Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence

On the Subexponential Time Complexity of CSP

Iyad Kanj[1] and **Stefan Szeider**[2*]
[1]School of Computing, DePaul University, Chicago, USA
ikanj@cs.depaul.edu
[2]Vienna University of Technology, Vienna, Austria
stefan@szeider.net

Abstract

A Constraint Satisfaction Problem (CSP) with n variables ranging over a domain of d values can be solved by brute-force in d^n steps (omitting a polynomial factor). With a more careful approach, this trivial upper bound can be improved for certain natural restrictions of the CSP. In this paper we establish theoretical limits to such improvements, and draw a detailed landscape of the subexponential-time complexity of CSP.

We first establish relations between the subexponential-time complexity of CSP and that of other problems, including CNF-SAT. We exploit this connection to provide tight characterizations of the subexponential-time complexity of CSP under common assumptions in complexity theory. For several natural CSP parameters, we obtain threshold functions that precisely dictate the subexponential-time complexity of CSP with respect to the parameters under consideration.

Our analysis provides fundamental results indicating *whether and when* one can significantly improve on the brute-force search approach for solving CSP.

Introduction

The Constraint Satisfaction Problems (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, and Walsh 2006). For instance, in database theory, the CSP is equivalent to the evaluation problem of conjunctive queries on relational databases (Gottlob, Leone, and 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 (Feder and Motwani 2002; Beigel and Eppstein 2005; Grandoni and Italiano 2006), in particular, for certain restrictions of CSP. For instance, binary CSP with domain size d can now be solved in time $(d-1)^n$ (omitting a polynomial factor in the input size) by a forward-checking algorithm employing a fail-first variable ordering heuristic (Razgon 2006). 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 the CSP. We note that the study of the existence of subexponential-time algorithms is of prime interest, as a subexponential-time algorithm for a problem would allow us to solve larger hard instances of the problem in comparison to an exponential-time algorithm.

Results We obtain lower and upper bounds and draw a detailed complexity landscape of CSP with respect to subexponential-time solvability. Our lower bounds are subject to (variants of) the *Exponential Time Hypothesis* (ETH), proposed by Impagliazzo and Paturi (2001), which states that 3-CNF-SAT has no subexponential-time algorithm.

It is easy to see that CSP of *bounded domain size* (i.e., the maximum number of values for each variable) and *bounded arity* (i.e., the maximum number of variables that appear together in a constraint) 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):

1. If BOOLEAN CSP is solvable in nonuniform subexponential time then so is (unrestricted) CNF-SAT.
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 $\Omega(n)$.

*Supported by the European Research Council (ERC), project COMPLEX REASON 239962.

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.

By a classical result of (Freuder 1990), CSP becomes easier if the instance has *small treewidth*. There are several ways of measuring the treewidth of a CSP instance, depending on the graph used to model the structure of the instance. The most common models are the primal graph and the incidence graph. The former has as vertices the variables of the CSP instance, and two variables are adjacent if they appear together in a constraint. The incidence graph is the bipartite graph on the variables and constraints, where a variable is incident to all the constraints in which it is involved. We show that the treewidth of these two graph models give rise to different subexponential-time complexities:

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 $\Omega(n)$, assuming the ETH.
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.

Our tight results, summarized in the table at the end of this 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 SAT and CSP, an important topic of recent AI research (Jeavons and Petke 2012; Dimopoulos and Stergiou 2006; Benhamou, Paris, and Siegel 2012; Bennaceur 2004).

Preliminaries

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 the 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.

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;
- dom: the number $|D|$ of values;
- cons: the number $|\mathcal{C}|$ of constraints;

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.

Subexponential time The time complexity functions used in this paper are assumed to be proper complexity functions that are unbounded and nondecreasing. The $o(\cdot)$ notation used denotes the $o^{\text{eff}}(\cdot)$ notation (Flum and Grohe 2006). More formally, for any two computable functions $f, g : \mathbb{N} \to \mathbb{N}$, by writing $f(n) = o(g(n))$ we mean that there exists a computable nondecreasing unbounded 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 $n \geq n_0$.

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 (Chen, Kanj, and Xia 2009; Flum and Grohe 2006), the above definition is equivalent to the following: The CSP (resp. CNF-SAT) problem 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. $2^{\varepsilon n} |I|^{O(1)}$). The CSP (resp. CNF-SAT) problem 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 subexponential-time algorithms running in $O(2^{\sqrt{n}})$ time do exist for many natural problems (Alber, Fernau, and Niedermeier 2004).

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* (Impagliazzo, Paturi, and Zane 2001; Flum and Grohe 2006), shortly a *serf-reduction*, 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 and 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 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 input formula. Therefore, there exists $c > 0$ such that k-CNF-SAT cannot be solved in time 2^{cn}.

The following result is implied from (Impagliazzo, Paturi, and Zane 2001, Corollary 1) and from the proof of the Sparsification Lemma (Impagliazzo, Paturi, and Zane 2001), (Flum and Grohe 2006, 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 upper bounded by a constant 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).*

The ETH has become a standard hypothesis in complexity theory (Lokshtanov, Marx, and Saurabh 2011).

We close this section by mentioning some further work on the subexponential-time complexity of CSP. There are several results on 2-CSP with bounds on tw, the treewidth of the primal graph (see the Introduction section for definitions). Lokshtanov, Marx, and Saurabh (2011) showed the following lower bound, using a result on list coloring (Fellows et al. 2011): 2-CSP cannot be solved in time $f(\mathsf{tw})n^{o(\mathsf{tw})}$ unless the ETH fails. Marx (2010a) 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. Traxler (2008) studied the subexponential-time complexity of CSP where the constraints are represented by listing the forbidden tuples (in contrast to the standard representation that we use, where the allowed tuples are given, and which naturally captures database problems (Gottlob, Leone, and Scarcello 2002; Grohe 2006; Papadimitriou and Yannakakis 1999)). This setting can be considered as a generalisation of CNF-SAT; a single clause gives rise to a constraint with exactly one forbidden tuple.

Relations between CSP and CNF-SAT

In this section, we investigate the relation between the subexponential-time complexity of CSP and that of CNF-SAT. A clause of constant width can be represented by a constraint of constant arity; the reverse holds as well (we get a constant number of clauses). Hence, we have:

Proposition 1. BOOLEAN r-CSP *is solvable in subexponential time if and only if the ETH fails.*

The following proposition suggests that Proposition 1 may not extend to r-CSP with unbounded domain size. Chen et al. (Chen et al. 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 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 skip the proof, and refer the reader to the original source (Papadimitriou and Yannakakis 1999).

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

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 above.

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 $l_1, \ldots, l_k$ 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. 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.

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 $2^{\varepsilon n} m^{O(1)}$.

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. □

Instance size and number of tuples

In this section we give characterizations of the subexponential-time complexity of CSP with respect to the instance size and the number of tuples. Recall that the size of an instance $I = (V, D, \mathcal{C})$ of CSP is $\mathsf{size} = \sum_{(S,R)\in\mathcal{C}} |S| \cdot |R|$. We also show that the subexponential-time solvability of BOOLEAN CSP with linear size, or linear number of tuples, is equivalent to the statement that the ETH fails.

Lemma 2. *Unless the ETH fails,* BOOLEAN CSP *is not solvable in subexponential-time if the instance size is* $\Omega(n)$.

Proof. Let $s(n) = \Omega(n) \geq cn$ be a complexity function, where $c > 0$ is a constant. Suppose that the restriction of 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, we can prove the preceding statement assuming any linear upper bound on the number of clauses; 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 $2^{o(n)} n^{O(1)} = 2^{o(n)}$. □

Lemma 3. CSP *restricted to instances with* $o(n)$ *tuples is solvable in subexponential-time.*

Proof. Let $s(n) = o(n)$ be a complexity function, and consider the restriction of CSP to instances with at most $s(n)$ tuples. We will show that this problem is solvable in time $\mathsf{dom}^{s(n)}|I|^{O(1)}$. Consider the algorithm A that, for each tuple in a constraint, branches on whether or not the tuple is satisfied by the satisfying assignment sought. A branch in which more than one tuple in any constraint is selected as satisfied is rejected, and likewise for a branch in which no tuple in a constraint is selected. For each remaining branch, the algorithm checks if the assignment to the variables stipulated by the branch is consistent. If it is, the algorithm accepts; the algorithm rejects if no branch corresponds to a consistent assignment. Clearly, the algorithm A is correct, and runs in time $2^{s(n)}|I|^{O(1)} = \mathsf{dom}^{s(n)}|I|^{O(1)}$. □

Noting that the number of tuples is a lower bound for the instance size, the following theorem follow from Lemma 2 and Lemma 3:

Theorem 2. 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 $\Omega(n)$.*

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 lemma.

Lemma 4. *If the ETH fails then* BOOLEAN CSP *with linear number of tuples is solvable in subexponential time.*

Proof. We give a serf-reduction from BOOLEAN CSP with linear number of tuples to BOOLEAN r-CSP for some constant $r \geq 3$ to be specified below. The statement will then follow from Proposition 1.

Let $s(n) \leq cn$ be a complexity function, where $c > 0$ is a constant. Consider the restriction of BOOLEAN CSP to instances in which the number of tuples is at most cn; we will refer to this problem as BOOLEAN LINEAR TUPLE CSP. Let $0 < \varepsilon < 1$ be given. Choose a positive integer-constant d large enough so that the unique root of the polynomial $x^d - x^{d-1} - 1$ in the interval $(1, \infty)$ is at most $2^{\varepsilon/c}$. (The uniqueness of the root was shown (Chen, Kanj, and Jia 2001, Lemma 4.1), and the fact that the root converges to 1 as $d \longrightarrow \infty$ can be easily verified.) Let I be an instance of BOOLEAN LINEAR TUPLE CSP. We will assume that, for any constraint C in I, and any two variables x, y in C, there must be at least one tuple in C in which the values of x and y differ. If not, then the values of x and y in any assignment that makes I consistent have to be the same; in this case we remove all tuples from I in which the values of x and y differ, replace y with x in every constraint in I, and simplify I accordingly (if a constraint becomes empty during the above process then we reject I).

We now apply the following branching procedure to I. For each constraint C in I with more than d tuples, pick a tuple t in C and branch on whether or not t is satisfied in an assignment that makes I consistent (if such an assignment exists). In the branch where t is satisfied, remove C from I, remove every tuple in I in which the value of a variable that appears in C does not conform to the value of the variable in t, and finally remove all variables in C from I and its tuples (if a constraint becomes empty reject I). In the branch where t is not satisfied, remove t from C. Note that each branch either removes a tuple or removes at least d tuples. We repeat the above branching until each constraint in the resulting instance contains at most d tuples. The above branching can be depicted by a binary search tree whose leaves correspond to all the possible outcomes from the above branching. The number of the leaves in the search tree is $O(x_0^{cn})$, where x_0 is the root of the polynomial $x^d - x^{d-1} - 1$ in the interval $(1, \infty)$. (The branching vector is not worse than $(1, d)$.) By the choice of d, the number of leaves in the search tree is $O(2^{\varepsilon n})$. Let I' be the resulting instance at a leaf of the search tree. We claim that the arity of I' is at most 2^d. Suppose not, and let C be a constraint in I' whose arity is more than 2^d. Pick an arbitrary ordering of the tuples in C, and list them as $t_1, \ldots, t_s$, where $s \leq d$. For each variable in C, we associate a binary sequence of length s whose ith bit is the value of the variable in t_i. Since the arity is more than 2^d, the number of binary sequences is more than 2^d. Since the length of each sequence is $s \leq d$, by the pigeon-hole principal, there exist two binary sequences that are identical. This contradicts our assumption that no constraint has two variables whose values are identical in all the tuples of the constraint. It follows that the instance I' is an instance of BOOLEAN 2^d-CSP. Since the number of variables in I' is at most that of I, and the number of leaves in the search tree is $O(2^{\varepsilon n})$, we have a serf-reduction from BOOLEAN LINEAR TUPLE CSP to BOOLEAN r-CSP for some constant r. □

Lemma 2, combined with Lemma 4 after noting that the size is an upper bound on the number of tuples, give the following result.

Theorem 3. BOOLEAN CSP *with linear number of tuples is solvable in subexponential time if and only if the ETH fails.*

Theorem 4. *The* BOOLEAN CSP *with linear size is solvable in subexponential time if and only if the ETH fails.*

Treewidth and number of constraints

In this section we characterize the subexponential-time complexity of CSP with respect to the *treewidth* of certain graphs that model the interaction of variables and constraints. Many NP-hard problems on graphs become polynomial-time solvable for graphs whose treewidth is bounded by a constant. For a definition of treewidth we refer to other sources (Bodlaender 1998). Freuder (1990) showed that CSP is polynomial-time solvable if a certain graph associated with the instance, the *primal graph*, is of bounded treewidth. The primal graph associated with a CSP instance I has the variables in I as its vertices; two variables are joined by an edge if and only if they occur together in the scope of a constraint. Freuder's result was generalized in various ways, and other restrictions on the graph structure of CSP instances have been considered (Gottlob, Leone, and Scarcello 2000; Marx 2010b). If the treewidth of the primal graph is bounded, then so is the arity of the constraints. The *incidence graph* provides a more general graph model, as it includes instances of unbounded arity even if the treewidth is bounded. The incidence graph associated with I is a bipartite graph with one partition being the set of variables in I and the other partition being the set of constraints in I; a variable and a constraint are joined by an edge if and only if the variable occurs in the scope of the constraint. For a CSP instance, we denote by tw the treewidth of its primal graph and by tw^* the treewidth of its incidence graph.

As shown by Bodlaender (1996), there exists for every fixed k a linear time algorithm that checks if a graph has treewidth at most k and, if so, outputs a tree decomposition of minimum width. It follows that we can check whether the treewidth of a graph is $O(1)$ in polynomial time.

Lemma 5. CSP *is solvable in polynomial time for instances whose incidence treewidth* tw^* *is* $O(1)$.

Proof. If the tw^* is $O(1)$ then the *hypertree-width* is also $O(1)$ (Gottlob, Leone, and Scarcello 2000), and CSP is solv-

able in polynomial-time if the hypertree-width is $O(1)$ (Gottlob, Leone, and Scarcello 2002). Combining the preceding statements gives the lemma. □

Lemma 6. *Unless the ETH fails,* CSP *is not solvable in subexponential-time if the number of constraints is* $\omega(1)$.

Proof. Let $\lambda(n) = \omega(1)$ be a complexity function. We show that, unless the ETH fails, the restriction of CSP to instances in which $\mathsf{cons} \leq \lambda(n)$, denoted CSP_λ 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_λ.

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_λ. 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_λ. □

Since $\mathsf{tw}^* = O(\mathsf{cons})$, Lemmas 5 and 6 give the following result.

Theorem 5. CSP *is solvable in polynomial time for instances with* $O(1)$ *constraints, and unless the ETH fails, is not solvable in subexponential-time if the number of constraints is* $\omega(1)$.

Theorem 6. CSP *is solvable in polynomial time for instances whose incidence treewidth* tw^* *is* $O(1)$*, and unless the ETH fails, is not solvable in subexponential-time for instances whose* tw^* *is* $\omega(1)$.

Theorem 7. CSP *is solvable in subexponential-time for instances whose primal treewidth* tw *is* $o(n)$*, and is not solvable in subexponential-time for instances whose* tw *is* $\Omega(n)$ *unless (the general)* CSP *is solvable in subexponential time.*

Proof. The fact that CSP is solvable in subexponential time if $\mathsf{tw} = o(n)$ follows from the facts that: (1) 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 10), and (2) CSP is solvable in time $O(\mathsf{dom}^{\mathsf{tw}})|I|^{O(n)}$ (Freuder 1990).

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 $\mathsf{tw} \leq n$. Therefore, if $c \geq 1$, 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 "ETH fails" by Proposition 1, and implies that CNF-SAT has a nonuniform subexponential-time algorithm by Theorem 1. We also note that the difference between the subexponential-time complexity of CSP with respect to the two structural parameters tw and tw^*: Whereas the threshold function for the subexponential-time solvability of CSP with respect to tw is $o(n)$, the threshold function with respect to tw^* is $O(1)$.

Degree and arity

In this section we give characterizations of the subexponential-time complexity of CSP with respect to the degree and the arity. The proofs are omitted.

Theorem 8. *Unless ETH fails,* CSP *is not solvable in subexponential-time if* $\mathsf{deg} \geq 2$.

There is a folklore reduction from an instance of 3-COLORABILITY with n vertices that results in an instance of CSP with n variables, $\mathsf{arity} = 2$, and $\mathsf{dom} = 3$. Since the 3-COLORABILITY problem is SNP-complete under serf-reductions (Impagliazzo, Paturi, and Zane 2001), we get:

Theorem 9. *Unless ETH fails,* CSP *is not solvable in subexponential-time if* $\mathsf{arity} \geq 2$ *(and* $\mathsf{dom} \geq 3$*).*

Conclusion

We have provided a first analysis of the subexponential-time complexity of CSP under various restrictions. We have obtained several tight thresholds that dictate the subexponential-time complexity of CSP. These tight results are summarized in the following table.

CSP ∈ SUBEXP	CSP ∉ SUBEXP (assuming the ETH)	Result
$\mathsf{tuples} \in o(n)$	$\mathsf{tuples} \in \Omega(n)$	Theorem 2
$\mathsf{cons} \in O(1)$ (even in P)	$\mathsf{cons} \in \omega(1)$	Theorem 5
$\mathsf{tw}^* \in O(1)$ (even in P)	$\mathsf{tw}^* \in \omega(1)$	Theorem 6
$\mathsf{tw} \in o(n)$	$\mathsf{tw} \in \Omega(n)$	Theorem 7

Furthermore, we have linked the subexponential-time complexity of CSP with bounded arity to CLIQUE, and CSP with bounded domain size to CNF-SAT. These results suggest that these restrictions of CSP may be "harder than" k-CNF-SAT—for which a subexponential-time algorithm would lead to the failure of the ETH—with respect to subexponential-time complexity. It would be interesting to provide stronger theoretical evidence for this separation.

References

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

Amir, E. 2010. Approximation Algorithms for Treewidth. *Algorithmica* 56(4):448–479.

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

Benhamou, B.; Paris, L.; and 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.

Bodlaender, H. L. 1996. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.* 25(6):1305–1317.

Bodlaender, H. L. 1998. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science* 209(1-2):1–45.

Calabro, C.; Impagliazzo, R.; and Paturi, R. 2006. A duality between clause width and clause density for SAT. In *21st Annual IEEE Conference on Computational Complexity, CCC 2006*, 252–260. IEEE Computer Society.

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

Chen, J.; Kanj, I. A.; and Jia, W. 2001. Vertex cover: further observations and further improvements. *J. Algorithms* 41(2):280–301.

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

Dimopoulos, Y., and Stergiou, K. 2006. Propagation in CSP and SAT. In Benhamou, F., ed., *Principles and Practice of Constraint Programming - CP 2006, Proceedings*, volume 4204 of *Lecture Notes in Computer Science*, 137–151. Springer Verlag.

Feder, T., and 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.; and Thomassen, C. 2011. On the complexity of some colorful problems parameterized by treewidth. *Information and Computation* 209(2):143–153.

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

Freuder, E. C. 1990. Complexity of k-tree structured constraint satisfaction problems. In Shrobe, H. E.; Dietterich, T. G.; and Swartout, W. R., eds., *Proceedings of the 8th National Conference on Artificial Intelligence.*, 4–9.

Gottlob, G.; Leone, N.; and Scarcello, F. 2000. A comparison of structural CSP decomposition methods. *Artificial Intelligence* 124(2):243–282.

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

Grandoni, F., and Italiano, G. F. 2006. Algorithms and constraint programming. In Benhamou, F., ed., *Principles and Practice of Constraint Programming - CP 2006, Proceedings*, volume 4204 of *Lecture Notes in Computer Science*, 2–14. Springer Verlag.

Grohe, M. 2006. The structure of tractable constraint satisfaction problems. In Kralovic, R., and Urzyczyn, P., eds., *Mathematical Foundations of Computer Science 2006, MFCS 2006, Proceedings*, volume 4162 of *Lecture Notes in Computer Science*, 58–72. Springer Verlag.

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

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

Jeavons, P., and Petke, J. 2012. Local consistency and sat-solvers. *J. Artif. Intell. Res.* 43:329–351.

Lokshtanov, D.; Marx, D.; and 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. 2010a. Can you beat treewidth? *Theory of Computing* 6:85–112.

Marx, D. 2010b. Tractable hypergraph properties for constraint satisfaction and conjunctive queries. In Schulman, L. J., ed., *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, 735–744. ACM.

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

Papadimitriou, C. H., and 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.; and Rossi, F., eds., *Recent Advances in Constraints, CSCLP 2005, Revised Selected and Invited Papers*, volume 3978 of *Lecture Notes in Computer Science*, 88–99. Springer Verlag.

Rossi, F.; van Beek, P.; and Walsh, T., eds. 2006. *Handbook of Constraint Programming*. Elsevier.

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., and Niedermeier, R., eds., *Proceedings of the Third International Workshop on Parameterized and Exact Computation 2008*, volume 5018 of *Lecture Notes in Computer Science*, 190–201. Springer Verlag.