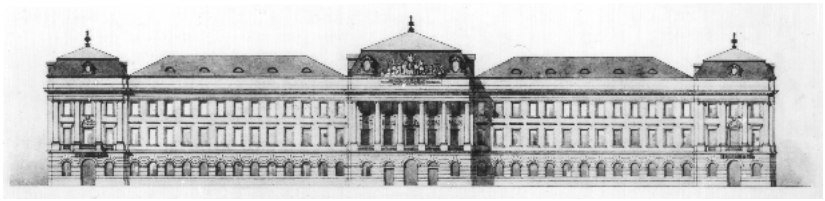
INFSYS
RESEARCH
REPORT



INSTITUT FÜR INFORMATIONSSYSTEME
ABTEILUNG WISSENSBASIERTE SYSTEME

SUBEXPONENTIAL TIME COMPLEXITY OF CSP WITH GLOBAL CONSTRAINTS

Ronald de Haan Iyad Kanj Stefan Szeider

INFSYS RESEARCH REPORT 1843-14-06
DECEMBER 2014

Institut für Informationssysteme
Abtg. Wissensbasierte Systeme
Technische Universität Wien
Favoritenstraße 9-11
A-1040 Wien, Austria
Tel: +43-1-58801-18405
Fax: +43-1-58801-18493
sek@kr.tuwien.ac.at
www.kr.tuwien.ac.at

TU
TECHNISCHE UNIVERSITÄT WIEN

INFSYS Research Report
INFSYS Research Report 1843-14-06, December 2014

Subexponential Time Complexity of CSP with Global Constraints

Ronald de Haan[1*] Iyad Kanj[2] Stefan Szeider[1*]

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. Recently, a first analysis of the subexponential time complexity of classical CSPs (where all constraints are given extensionally as tables) was given.

In this paper, we extend this analysis to CSPs in which constraints are given intensionally in the form of global constraints. In particular, we consider CSPs that use the fundamental global constraints *AllDifferent*, *AtLeastNValue*, *AtMostNValue*, and constraints that are specified by compressed tuples (*cTable*). We provide tight characterizations of the subexponential time complexity of the aforementioned problems with respect to several natural structural parameters.

---

[1] Institute of Information Systems, Vienna University of Technology, Vienna, Austria
[2] School of Computing, DePaul University, Chicago, IL

* Supported by the European Research Council (ERC), project 239962 (COMPLEX REASON), and the Austrian Science Fund (FWF), project P26200 (Parameterized Compilation).

Publication Information: This is the authors' self-archived copy of a paper that appeared in the *Proceedings of the 20th International Conference on Principles and Practice of Constraint Programming (CP 2014)*, dx.doi.org/10.1007/978-3-319-10428-7_21. The final publication is available at www.springer.com.

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. In particular, there seem to be various grades of "empirical hardness" among several NP-complete variants of the constraint satisfaction problem (CSP). It is a central challenge for theoreticians to develop a framework, that is more fine graded 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 [27]. 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 (see [8, 12]). 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 [21], 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).

In a recent paper, Kanj and Szeider [23] provided a first analysis of the subexponential time complexity of the classical CSP, where all constraints are given extensionally in the form of tables. In this paper, we extend this line of research by considering CSPs where constraints are specified intensionally using *global constraints*. This extension is highly relevant since 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 [33, 36].

In particular, we consider CSPs in which the global constraints are all either *AllDifferent* constraints, *NValue* constraints, *AtLeastNValue* constraints, *AtMostNValue* constraints, or constraints that are specified by tables with compressed tuples (*cTable*). We provide tight characterizations of the subexponential time complexity of the aforementioned CSPs with respect to several natural parameters of the problem instance. For example, we show that the CSP with *AllDifferent* constraints is solvable in subexponential time if the domain size is $\omega(1)$ (that is, lower bounded by any nondecreasing unbounded function of the number of variables), whereas, unless the ETH fails, the problem is not solvable in subexponential time for any constant domain size that is at least 3. For the CSP with *AtLeastNValue* constraints, we show that the problem is solvable in subexponential time if the number of constraints is constant and the domain size is $\omega(1)$, and unless the ETH fails, the problem is not solvable in subexponential time if the number of constraints is linear and the domain size is constant. The results in this paper shed some light on which instances of the aforementioned CSPs with global constraints are feasible with respect to exact computation.

2 Preliminaries

2.1 CSP

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 mapping that assigns each variable $v \in C$ a finite set $D(v)$ of *domain values*, and $\mathcal{C}$ is a finite set of *constraints*. We write $D = \bigcup_{v \in V} D(x)$.

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 we also call such a constraint a *table constraint*. The *size* of a CSP instance $I = (V, D, \mathcal{C})$ is the sum $\sum_{(S,R)\in\mathcal{C}} |S| \cdot |R|$. 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.

Bounding the *treewidth* is a classical method for restricting the structure of CSP instances. The method dates back to Freuder [15]. Treewidth is a graph parameter that can be applied to CSP in terms of *primal graphs* or *incidence graphs* giving rise to the CSP parameters *primal treewidth* (also called *induced width* [11]) and *incidence treewidth*, respectively [35]. For self-containment we give the definitions. 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 is a bipartite graph, one side of which consists of the variables and the other side consists of the constraints; 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 $\chi(t_2) \subseteq \chi(t_1) \cap \chi(t_3)$ 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.

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;
- dom: the number $|D|$ of values;
- cons: the number $|\mathcal{C}|$ of constraints;
- arity: the maximum size of a constraint scope;
- tw: the treewidth of the primal graph of I;

- tw^*: the treewidth of the incidence graph of I.

BOOLEAN CSP denotes the CSP with the *Boolean domain* $\{0, 1\}$. CNF-SAT is the satisfiability problem for propositional formulas in conjunctive normal form (CNF). k-CNF-SAT denotes CNF-SAT restricted to formulas where each clause is of width at most k, i.e., contains at most k literals.

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* [33, 36]. The *Global Constraints Catalogue* [1] 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 [36]. It admits efficient matching based filtering algorithms [31]. 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* [29], *AtLeastNValue* [32], and *AtMostNValue* [2] are widely used in constraint programming [1]. Each such constraint C is associated with an integer $n_C \in \mathbb{N}$. The *NValue* constraint C over a set S of variables is satisfied if the number of distinct values assigned to the variables in S 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 [24]. *cTable* constraints have also been studied under the name *generalized DNF constraints* [6]. 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.

The CSP where all constraints are *AllDifferent* constraints is denoted $\text{CSP}^{\neq}$. This variant of the CSP was studied by Fellows et al. [13] who called it MAD-CSP (multiple all different CSP). The CSP where all constraints are *NValue*, *AtLeastNValue*, or *AtMostNValue* constraints, is denoted $\text{CSP}^{=}$, $\text{CSP}^{\geq}$, and $\text{CSP}^{\leq}$, respectively. The CSP where all constraints are *cTable* constraints is denoted CSP^{c}.

We note that all the problems $\text{CSP}^{\neq}$, $\text{CSP}^{=}$, $\text{CSP}^{\geq}$, $\text{CSP}^{\leq}$, CSP^{c}, are NP-complete. In fact, $\text{CSP}^{\neq}$ (and therefore the more general problems $\text{CSP}^{\geq}$, $\text{CSP}^{\leq}$) is even NP-hard for instances consisting of two constraints only [26], and $\text{CSP}^{\leq}$ is even NP-hard for instances consisting of a single constraint [3]. CSP^{c} is clearly NP-hard as it contains the 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 $\text{CSP}^{\neq}$. Such a combination of several global constraints into a new one has been considered for several different global constraints (see, e.g., [20, 34]).

Guarantees and limits for polynomial-time preprocessing for single *NValue*, *AtLeastNValue*, and *AtMostNValue* constraints have been given by Gaspers and Szeider [18].

The Boolean versions of the above global constraints problems, and the parameters vars, dom, cons, arity, tw, and tw^*, are defined as in the CSP.

2.3 Subexponential Time Complexity

The time complexity functions used in this paper are assumed to be proper complexity functions that are unbounded and nondecreasing.

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 the CSP (resp. CNF-Sat) problem 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 [9, 14], the above definition is equivalent to the following: The CSP (resp. CNF-Sat) problem is solvable in *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. $2^{\varepsilon n}|I|^{O(1)}$). 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 parameter functions defined on instances of Q and Q', respectively. In the case of CSP and CNF-Sat, μ and μ' will be the number of variables in the instances of these problems. A *subexponential time Turing reduction family* (SERF-reduction) [14, 22] 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 $\mu'(I') \leq g(1/\varepsilon)(\mu(I) + \log|I|)$.

Since we focus on the super-polynomial factor in the running time, we will often use the O^* notation, which suppresses the polynomial factor in the input length $|I|$.

The optimization class SNP consists of all search problems expressible by second-order existential formulas whose first-order part is universal [30]. [22] 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 complete for SNP 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 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, from [22, Corollary 1] and from the proof of the Sparsification Lemma [22], [14, Lemma 16.17].

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.*

The ETH has become a standard hypothesis in complexity theory [27].

Remark 1. In this paper, when we consider the CSP with 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 the CSP restricted to all the instances in which the parameter is *upper bounded* by a (prespecified) function that is $\Omega(g(n))$ (resp. $\omega(g(n))$, $O(g(n))$, $o(g(n))$).

3 The Problem 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 2. CSP$^{\neq}$ *can be solved in time $O^*(2^n)$.*

Proof. We reduce the instance $\mathcal{I}$ to an instance of List Coloring. 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 List Coloring is solvable in time $O^*(2^n)$ [4], and hence so is CSP$^{\neq}$. □

Corollary 3. *Let $d(n) = \omega(1)$ be a proper complexity function. The* CSP$^{\neq}$ *restricted 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 CSP$^{\neq}$ restricted to instances in which $\mathsf{dom} \geq d(n)$. By Proposition 2, CSP$^{\neq}$ is solvable in time $O^*(2^n) = O^*(d(n)^{n/\log(d(n))}) \subseteq O^*(\mathsf{dom}^{o(n)})$. □

By Corollary 3, we can focus our investigation of the subexponential time complexity of the problem CSP$^{\neq}$ on instances in which $\mathsf{dom} = O(1) = d$, for some 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 4. *Unless the ETH fails,* CSP$^{\neq}$ *restricted 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 $s(n) = \Theta(n)$, 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 $s()$).

By Lemma 1, 3-3-Sat is not solvable in subexponential time unless ETH fails. The standard polynomial-time reduction from 3-Sat to 3-Colorability (see [10]), 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. Let Linear-3-Colorability be the restriction of 3-Colorability to instances in which the number of edges is linear in the number of vertices. The previous argument shows that if Linear-3-Colorability is solvable in subexponential time then so is 3-3-Sat, and then the ETH would fail. 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 $\mathsf{dom} = 3$. Combining the above sequence of arguments proves the proposition. □

Remark 2. 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 4 and Corollary 3 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 $\mathrm{CSP}^{\neq}$ with respect to the treewidth of the primal graph:

Proposition 5. *$\mathrm{CSP}^{\neq}$ is solvable in subexponential time for instances in which $\mathsf{tw} = o(n)$, and unless the ETH fails, $\mathrm{CSP}^{\neq}$ is not solvable in subexponential time for instances in which $\mathsf{tw} = \Omega(n)$.*

Proof. Let $\mathcal{I}$ be an instance of $\mathrm{CSP}^{\neq}$ 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. It is well known [16] that CSP is solvable in time $O^*(d^{\mathsf{tw}}) \subseteq O^*(d^{o(n)})$, and hence $\mathcal{I}$ can be decided in subexponential time.

The hardness result follows from a general observation about the primal treewidth of the CSP. 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 $\mathrm{CSP}^{\neq}$ 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 $\mathrm{CSP}^{\neq}$ to an instance in which $\mathsf{tw} \leq s(n) = \Omega(n)$. The result now follows from the same result for $\mathrm{CSP}^{\neq}$ on general instances (implied, e.g., from Proposition 4).[1] □

It is well-known that (see [25]) $\mathsf{tw} \leq \mathsf{arity} \cdot (\mathsf{tw}^* - 1)$ and $\mathsf{tw}^* \leq \mathsf{tw} + 1$. If $\mathsf{arity} = O(1)$, then tw and tw^* are within a multiplicative constant from one another. Therefore, from Proposition 5 we can infer the following tight result:

Proposition 6. *$\mathrm{CSP}^{\neq}$ is solvable in subexponential time for instances in which $\mathsf{tw}^* = o(n)$, and unless the ETH fails, $\mathrm{CSP}^{\neq}$ is not solvable in subexponential time for instances in which $\mathsf{tw}^* = \Omega(n)$.*

Remark 3. 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, also the other width parameter is small; but there are instances for which the other width parameter is small but tw^* can be arbitrarily large. Prominent examples for such with parameters are *hypertree width* [19] and *submodular width* [28]. The lower bound statement of Proposition 6 clearly carries over to the more general width parameters. The same holds true for the lower bound statements in Proposition 14 and Theorem 17.

4 The Problems $\mathrm{CSP}^{=}$, $\mathrm{CSP}^{\geq}$, and $\mathrm{CSP}^{\leq}$

We start by presenting an exact algorithm for $\mathrm{CSP}^{\geq}$; we do so by reducing $\mathrm{CSP}^{\geq}$ to $\mathrm{CSP}^{\neq}$. We use the example illustrated in Figure 1 as a running example to explain the idea behind this

[1] This padding argument applies as well to the other variants of the CSP with global constraints considered in this paper, and will prove useful for the hardness results on their subexponential time complexity when $\mathsf{tw} \leq s(n) = \Omega(n)$.

reduction. In this example, the instance $\mathcal{I}$ 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 1.) 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 7, 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 1.) 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

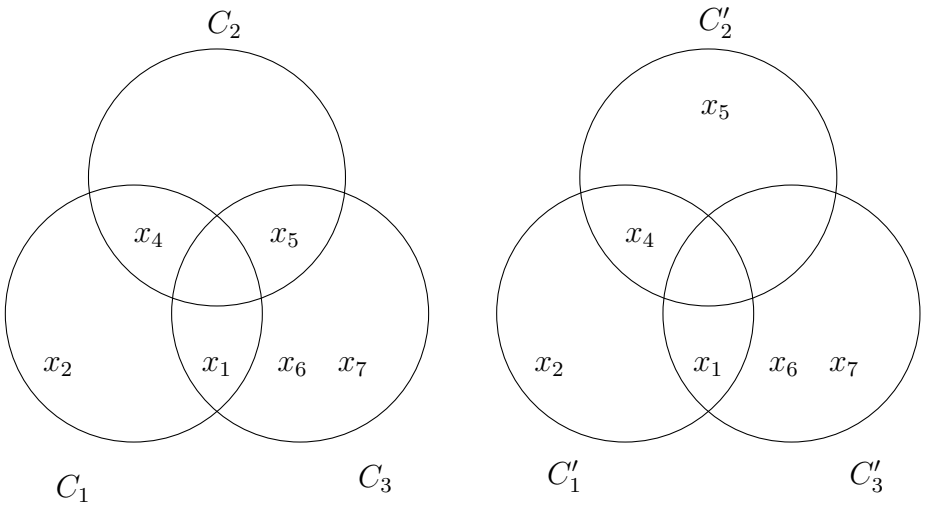


Figure 1: Illustration of the example of the reduction from $\text{CSP}^{\geq}$ to $\text{CSP}^{\neq}$.

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 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 $\text{CSP}^{\geq}$, 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 7 (**The Repartitioning Lemma**). *Let $\mathcal{I}$ be an instance of $\text{CSP}^{\geq}$. 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 8. $\text{CSP}^{\geq}$ *can be solved in time* $O^*((2^{(\mathsf{cons}+1)} + 1)^n)$.

Proof. Let $\mathcal{I}$ be an instance of $\text{CSP}^{\geq}$ 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 7, 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 $\text{CSP}^{\neq}$ on the set of variables X and the set of constraints $C_1', \ldots, C_c'$, and invoke the algorithm for $\text{CSP}^{\neq}$ described in Proposition 2 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 $\text{CSP}^{\neq}$ 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 $\text{CSP}^{\neq}$ 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 7, 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 9. CSP$^{\geq}$ *restricted to instances in which* $\mathsf{cons} = O(1)$ *is solvable in* $O^*(2^{O(n)})$ *time.*

Corollary 10. CSP$^{\geq}$ *restricted to instances in which* $\mathsf{cons} = o(\log \mathsf{dom})$ *is solvable in subexponential time.*

Proof. The result follows from Theorem 8 after noticing that if $\mathsf{cons} = o(\log \mathsf{dom})$ then $2^{\mathsf{cons}} = \mathsf{dom}^{o(1)}$. □

Proposition 11. *Let* $d(n) = \omega(1)$ *be a proper complexity function. Then* CSP$^{\geq}$ *restricted to instances in which* $\mathsf{cons} = O(1)$ *and* $\mathsf{dom} \geq d(n)$ *is solvable in subexponential time, and unless the ETH fails,* CSP$^{\geq}$ *restricted 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 10. The hardness result follows from the hardness result for CSP$^{\neq}$ in Proposition 4 (CSP$^{\neq}$ is a special case of CSP$^{\geq}$). □

Theorem 12. CSP$^{\leq}$ *restricted to instances where* $\mathsf{dom} = O(1)$ *and* $\mathsf{cons} = \Omega(n)$ *is 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. 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 [17] 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 $c_1, \ldots, c_m$, 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 2.

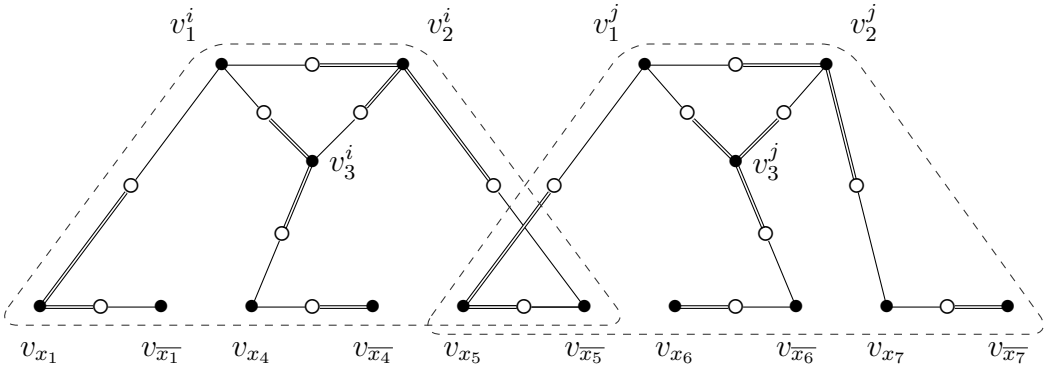


Figure 2: The CSP$^{\leq}$ constraints corresponding to example clauses $c_i = (x_1 \vee x_4 \vee \overline{x_5})$ and $c_j = (x_5 \vee \overline{x_6} \vee x_7)$. Variables are denoted by $\circ$, and values by $\bullet$. The constraints are indicated by dashed lines. The nine variables in each constraint 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 $\mathrm{CSP}^{\leq}$. Consequently, we can identify all m many domain values $v_1^1, \ldots, v_1^m$ into a single value, and similarly identify all domain values $v_2^1, \ldots, v_2^m$ and $v_3^1, \ldots, v_3^m$. 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^p_φ 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^p_φ is bounded above by 8. Then, by Brooks' Theorem [5], we know that there exists a proper coloring of G^p_φ by at most 9 colors, and that such a coloring can be computed in linear time. Take such a proper coloring c of G^p_φ. Now, for each color b used by the coloring c, we let $X_b \subseteq \mathrm{Var}(\varphi)$ be the set of variables x such that $c(x) = b$. Then, since c is a proper coloring of the primal graph G^p_φ 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 $\mathrm{CSP}^{\leq}$, 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 $\mathrm{CSP}^{\leq}$ with $\mathsf{cons} = O(n)$ and $\mathsf{dom} = O(1)$. □

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

Proposition 13. $\mathrm{CSP}^{=}$, $\mathrm{CSP}^{\geq}$, *and* $\mathrm{CSP}^{\leq}$ *restricted to instances in which* $\mathsf{tw} = o(n)$ *are solvable in subexponential time, and unless the ETH fails,* $\mathrm{CSP}^{=}$, $\mathrm{CSP}^{\geq}$, *and* $\mathrm{CSP}^{\leq}$ *restricted to instances in which* $\mathsf{tw} = \Omega(n)$ *are not solvable in subexponential time.*

Proof. The proof of this proposition for each of the $\mathrm{CSP}^{=}$, $\mathrm{CSP}^{\geq}$, and $\mathrm{CSP}^{\leq}$ is exactly the same as the proof of Proposition 5. □

Finally, the following hardness result for $\mathrm{CSP}^{=}$ and $\mathrm{CSP}^{\geq}$ with respect to tw^* follows from Proposition 6 since $\mathrm{CSP}^{\neq}$ is a special case of each of $\mathrm{CSP}^{=}$ and $\mathrm{CSP}^{\geq}$:

Proposition 14. *Unless the ETH fails,* $\mathrm{CSP}^{=}$ *and* $\mathrm{CSP}^{\geq}$ *are not solvable in subexponential time for instances in which* $\mathsf{tw}^* = \Omega(n)$.

5 The Problem CSP^c

We start by providing strong evidence that Boolean CSP^c is not solvable in subexponential time. By SAT[3] we denote the satisfiability of normalized propositional formulas of depth 3 (see [14]), 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 time $O^*(2^{o(n)})$ then the W-hierarchy in parameterized complexity collapses at the second level [7], that is, $W[2] = \mathrm{FPT}$, which is a consequence that is deemed very unlikely and would imply that the ETH fails [14]. We have the following result:

Proposition 15. *Unless* $W[2] = \mathrm{FPT}$, Boolean CSP^c *is not solvable in time* $O^*(2^{o(n)})$.

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 in [7]. □

Next, we consider the subexponential time complexity of CSP^c with respect to the number of constraints cons. We have the following proposition:

Proposition 16. CSP^c *restricted to instances in which* $\mathsf{cons} = O(1)$ *is solvable in subexponential time (even in* P*), and unless the ETH fails,* CSP^c *restricted to instances in which* $\mathsf{cons} = \omega(1)$ *is not solvable in subexponential time.*

Proof. If the number of constraints in an instance is $O(1)$, then in polynomial time we can enumerate each subset of tuples T such that T contains exactly one compressed tuple from each constraint in the instance (because the size of T is $O(1)$). We can then verify consistency, and deduce an instantiation of the set of variables if it exists in polynomial time. The hardness result follows from the same hardness result for CSP [23] since CSP is a special case of CSP^c. □

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

Theorem 17. *The following statements are true:*

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

(ii) CSP^c *restricted to instances in which* $\mathsf{tw}^* = O(1)$ *is solvable in subexponential time (even in* P*), and unless the ETH fails,* CSP^c *restricted 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. It is well known [16] 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 the CSP [23].

(ii) The hardness result is a direct consequence of the hardness result in Proposition 16, 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 [35] 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 in [35] 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 [35, Lemma 2(5)] applies, which is based on a construction due to Kolaitis and Vardi [25], and from which it follows that the treewidth of G^d is at most $2w+2$.

Next we obtain 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., [11, 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 [16]. □

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 15 for the Boolean CSP^c is stronger than that known for Boolean CSP [23], the latter of which states that a (nonuniform) subexponential time algorithm for CSP implies a (nonuniform) subexponential time algorithm for CNF-Sat.

6 Conclusion

We have provided a first analysis of the subexponential time complexity of CSP with global constraints, 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

[1] N. Beldiceanu, M. Carlsson, and J.-X. Rampon. Global constraint catalog. Technical Report T2005:08, SICS, SE-16 429 Kista, Sweden, Aug. 2006. On-line version at `http://www.emn.fr/x-info/sdemasse/gccat/`.

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

[3] C. Bessière, E. Hebrard, B. Hnich, and T. Walsh. The complexity of global constraints. In D. L. McGuinness and G. Ferguson, editors, *Proceedings of the Nineteenth National Conference on Artificial Intelligence, July 25-29, 2004, San Jose, California, USA*, pages 112–117. AAAI Press / The MIT Press, 2004.

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

[5] R. L. Brooks. On colouring the nodes of a network. *Mathematical Proceedings of the Cambridge Philosophical Society*, 37:194–197, 4 1941.

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

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

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

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

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, Cambridge, MA, third edition, 2009.

[11] R. Dechter. *Constraint Processing*. Morgan Kaufmann, 2003.

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

[13] M. R. Fellows, T. Friedrich, D. Hermelin, N. Narodytska, and F. A. Rosamond. Constraint satisfaction problems: Convexity makes alldifferent constraints tractable. In T. Walsh, editor, *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 522–527. IJCAI/AAAI, 2011. Full version appeared in *Theoretical Computer Science*, 472: 81-89 (2013).

[14] J. Flum and M. Grohe. *Parameterized Complexity Theory*, volume XIV of *Texts in Theoretical Computer Science. An EATCS Series*. Springer Verlag, Berlin, 2006.

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

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

[17] M. R. Garey and D. R. Johnson. *Computers and Intractability*. W. H. Freeman and Company, New York, San Francisco, 1979.

[18] S. Gaspers and S. Szeider. Kernels for global constraints. In T. Walsh, editor, *Proceedings of the 22nd International Joint Conference on Artificial Intelligence, IJCAI 2011*, pages 540–545. AAAI Press/IJCAI, 2011.

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

[20] B. Hnich, Z. Kiziltan, and T. Walsh. 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*, 2004.

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

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

[23] I. Kanj and S. Szeider. On the subexponential time complexity of CSP. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*. AAAI Press, 2013.

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

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

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

[27] D. Lokshtanov, D. Marx, and S. Saurabh. Lower bounds based on the exponential time hypothesis. *Bulletin of the European Association for Theoretical Computer Science*, 105:41–72, 2011.

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

[29] F. Pachet and P. Roy. Automatic generation of music programs. In J. Jaffar, editor, *Principles and Practice of Constraint Programming - CP'99, 5th International Conference, Alexandria, Virginia, USA, October 11-14, 1999, Proceedings*, volume 1713 of *Lecture Notes in Computer Science*, pages 331–345. Springer Verlag, 1999.

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

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

[32] J.-C. Régin. *Développement d'outils algorithmiques pour l'Intelligence Artificielle*. PhD thesis, Montpellier II, 1995. in French.

[33] J.-C. Régin. Global constraints: A survey. In P. van Hentenryck and M. Milano, editors, *Hybrid Optimization: The Ten Years of CPAIOR*, volume 45 of *Optimization and Its Applications*, chapter 3, pages 63–134. Springer Verlag, 2011.

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

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

[36] W.-J. van Hoeve and I. Katriel. Global constraints. In F. Rossi, P. van Beek, and T. Walsh, editors, *Handbook of Constraint Programming*, chapter 6. Elsevier, 2006.