. In order to do so, we will use the following observation. Whenever two vertices $v_1, v_2 \in V$ have the property that there is no constraint both containing a variable u_{e_1} for some edge e_1 incident with v_1 and a variable u_{e_2} for some edge e_2 incident with v_2, then we can safely identify the domain values v_1 and v_2 in the instance of CSP$^{\leq}$. Consequently, we can identify all m many domain values $v_1^1, \dots, v_1^m$ into a single value, and similarly identify all domain values $v_2^1, \dots, v_2^m$ and $v_3^1, \dots, v_3^m$. Next, to reduce dom even more, we will identify a number of domain values v_x with each other (and similarly identify their complementary values $v_{\overline{x}}$ with each other). Consider the primal graph of φ, i.e., the graph G_φ^p containing as vertices the variables of φ where two vertices x, x' are adjacent if and only if x and x' occur together in a clause (positively or negatively). Since each variable occurs at most 3 times in φ, we know that the maximum degree of G_φ^p is bounded above by 8. Then, by Brooks' Theorem (Brooks, 1941), we know that there exists a proper coloring of G_φ^p by at most 9 colors, and that such a coloring can be computed in linear time. Take such a proper coloring c of G_φ^p. Now, for each color b used by the coloring c, we let $X_b \subseteq \text{Var}(\varphi)$ be the set of variables x such that $c(x) = b$. Then, since c is a proper coloring of the primal graph G_φ^p of φ, we know that for any color b no two variables $x, x' \in X_b$ occur together in any clause of φ. Therefore, for each color $1 \leq b \leq 3$ we can safely identify all domain values v_x for $x \in X_b$ with each other in the instance of CSP$^{\leq}$, and similarly we can safely identify all domain values $v_{\overline{x}}$ for $x \in X_b$ with each other. This results in an equivalent instance of CSP$^{\leq}$ with $\mathsf{cons} = O(n)$ and $\mathsf{dom} = O(1)$. $\square$

We next consider the subexponential-time complexity of $\text{CSP}^{=}$, $\text{CSP}^{\geq}$, and $\text{CSP}^{\leq}$ with respect of the primal treewidth. We have the following tight result:

Proposition 18. *The restriction of each of* $\text{CSP}^{=}$, $\text{CSP}^{\geq}$, *and* $\text{CSP}^{\leq}$ *to instances in which* $\mathsf{tw} = o(n)$ *is solvable in subexponential time, and unless the ETH fails, the restriction of each of* $\text{CSP}^{=}$, $\text{CSP}^{\geq}$, *and* $\text{CSP}^{\leq}$ *to instances in which* $\mathsf{tw} = \Omega(n)$ *is not solvable in subexponential time.*

Proof. The proof of this proposition for each of $\text{CSP}^{=}$, $\text{CSP}^{\geq}$, and $\text{CSP}^{\leq}$ is exactly the same as the proof of Proposition 15. □

Finally, the following hardness result for $\text{CSP}^{=}$ and $\text{CSP}^{\geq}$ with respect to tw^{*} follows from Proposition 16 since $\text{CSP}^{\neq}$ is a special case of each of $\text{CSP}^{=}$ and $\text{CSP}^{\geq}$:

Proposition 19. *Unless the ETH fails, the restriction of* $\text{CSP}^{=}$ *(resp.* $\text{CSP}^{\geq}$*) to instances in which* $\mathsf{tw}^{*} = \Omega(n)$ *is not solvable in subexponential time.*

5. Conclusion

We have provided a first analysis of the subexponential-time complexity of CSP with extensionally represented constraints and CSP with global constraints, for the latter focusing on instances that are composed of the fundamental global constraints *AllDifferent*, *AtLeastNValue*, *AtMostNValue*, and *cTable*, respectively. Our results show a detailed complexity landscape for these problems under various natural structural restrictions. In most cases, we were able to obtain tight bounds that exactly determine the borderline between the classes of instances that can be solved in subexponential time, and those for which the existence of subexponential-time algorithms is unlikely. There are several ways for extending the current work such as considering other global constraints, the combination of different global constraints, and other structural restrictions on the primal or incidence graphs.

References

Alber, J., Fernau, H., & Niedermeier, R. (2004). Parameterized complexity: exponential speed-up for planar graph problems. *Algorithmica*, *52*(1), 26–56.

Amir, E. (2010). Approximation algorithms for treewidth. *Algorithmica*, *56*(4), 448–479.

Bäckström, C., & Jonsson, P. (2011). All pspace-complete planning problems are equal but some are more equal than others. In Borrajo, D., Likhachev, M., & López, C. L. (Eds.), *Proceedings of the Fourth Annual Symposium on Combinatorial Search, SOCS 2011, Castell de Cardona, Barcelona, Spain, July 15.16, 2011*. AAAI Press.

Beigel, R., & Eppstein, D. (2005). 3-coloring in time $\mathcal{O}(1.3289^n)$. *J. Algorithms*, *54*(2), 168–204.

Beldiceanu, N., Carlsson, M., & Rampon, J.-X. (2006). Global constraint catalog. Tech. rep. T2005:08, SICS, SE-16 429 Kista, Sweden. On-line version at http://www.emn.fr/x-info/sdemasse/gccat/.

Benhamou, B., Paris, L., & Siegel, P. (2012). Dealing with satisfiability and n-ary CSPs in a logical framework. *Journal of Automated Reasoning*, *48*(3), 391–417.

Bennaceur, H. (2004). A comparison between SAT and CSP techniques. *Constraints*, *9*(2), 123–138.

Bessiere, C., Hebrard, E., Hnich, B., Kiziltan, Z., & Walsh, T. (2006). Filtering algorithms for the NValue constraint. *Constraints*, *11*(4), 271–293.

Bessiere, C., Hebrard, E., Hnich, B., & Walsh, T. (2007). The complexity of reasoning with global constraints. *Constraints*, *12*(2), 239–259.

Björklund, A., Husfeldt, T., & Koivisto, M. (2009). Set partitioning via inclusion-exclusion. *SIAM J. Comput.*, *39*(2), 546–563.

Brooks, R. L. (1941). On colouring the nodes of a network. *Mathematical Proceedings of the Cambridge Philosophical Society*, *37*, 194–197.

Calabro, C., Impagliazzo, R., & Paturi, R. (2006). A duality between clause width and clause density for SAT. In *21st Annual IEEE Conference on Computational Complexity (CCC 2006), 16-20 July 2006, Prague, Czech Republic*, pp. 252–260. IEEE Computer Society.

Chen, H., & Grohe, M. (2010). Constraint satisfaction with succinctly specified relations. *J. of Computer and System Sciences*, *76*(8), 847–860.

Chen, J., Kanj, I., Perkovic, L., Sedgwick, E., & Xia, G. (2007). Genus characterizes the complexity of certain graph problems: Some tight results. *Journal of Computer and System Sciences*, *73*(6), 892–907.

Chen, J., Chor, B., Fellows, M., Huang, X., Juedes, D., Kanj, I. A., & Xia, G. (2005). Tight lower bounds for certain parameterized NP-hard problems. *Information and Computation*, *201*(2), 216–231.

Chen, J., Huang, X., Kanj, I. A., & Xia, G. (2006). Strong computational lower bounds via parameterized complexity. *J. of Computer and System Sciences*, *72*(8), 1346–1367.

Chen, J., Kanj, I. A., & Xia, G. (2009). On parameterized exponential time complexity. *Theoretical Computer Science*, *410*(27-29), 2641–2648.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms* (Third edition). The MIT Press, Cambridge, MA.

Dechter, R. (2003). *Constraint Processing*. Morgan Kaufmann.

Demaine, E., Fomin, F., Hajiaghayi, M., & Thilikos, D. (2005). Subexponential parameterized algorithms on bounded-genus graphs and H-minor-free graphs. *J. ACM*, *52*, 866–893.

Dimopoulos, Y., & Stergiou, K. (2006). Propagation in CSP and SAT. In Benhamou, F. (Ed.), *Principles and Practice of Constraint Programming - CP 2006, 12th International Conference, CP 2006, Nantes, France, September 25-29, 2006, Proceedings*, Vol. 4204 of *Lecture Notes in Computer Science*, pp. 137–151. Springer Verlag.

Feder, T., & Motwani, R. (2002). Worst-case time bounds for coloring and satisfiability problems. *J. Algorithms*, *45*(2), 192–201.

Fellows, M. R., Fomin, F. V., Lokshtanov, D., Rosamond, F., Saurabh, S., Szeider, S., & Thomassen, C. (2011a). On the complexity of some colorful problems parameterized by treewidth. *Information and Computation*, *209*(2), 143–153.

Fellows, M. R., Friedrich, T., Hermelin, D., Narodytska, N., & Rosamond, F. A. (2011b). Constraint satisfaction problems: Convexity makes alldifferent constraints tractable. In Walsh, T. (Ed.), *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pp. 522–527. IJCAI/AAAI.

Flum, J., & Grohe, M. (2006). *Parameterized Complexity Theory*, Vol. XIV of *Texts in Theoretical Computer Science. An EATCS Series*. Springer Verlag, Berlin.

Freuder, E. C. (1982). A sufficient condition for backtrack-bounded search. *J. of the ACM*, *29*(1), 24–32.

Freuder, E. C. (1990). Complexity of k-tree structured constraint satisfaction problems. In Shrobe, H. E., Dietterich, T. G., & Swartout, W. R. (Eds.), *Proceedings of the 8th National Conference on Artificial Intelligence. Boston, Massachusetts, July 29 - August 3, 1990, 2 Volumes*, pp. 4–9. AAAI Press / The MIT Press.

Garey, M. R., & Johnson, D. R. (1979). *Computers and Intractability*. W. H. Freeman and Company, New York, San Francisco.

Gaspers, S., & Szeider, S. (2014). Guarantees and limits of preprocessing in constraint satisfaction and reasoning. *Artificial Intelligence*, *216*, 1–19.

Ge, R. (2013). *Provable Algorithms for Machine Learning Problems*. Ph.D. thesis, Princeton University.

Gottlob, G., Leone, N., & Scarcello, F. (2002). Hypertree decompositions and tractable queries. *J. of Computer and System Sciences*, *64*(3), 579–627.

Grandoni, F., & Italiano, G. F. (2006). Algorithms and constraint programming. In Benhamou, F. (Ed.), *Principles and Practice of Constraint Programming - CP 2006, 12th International Conference, CP 2006, Nantes, France, September 25-29, 2006, Proceedings*, Vol. 4204 of *Lecture Notes in Computer Science*, pp. 2–14. Springer Verlag.

Grohe, M. (2006). The structure of tractable constraint satisfaction problems. In Kralovic, R., & Urzyczyn, P. (Eds.), *Mathematical Foundations of Computer Science 2006, 31st International Symposium, MFCS 2006, Stará Lesná, Slovakia, August 28-September 1, 2006, Proceedings*, Vol. 4162 of *Lecture Notes in Computer Science*, pp. 58–72. Springer Verlag.

de Haan, R., Kanj, I., & Szeider, S. (2014). Subexponential time complexity of CSP with global constraints. In *Proceedings of CP 2014, the 20th International Conference on Principles and Practice of Constraint Programming, Lyon, France, September 8-12, 2014*. Springer Verlag.

Hnich, B., Kiziltan, Z., & Walsh, T. (2004). Combining symmetry breaking with other constraints: Lexicographic ordering with sums. In *AI&M 1-2004, Eighth International Symposium on Artificial Intelligence and Mathematics, January 4-6, 2004, Fort Lauderdale, Florida, USA*.

van Hoeve, W.-J., & Katriel, I. (2006). Global constraints. In Rossi, F., van Beek, P., & Walsh, T. (Eds.), *Handbook of Constraint Programming*, chap. 6. Elsevier.

Impagliazzo, R., & Paturi, R. (2001). On the complexity of k-SAT. *J. of Computer and System Sciences*, *62*(2), 367–375.

Impagliazzo, R., Paturi, R., & Zane, F. (2001). Which problems have strongly exponential complexity?. *J. of Computer and System Sciences*, *63*(4), 512–530.

Jeavons, P., & Petke, J. (2012). Local consistency and SAT-solvers. *J. Artif. Intell. Res.*, *43*, 329–351.

Jonsson, P., Lagerkvist, V., & Nordh, G. (2013). Blowing holes in various aspects of computational problems, with applications to constraint satisfaction. In Schulte, C. (Ed.), *Principles and Practice of Constraint Programming - 19th International Conference, CP 2013, Uppsala,* *Sweden, September 16-20, 2013. Proceedings*, Vol. 8124 of *Lecture Notes in Computer Science*, pp. 398–414. Springer Verlag.

Kanj, I., & Szeider, S. (2013). On the subexponential time complexity of CSP. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*. AAAI Press.

Katsirelos, G., & Walsh, T. (2007). A compression algorithm for large arity extensional constraints. In Bessiere, C. (Ed.), *Principles and Practice of Constraint Programming - CP 2007, 13th International Conference, CP 2007, Providence, RI, USA, September 23-27, 2007, Proceedings*, Vol. 4741 of *Lecture Notes in Computer Science*, pp. 379–393. Springer Verlag.

Kolaitis, P. G., & Vardi, M. Y. (2000). Conjunctive-query containment and constraint satisfaction. *J. of Computer and System Sciences*, *61*(2), 302–332. Special issue on the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (Seattle, WA, 1998).

Kutz, M., Elbassioni, K., Katriel, I., & Mahajan, M. (2008). Simultaneous matchings: hardness and approximation. *J. of Computer and System Sciences*, *74*(5), 884–897.

Kwisthout, J., Bodlaender, H. L., & van der Gaag, L. C. (2010). The necessity of bounded treewidth for efficient inference in Bayesian networks. In Coelho, H., Studer, R., & Wooldridge, M. (Eds.), *ECAI 2010 - 19th European Conference on Artificial Intelligence, Lisbon, Portugal, August 16-20, 2010, Proceedings*, Vol. 215 of *Frontiers in Artificial Intelligence and Applications*, pp. 237–242. IOS Press.

Lokshtanov, D., Marx, D., & Saurabh, S. (2011). Lower bounds based on the exponential time hypothesis. *Bulletin of the European Association for Theoretical Computer Science*, *105*, 41–72.

Marx, D. (2010). Can you beat treewidth?. *Theory of Computing*, *6*, 85–112.

Marx, D. (2013). Tractable hypergraph properties for constraint satisfaction and conjunctive queries. *J. of the ACM*, *60*(6), Art. 42, 51.

Moser, R. A., & Scheder, D. (2011). A full derandomization of Schöning's k-SAT algorithm. In *STOC'11—Proceedings of the 43rd ACM Symposium on Theory of Computing*, pp. 245–251. ACM, New York.

Pachet, F., & Roy, P. (1999). Automatic generation of music programs. In Jaffar, J. (Ed.), *Principles and Practice of Constraint Programming - CP'99, 5th International Conference, Alexandria, Virginia, USA, October 11-14, 1999, Proceedings*, Vol. 1713 of *Lecture Notes in Computer Science*, pp. 331–345. Springer Verlag.

Papadimitriou, C. H. (1994). *Computational Complexity*. Addison-Wesley.

Papadimitriou, C. H., & Yannakakis, M. (1991). Optimization, approximation, and complexity classes. *J. of Computer and System Sciences*, *43*(3), 425–440.

Papadimitriou, C. H., & Yannakakis, M. (1999). On the complexity of database queries. *J. of Computer and System Sciences*, *58*(3), 407–427.

Razgon, I. (2006). Complexity analysis of heuristic CSP search algorithms. In Hnich, B., Carlsson, M., Fages, F., & Rossi, F. (Eds.), *Recent Advances in Constraints, Joint ERCIM/CoLogNET International Workshop on Constraint Solving and Constraint Logic Programming, CSCLP*

2005, Uppsala, Sweden, June 20-22, 2005, Revised Selected and Invited Papers, Vol. 3978 of *Lecture Notes in Computer Science*, pp. 88–99. Springer Verlag.

Régin, J.-C. (1994). A filtering algorithm for constraints of difference in CSPs. In Hayes-Roth, B., & Korf, R. E. (Eds.), *Proceedings of the 12th National Conference on Artificial Intelligence, Seattle, WA, USA, July 31 - August 4, 1994, Volume 1*, pp. 362–367. AAAI Press / The MIT Press.

Régin, J.-C. (1995). *Développement d'outils algorithmiques pour l'Intelligence Artificielle*. Ph.D. thesis, Montpellier II. in French.

Régin, J.-C. (2011). Global constraints: A survey. In van Hentenryck, P., & Milano, M. (Eds.), *Hybrid Optimization: The Ten Years of CPAIOR*, Vol. 45 of *Optimization and Its Applications*, chap. 3, pp. 63–134. Springer Verlag.

Régin, J.-C., & Rueher, M. (2000). A global constraint combining a sum constraint and difference constraints. In Dechter, R. (Ed.), *Principles and Practice of Constraint Programming - CP 2000, 6th International Conference, Singapore, September 18-21, 2000, Proceedings*, Vol. 1894 of *Lecture Notes in Computer Science*, pp. 384–395. Springer Verlag.

Rossi, F., van Beek, P., & Walsh, T. (Eds.). (2006). *Handbook of Constraint Programming*. Elsevier.

Samer, M., & Szeider, S. (2010). Constraint satisfaction with bounded treewidth revisited. *J. of Computer and System Sciences*, *76*(2), 103–114.

Schöning, U. (1999). A probabilistic algorithm for k-SAT and constraint satisfaction problems. In *40th Annual Symposium on Foundations of Computer Science (New York, 1999)*, pp. 410–414. IEEE Computer Soc., Los Alamitos, CA.

Schuler, R. (2005). An algorithm for the satisfiability problem of formulas in conjunctive normal form. *J. Algorithms*, *54*(1), 40–44.

Traxler, P. (2008). The time complexity of constraint satisfaction. In Grohe, M., & Niedermeier, R. (Eds.), *Parameterized and Exact Computation, Third International Workshop, IWPEC 2008, Victoria, Canada, May 14-16, 2008. Proceedings*, Vol. 5018 of *Lecture Notes in Computer Science*, pp. 190–201. Springer Verlag.