The Time Complexity of Constraint Satisfaction

Patrick Traxler*

Institute of Theoretical Computer Science, ETH Zürich, Switzerland.
patrick.traxler@inf.ethz.ch

Abstract. We study the time complexity of (d, k)-CSP, the problem of deciding satisfiability of a constraint system $\mathcal{C}$ with n variables, domain size d, and at most k variables per constraint. We are interested in the question how the domain size d influences the complexity of deciding satisfiability. We show, assuming the Exponential Time Hypothesis, that two special cases, namely $(d, 2)$-CSP with bounded variable frequency and d-UNIQUE-CSP, already require exponential time $\Omega(d^{c \cdot n})$ for some $c > 0$ independent of d. UNIQUE-CSP is the special case for which it is guaranteed that every input constraint system has at most 1 satisfying assignment.

1 Introduction

In this work we study the time complexity of the NP-complete Constraint Satisfaction Problem (CSP). We are interested in the following question: *What makes CSP hard to solve?* Besides being NP-hard – already $(3, 2)$-CSP and $(2, 3)$-CSP are NP-hard – many algorithms and heuristics for CSP slow down with increasing domain size d. It is however not clear that CSP effectively becomes harder with increasing d.

A promising result [10, 1] is that we can solve d-COL, the d-Graph Colorability Problem, in time $2^{\widetilde{n}} \cdot \text{poly}(\text{input-size})$, where $\widetilde{n}$ is the number of vertices of the input graph. Such a result is however not known for (d, k)-CSP or the special cases $(d, 2, 3d^2)$-FREQ-CSP and d-UNIQUE-CSP.

- (d, k, f)-FREQ-CSP is the (d, k)-CSP for which every input constraint system has maximum variable frequency f.
- (d, k)-UNIQUE-CSP is the (d, k)-CSP for which every input constraint system is guaranteed to have at most 1 satisfying assignment. Without any restriction on the constraint size we have d-UNIQUE-CSP.

We provide precise definitions in Section 2. We now introduce some definitions to state our results. We call an algorithm a $2^{c \cdot n}$-randomized algorithm iff its running time is bounded by $2^{c \cdot n} \cdot \text{poly}(\text{input-size})$ and its error probability is at most $1/3$. Let

$$c_{d,k} := \inf\{c : \exists 2^{c \cdot n}\text{-randomized algorithm for } (d, k)\text{-CSP}\}.$$

* This work was supported by the Swiss National Science Foundation SNF under project 200021-118001/1.

Define $c^{\mathrm{FQ}}_{d,k,f}$ and $c^{\mathrm{UQ}}_{d,k}$ analogously for (d,k,f)-FREQ-CSP and (d,k)-UNIQUE-CSP. Let $c_{d,\infty} := \lim_{k\to\infty} c_{d,k}$.

The variant of the *Exponential Time Hypothesis* (ETH) we assume here states that $c_{2,3} > 0$, i.e., 3-SAT is exponentially hard. It is straight forward to apply the results from [7] to show that $c_{3,2} > 0$ iff $c_{2,3} > 0$. In this work we improve on the lower bound $c_{d,2} > 0$, assuming ETH.

Theorem 1. *If ETH holds, there exists $c > 0$ such that for all $d \geq 3$*

$$c \cdot \log(d) \leq c^{FQ}_{d,2,3d^2}$$

(where c depends on $c_{3,2}$.)

Theorem 1 strongly contrasts the time complexity of d-COL for which we know a $2^{\widetilde{n}}$-algorithm [10, 1]. Such an algorithm is however unlikely to exist for $(d,2,3d^2)$-FREQ-CSP because its existence implies that ETH fails.

The second special case of (d,k)-CSP we study is d-UNIQUE-CSP.

Theorem 2. *For all $d \geq 2$, it holds that*

$$c_{2,\infty} \cdot \lfloor \log(d) \rfloor \leq c^{UQ}_{d,\infty}.$$

Theorem 2 roughly says that the unique case is already the hardest one. Note that the currently best upper bound for $c_{2,\infty}$ is 1.

Motivation. The motivation for our results comes from the design and analysis of exponential time algorithms. We usually fix some natural parameter like the number of variables n and try to find some small c such that we can solve CSP in time $O(c^n)$. The best known upper bound $(d(1-1/k)+\varepsilon)^n \cdot \mathrm{poly}(\text{input-size})$, $\varepsilon > 0$, for (d,k)-CSP [12] is achieved by Schöning's algorithm and it was improved to, omitting the polynomial factor, $d!^{n/d}$ for $(d,2)$-CSP [4], to 1.8072^n for $(4,2)$-CSP [3], and to 1.3645^n for $(3,2)$-CSP [3]. The problem of maximizing the number of satisfied constraints of a $(d,2)$-constraint system is considered in [15]. Our results say that the dependency on d of these algorithms comes close to the best possible.

Studying the special case $(d,2,3d^2)$-FREQ-CSP is motivated by the observation that algorithms for CSP are also analyzed w.r.t. to the number of constraints m instead of n. This is in particular the case if optimization variants of CSP are considered. See [13] for such an algorithm and also for further references. A $(d,2)$-constraint system in which every variable has maximum frequency $3d^2$ has at most $3d^2n$ constraints. Our results imply therefore limitations of algorithms which are analyzed w.r.t. m (Corollary 2).

The second special case we study, d-UNIQUE-CSP, is motivated by the use of randomness. The expected running for finding a satisfying assignment of a constraint system with $s > 0$ satisfying assignments is roughly d^n/s. A considerable improvement, namely $(2^n/s)^{1-1/k}$, exists for k-SAT [2]. It also seems likely that the algorithms in [12, 4] become faster if many satisfying assignments

are present. Our results say that d-UNIQUE-CSP still has increasing complexity w.r.t. to d and that CSP can only become easier if s is large enough. The observed dependency of randomized algorithms on d and s seem therefore to be unavoidable.

Related Work. This work builds upon a series of papers [6, 7, 2] which mainly deal with SAT and special cases of SAT like k-SAT. A central question is: What makes SAT hard to solve? This question is motivated by the observation that many algorithms and heuristics for SAT work better on instances with special properties. For example, there exists a 1.324^n-randomized algorithm for 3-SAT [8], whereas the best algorithms for SAT still take 2^n steps in the worst case. In [6] it was shown, assuming ETH, that for every k there exists $k' > k$ such that $c_{2,k} < c_{2,k'}$. In other words, k-SAT becomes harder with increasing k. Our results, Theorem 1 and 2, are of the same kind. It is however not clear how to adapt the techniques in [6] to our problem. In particular, Impagliazzo & Paturi [6] ask if a similar result as theirs holds for d-COL. Our approach is indeed different from theirs. They use the concept of a forced variable whereas we work with a different technique of partitioning variables (see Lemma 1).

In [9] the exponentially hard instances of the Maximum Independent Set (MIS) problem with respect to the maximum degree were identified. It was shown that if there exists a subexponential time algorithm for MIS with maximum degree 3, then there exists one for MIS (which would contradict ETH). One part of the proof of this theorem is a sparsification lemma for MIS. In the proof of our Lemma 2, a sparsification lemma for $(d, 2)$-CSP, we apply the same technique as there. We prove a new sparsification lemma for $(d, 2)$-constraint systems because we want good bounds. The sparsification lemma in [7] could also be used. But it gives much worse bounds.

Calabro et al. [2] proved that $c_{2,k} \leq c_{2,k}^{\mathrm{UQ}} + O(\log^2(k)/k)$ (Lemma 5 of [2]). We generalize and improve this to $c_{d,k} \leq c_{d,k}^{\mathrm{UQ}} + O(\log(dk)/k)$ (see Section 4). Calabro et al. [2] concluded $c_{2,\infty}^{\mathrm{UQ}} = c_{2,\infty}$ from their result. This theorem generalizes to $c_{d,\infty}^{\mathrm{UQ}} = c_{d,\infty}$ by the previous relation. We remark that our proof is different from theirs although the dependency on k is similar. Calabro et al. adapt the isolation lemma from [14] whereas we build upon the isolation lemma from [11]. In particular, we need a new idea to apply a generalization of the isolation lemma from [11] in our situation (see Lemma 4).

Overview of Work. In Section 2 we introduce the constraint satisfaction problem we study in this work. In Section 3 we prove Theorem 1 and in Section 4 we prove Theorem 2.

2 Preliminaries

A (d, k)-*constraint system* $\mathcal{C}$ consists of a set of values Σ with $|\Sigma| = d$, called the *domain* of $\mathcal{C}$, and a set of constraints of the form

$$C := \{x_1 \neq s_1, x_2 \neq s_2, ...\}$$

with $|C| \leq k$, x_i being some variable and $s_i \in \Sigma$. We often identify $\mathcal{C}$ with the set of constraints and denote by $\mathrm{Dom}(\mathcal{C})$ the associated domain Σ. Let $\mathrm{Var}(\mathcal{C})$ denote the set of variables occurring in $\mathcal{C}$. Unless stated otherwise $n := |\mathrm{Var}(\mathcal{C})|$. We call a mapping $a : \mathrm{Var}(\mathcal{C}) \to \mathrm{Dom}(\mathcal{C})$ an *assignment*. A constraint $C \in \mathcal{C}$ is *satisfied* by an assignment a iff there exists some $(x \neq s) \in C$ such that $a(x) \neq s$. A constraint system $\mathcal{C}$ is *satisfied* iff every $C \in \mathcal{C}$ is satisfied. We denote by $\mathrm{Sat}(\mathcal{C})$ the set of all satisfying assignments of $\mathcal{C}$. The *Constraint Satisfaction Problem* (d, k)-CSP is the problem of deciding if a satisfying assignment for a given (d, k)-constraint system exists. The (d, k, f)-FREQ-CSP is the special case of (d, k)-CSP with maximum variable frequency f, that is, we require that every variable occurs at most f times in an input (d, k)-constraint system, and (d, k)-UNIQUE-CSP is the special case for which an input (d, k)-constraint system is guaranteed to have at most 1 satisfying assignment.

The $(2, k)$-CSP is the k-Boolean Satisfiability Problem (k-SAT). The $(d, 2)$-CSP is a generalization of the d-Graph Colorability Problem (d-COL). For seeing this, consider the following example.

Example 1. Let $G = (U, E)$ be a graph. For $\{u, v\} \in E$ the constraints

$$\{u \neq 1, v \neq 1\}, \{u \neq 2, v \neq 2\}, ..., \{u \neq d, v \neq d\}$$

are in $\mathcal{C}$. Set $\mathrm{Dom}(\mathcal{C}) := \{1, ..., d\}$. Then $\mathcal{C}$ is satisfiable iff G is d-colorable. □

3 Binary Sparse CSP (Proof of Theorem 1)

The proof of Theorem 1 consists of three steps. We show first how to reduce the number of variables by increasing the domain size (Lemma 1). We need this lemma to provide a relation between (d, k)-CSP and (d', k)-CSP for d' larger than d. This is the core of our result that CSP has increasing complexity w.r.t. d. Then, we show in the second step how to transform a $(d, 2)$-constraint system into a sparse $(d, 2)$-constraint system, i.e., we prove a sparsification lemma for $(d, 2)$-constraint systems (Lemma 2). Our transformation can be carried out in subexponential time. Combining both lemmas we are finally able to prove Theorem 1.

At the end of this section we point out how our result relates to algorithms which are analyzed w.r.t. the number of constraints.

Lemma 1. *Let $r \in \mathbb{N}$, $r > 0$. For every (d, k)-constraint system $\mathcal{C}$ over n variables there exists a satisfiability equivalent (d^r, k)-constraint system $\mathcal{C}'$ over $n' := \lceil \frac{n}{r} \rceil$ variables which is computable in time $d^{rk} \cdot \mathrm{poly}(|\mathcal{C}|)$.*

Proof. The idea of our algorithm is to group the variables in groups of size r and replace every group of variables by a new variable. We need the following definition. Let $U \subseteq \mathrm{Var}(\mathcal{C})$ and D be a constraint. Define $\mathrm{Nonsat}_U(D)$ to be the set of all assignments $a : U \to \mathrm{Dom}(\mathcal{C})$ which do not satisfy D. Our algorithm gets as input a (d, k)-constraint system $\mathcal{C}$ and outputs a (d^r, k)-constraint system

$\mathcal{C}'$. It works as follows.

Compute a partition of pairwise disjoint subsets $P_1, ..., P_t$ of $\mathrm{Var}(\mathcal{C})$ such that $|P_i| = r$ for $1 \leq i < t$ and $1 < |P_t| \leq r$. Extend P_t with new variables such that $|P_t| = r$. Find new variables $y_1, ..., y_t$, i.e., variables which are not in $\mathrm{Var}(\mathcal{C})$. Set $\mathcal{C}' \leftarrow \mathcal{C}$ and $\mathrm{Dom}(\mathcal{C}') \leftarrow \mathrm{Dom}(\mathcal{C})^r$, i.e., $\mathrm{Dom}(\mathcal{C}')$ is the set of all strings of length r with symbols from $\mathrm{Dom}(\mathcal{C})$. For every $C \in \mathcal{C}'$ and every $1 \leq i \leq t$: if $\mathrm{Var}(C) \cap P_i \neq \{\}$ then replace all the variables of $\mathrm{Var}(C) \cap P_i$ in C in the following way. Let $D \subseteq C$ be the set of all inequalities with variables from $\mathrm{Var}(C) \cap P_i$. Add the constraint $C' \leftarrow (C \setminus D) \cup \{y_i \neq b\}$ to $\mathcal{C}'$ for every $b \in \mathrm{Nonsat}_{P_i}(D)$. Remove C from $\mathcal{C}'$.

We claim that $\mathcal{C}$ is satisfiable iff $\mathcal{C}'$ is satisfiable. Let $a \in \mathrm{Sat}(\mathcal{C})$. For y_i we define $a'(y_i) := a(x'_1) \cdot ... \cdot a(x'_r)$ with $\{x'_1, ..., x'_r\} = P_i$, i.e., $a'(y_i)$ is the concatenation of the values of variables in P_i. Let $C' \in \mathcal{C}'$ and $C \in \mathcal{C}$ be the corresponding constraint C' emerged from. Since a satisfies C there exists some inequality $x \neq s \in C$ satisfied by a, i.e., $a(x) \neq s$. Assume $x \in P_i$. Then $x \neq s \in D$, D as in the algorithm. This implies that $y_i \neq b \in C'$ for some $b \in \mathrm{Nonsat}_{P_i}(D)$. Since $a'(y_i) \neq b$ for all $b \in \mathrm{Nonsat}_{P_i}(D)$ (because of $a(x) \neq s$) it follows that C' is satisfied by a' and therefore $a' \in \mathrm{Sat}(\mathcal{C}')$.

For the other direction, assume that $a \notin \mathrm{Sat}(\mathcal{C})$ for all assignments a of $\mathcal{C}$. We have to show that $a' \notin \mathrm{Sat}(\mathcal{C}')$ for all assignments a' of $\mathcal{C}'$. For $x \in \mathrm{Var}(\mathcal{C})$ and $x \in P_i$, $1 \leq i \leq t$, we define $a(x) := (a'(y_i))(x)$. The assignment a is well defined because $P_1, ..., P_t$ is a partition of $\mathrm{Var}(\mathcal{C})$. Also note that we consider $a'(y_i)$ here as an assignment of the form $P_i \to \mathrm{Dom}(\mathcal{C})$. We know that there exists some $C \in \mathcal{C}$ which is not satisfied by a. This implies that there exists some $C' \in \mathcal{C}'$ which is not satisfied by a' and which emerged from C. For seeing this, choose in the construction of C' the partial assignment $b \in \mathrm{Nonsat}_{P_i}(D)$ according to a, i.e., choose b such that $b(x) = a(x)$ for all $x \in P_i$.

It holds that $n' = t = \lceil \frac{n}{r} \rceil$. Note that the length of some assignment in $\mathrm{Nonsat}_{P_i}(D)$ is r and we introduce therefore d^r new values. The old values are not used any longer. The running time is polynomial in the input size with the exception of enumerating $\mathrm{Nonsat}_{P_i}(D)$ which takes time $O(d^r \cdot |\mathcal{C}|)$ and we may have to do this for every variable in a constraint of size at most k. This yields $O(d^{rk} \cdot |\mathcal{C}|)$. □

The following result is a direct implication of this lemma.

Corollary 1. *For constant d, d' and k with $d' \geq d$. It holds that $c_{d',k} \geq \lfloor \log_d(d') \rfloor \cdot c_{d,k}$.*

To prove Lemma 2 we will use algorithm SPARSIFY$_\varepsilon$ defined in Figure 1. The idea of our algorithm is similar to one of the many backtracking algorithms for the Maximum Independent Set Problem (MIS), namely, branching on vertices with large degree first. Johnson & Szegedy [9] applied the same technique to prove a sparsification lemma for MIS. Let $\varepsilon > 0$ and $K_{\varepsilon,d} := \frac{d}{\varepsilon \cdot \log(d)} \cdot \log(d^{(\varepsilon/d)}/(d^{(\varepsilon/d)} - 1))$. SPARSIFY$_\varepsilon$ uses the procedure SUBS($\mathcal{C}$) which

searches in $\mathcal{C}$ for constraints of the form $\{x \neq s\}$ and removes all $C \in \mathcal{C}$ with $|C| \geq 2$ and $(x \neq s) \in C$. It also uses the operation $\mathcal{C}^{[x \mapsto f]}$ by which all constraints $C \in \mathcal{C}$ with $(x \neq f') \in C$, $f' \neq f$, and all inequalities $x \neq f$ get removed from $\mathcal{C}$. Let $\text{freq}(\mathcal{C}, x, s)$ be the number of times $x \neq s$ occurs in $\mathcal{C}$.

Input: a $(d, 2)$-constraint system $\mathcal{C}$.
Output: a list $\mathcal{L}$ of $(d, 2)$-constraint systems.
1. if there exists $x \in \text{Var}(\mathcal{C})$ and $s \in \text{Dom}(\mathcal{C})$ s.t. $\text{freq}(\mathcal{C}, x, s) > \lceil K_{\varepsilon,d} \rceil$, then
2. call $\text{SPARSIFY}_\varepsilon(\text{SUBS}(\mathcal{C}^{[x \mapsto s]}))$;
3. call $\text{SPARSIFY}_\varepsilon(\text{SUBS}(\mathcal{C} \cup \{\{x \neq s\}\}))$;
4. else output $\mathcal{C}$;

Figure 1: Algorithm $\text{SPARSIFY}_\varepsilon$

Lemma 2. *Let $\mathcal{C}$ be a $(d, 2)$-constraint system and $\varepsilon > 0$. $\text{SPARSIFY}_\varepsilon$ enumerates with polynomial delay a list $\mathcal{L}$ of $(d, 2)$-constraint systems which has the following properties:*

1. *(Correctness) it holds that $\mathcal{C}$ is satisfiable iff there exists some satisfiable $\mathcal{C}' \in \mathcal{L}$,*
2. *(Bounded frequency) for all $\mathcal{D} \in \mathcal{L}$, $x \in \text{Var}(\mathcal{D})$, and $s \in \text{Dom}(\mathcal{D})$:*

$$\text{freq}(\mathcal{D}, x, s) \leq \lceil K_{\varepsilon,d} \rceil,$$

3. *(Size) $|\mathcal{L}| \leq d^{\varepsilon \cdot n}$.*

Proof. To see the correctness of algorithm $\text{SPARSIFY}_\varepsilon$ note that $\mathcal{C}$ is satisfiable iff $\text{SUBS}(\mathcal{C}^{[x \mapsto s]})$ or $\text{SUBS}(\mathcal{C} \cup \{\{x \neq s\}\})$ is satisfiable. The bounded frequency property holds because of the branching rule. It remains to prove the last property. $\text{SPARSIFY}_\varepsilon$ branches on pairs (x, s) with $\text{freq}(\mathcal{C}, x, s) > \lceil K_{\varepsilon,d} \rceil$. There are at most $d \cdot n$ such pairs. Let n' be the number of these pairs in $\mathcal{C}$ and $t(n')$ be the size of the search tree induced by $\text{SPARSIFY}_\varepsilon$. If we can show that $t(n') \leq d^{(\varepsilon/d) \cdot n'}$, then $|\mathcal{L}| \leq d^{\varepsilon \cdot n}$. For $n' \leq \lceil K_{\varepsilon,d} \rceil$ we can assume that $t(n') \leq d^{(\varepsilon/d) \cdot n'}$ holds. Now assume that the induction hypothesis $t(i) \leq d^{(\varepsilon/d) \cdot i}$ holds for $i \leq n' - 1$. $\text{SPARSIFY}_\varepsilon$ removes either at least 1 or at least $\lceil K_{\varepsilon,d} \rceil$ pairs according to the two cases of the branching rule. In the first case, $\text{SUBS}(\mathcal{C} \cup \{\{x \neq s\}\})$ yields a constraint system in which $\{x \neq s\}$ occurs once. No superset of $\{x \neq s\}$ occurs in $\text{SUBS}(\mathcal{C} \cup \{\{x \neq s\}\})$. In the second case, the constraint system $\mathcal{C}^{[x \mapsto s]}$ contains $\text{freq}(\mathcal{C}, x, s)$ new constraints of size 1 with no superset in $\text{SUBS}(\mathcal{C}^{[x \mapsto s]})$. Hence $t(n') \leq t(n' - 1) + t(n' - \lceil K_{\varepsilon,d} \rceil)$ which is by the induction hypothesis at most $d^{(\varepsilon/d) \cdot n' - (\varepsilon/d)} + d^{(\varepsilon/d) \cdot n' - (\varepsilon/d) \cdot \lceil K_{\varepsilon,d} \rceil} \leq d^{(\varepsilon/d) \cdot n'} \cdot (d^{-(\varepsilon/d)} + d^{-(\varepsilon/d) \cdot K_{\varepsilon,d}})$. By the definition of $K_{\varepsilon,d}$: $d^{-(\varepsilon/d)} + d^{-(\varepsilon/d) \cdot K_{\varepsilon,d}} = 1$. □

Proof (of Theorem 1). We apply Lemma 2 to a $(d, 2)$-constraint system $\mathcal{C}$ with fixed $\varepsilon = \gamma := c_{3,2}/(4 \log(3))$ and get a list $\mathcal{L}$ of constraint systems. Every

$\mathcal{C}' \in \mathcal{L}$ has maximum frequency $K_{\gamma,d} \cdot d$. Let $K_\gamma := \lceil \gamma^{-2} \rceil$. Then, $K_{\gamma,d} \leq K_\gamma \cdot d$. $K_{\gamma,d} \leq \gamma^{-2} d$ simplifies to $y - \frac{\ln(d)^2}{d} \frac{1}{y} \leq \ln(\mathrm{e}^y - 1)$ with $y := \gamma \frac{\ln(d)}{d}$. Note that $0 < \gamma \leq 1/4$ by the definition of γ and therefore we can assume $0 < y \leq \frac{\ln(d)}{4d}$. The function $f(y) := \ln(\mathrm{e}^y - 1) - y + \frac{\ln(d)^2}{d} \frac{1}{y}$ takes the minimum in $y = \frac{\ln(d)}{4d}$. The claim follows from $f(\frac{\ln(d)}{4d}) \geq 0$ for all $d \geq 3$. To reduce the maximum frequency to $3d^2$, we introduce for every variable x new variables $x^{(1)}, ..., x^{(K_\gamma)}$. We can express that $x^{(i)}$ has exactly the same value as $x^{(i+1)}$ with at most d^2 constraints, namely, with all constraints $\{x^{(i)} \neq s_1, x^{(i+1)} \neq s_2\}$, $s_1 \neq s_2$. We add all constraints for $1 \leq i \leq K_\gamma - 1$ to $\mathcal{C}'$ and replace every occurrence of x in such a way that for all $x \in \mathrm{Var}(\mathcal{C}')$, $s \in \mathrm{Dom}(\mathcal{C}')$: $\mathrm{freq}(\mathcal{C}', x, s) \leq 3d$. The number of variables is at most $K_\gamma \cdot n$. Using Corollary 1 we get the relation $c_{d,2} \geq \lfloor \log_3(d) \rfloor \cdot c_{3,2}$. Thus

$$c^{\mathrm{FQ}}_{d,2,3d^2} \cdot K_\gamma + \gamma \cdot \log(d) \geq c_{d,2} \geq \lfloor \log_3(d) \rfloor \cdot c_{3,2},$$

and $c^{\mathrm{FQ}}_{d,2,3d^2} \geq \lfloor \log_3(d) \rfloor \cdot c_{3,2}/(2K_\gamma)$. This completes the proof of Theorem 1. □

As a direct consequence we get a lower bound for

$$e_{d,k} := \inf\{c : \exists 2^{c \cdot m}\text{-randomized algorithm for } (d,k)\text{-CSP}\}$$

where m is the number of constraints.

Corollary 2. *If ETH holds, there exists $c > 0$ such that for all $d \geq 3$: $e_{d,2} \geq c \cdot \log(d)/d^2$.*

Note that $e_{d,2} \leq 2 \cdot \log(d)/d$ since we can remove every variable x which occurs less than d times (because then there is a remaining value we can assign to x to satisfy every constraint x occurs in). Hence, we may assume $|\mathcal{C}| \geq d/2 \cdot n$. Enumerating all possible assignments of the n variables yields the claimed upper bound.

Lemma 2 and the transformation afterwards give an upper bound of $3d^2$ on the variable frequency and actually the bound $\mathrm{freq}(\mathcal{C}, x, s) \leq 3d$. Let p_x be the number of possible values of x, that is, d minus the number of constraints of size 1 in which x occurs. Since we expect in the worst case that $p_x = \Omega(d)$ for $x \in V$ the following result suggests that this upper bound comes close to the best possible. For example, an improvement of $\mathrm{freq}(\mathcal{C}, x, s) \leq \sqrt{d}$ seems to be questionable.

Proposition 1 ([5]). *Let $\mathcal{C}$ be a $(d,2)$-constraint system and define $p_{\min} := \min_{x \in \mathrm{Var}(\mathcal{C})} p_x$. Then $\mathcal{C}$ is satisfiable, if for all $x \in \mathrm{Var}(\mathcal{C})$ and $s \in \mathrm{Dom}(\mathcal{C})$: $\mathrm{freq}(\mathcal{C}, x, s) \leq \frac{p_{\min}}{2}$.*

4 Unique CSP (Theorem 2)

The proof of Theorem 2 consists of four steps. Our goal is to prove the relation

$$c_{2,k} \cdot \lfloor \log(d) \rfloor \leq c_{d,k} \leq c^{\mathrm{UQ}}_{d,k} + O\left(\frac{\log(dk)}{k}\right) \text{ (Corollary 3).}$$

Taking the limit $k \to \infty$ proves Theorem 2. In this section we prove the upper bound on $c_{d,k}$. The lower bound follows from Corollary 1, Section 3. The first step in our proof of the upper bound is a generalization of the isolation lemma from [11]. We generalize this lemma from the boolean to the non-boolean case (Lemma 3). We can however not apply this lemma directly to prove our upper bound. In a second step, we therefore show how to use it to get an isolation lemma (Lemma 4) which fits our needs. The crucial difference between the isolation lemma from [11] and Lemma 4 is that we can encode the random linear equations from Lemma 4 by a (d,k)-constraint system. This is done in the third step (Lemma 5). Finally, we can put it all together (Corollary 3) and prove Theorem 2.

We conclude this section with a remark on our main technical contribution, Lemma 4.

We start with a generalization of Lemma 1 from [11]. The lemma there states that if $S \subseteq \{0,1\}^n$ is non-empty, then the probability that S has a unique minimum w.r.t. a random weight function is at least $1/2$. Our result works for non-empty $S \subseteq \{0,...,d-1\}^n$.

Lemma 3. *Let $n, c \in \mathbb{N}$ and $S \subseteq \{0,...,d-1\}^n$, $S \neq \{\}$. Choose w_i, $1 \leq i \leq n$, independently and uniformly from $\{1,...,c\}$. Define a random weight function $w : \{0,...,d-1\}^n \to \mathbb{N}$ as $w : a \mapsto \sum_{i=1}^{n} w_i \cdot a_i$. It holds that*

$$\Pr_w(S \text{ has a unique minimum w.r.t. } w) \geq 1 - n \cdot \frac{\binom{d}{2}}{c}.$$

Proof. Let $1 \leq i \leq n$ and $0 \leq l \leq d-1$. Define $S_{i,l} := \{a \in S : a_i = l\}$ and

$$M_{i,l} := \begin{cases} \min_{a \in S_{i,l}} w(a) - l \cdot w_i \text{ if } S_{i,j} \neq \{\} \\ 0 \end{cases}.$$

Denote by E_i the event that $\exists 0 \leq j < k \leq d-1 : M_{i,j} + j \cdot w_i = M_{i,k} + k \cdot w_i$. For any i it holds that

$$\Pr_w(E_i) = \Pr_w(\exists 0 \leq j < k \leq d-1 : (M_{i,j} - M_{i,k})/(k-j) = w_i) \leq$$
$$\binom{d}{2} \Pr_w((M_{i,j} - M_{i,k})/(k-j) = w_i) \leq \binom{d}{2}/c.$$

Here, we used the union bound and the fact that

$$\Pr_w((M_{i,j} - M_{i,k})/(k-j) = w_i) = \begin{cases} \frac{1}{c} \text{ if } (M_{i,j} - M_{i,k})/(k-j) \in \{1,...,c\} \\ 0 \end{cases}$$

(w_i is chosen independently of $w_1,...,w_{i-1},w_{i+1},...,w_n$). Applying the union bound we get (*)

$$\Pr_w(\exists 0 \leq i \leq n : E_i) \leq n \cdot \frac{\binom{d}{2}}{c}.$$

Finally, assume that there exist $a \neq b \in S$ which take the minimum value w.r.t. w. Since $a \neq b$ there exists $1 \leq i \leq n$ such that $a_i \neq b_i$ and $M_{i,a_i} + a_i \cdot w_i =$

$M_{i,b_i} + b_i \cdot w_i$. This can happen with probability at most $n \cdot \binom{d}{2}/c$ because of (*). Hence, the probability that S has a unique minimum w.r.t. w is at least $1 - n \cdot \binom{d}{2}/c$. □

The random weight function w depends on n variables. This makes it at the first sight useless for our needs since we can not encode it as a constraint system in subexponential time. We can however apply it iteratively as we will see in the proof of the following lemma.

Lemma 4. *Let $d \geq 2$, $k \geq 1$, and $S \subseteq \{0, ..., d-1\}^n$ be non-empty. There exists a polynomial time computable set $\mathcal{L}$ of $\lceil \frac{n}{k} \rceil$ random linear equations, each depending on at most k variables, such that*

$$\Pr_{\mathcal{L}}(|S \cap \mathrm{Sol}_d(\mathcal{L})| = 1) \geq 2^{-O(n\frac{\log(dk)}{k})},$$

where $\mathrm{Sol}_d(\mathcal{L})$ is the set of solutions of $\mathcal{L}$ in $\{0, ..., d-1\}^n$.

Proof. We employ Lemma 3. Let $c := 2 \cdot \binom{d}{2} \cdot k$. Independently and uniformly choose w_i from $\{1, ..., c\}$ for all i. We define $\mathcal{L}$ to be the set of linear equations

$$\sum_{i=1+j\cdot k}^{(1+j)\cdot k} w_i \cdot x_i = r_j$$

for $0 \leq j \leq t-1$ and $\sum_{i=1+t\cdot k}^{n} w_i \cdot x_i = r_t$, where $r_0, ..., r_t$ are chosen uniformly at random from $\{0, ..., c \cdot (d-1) \cdot k\}$.

For simplicity we assume that n is a multiple of k, i.e., there exists i such that $n = ik$. We prove by induction over i that the i equations in $\mathcal{L}$ have a unique solution in S with probability at least $(2c(d-1)k+2)^{-i}$. If $i = 1$, then the probability that S has a unique minimum w.r.t. $w_1, ..., w_k$ is at least $1/2$ by Lemma 3 and the probability of guessing the right value r_1 is at least $1/(c(d-1)k+1)$; together at least $1/(2c(d-1)k+2)$. Now, assume the induction hypothesis holds for $i-1$. Let $S' := \{a_{n-k+1}...a_n : a \in S\}$. The probability that the corresponding equation in the variables $x_{n-k+1}, ..., x_n$ has a unique solution in S' is at least $1/(2c(d-1)k+2)$. Let $a_{n-k+1}...a_n$ be this solution and $S'' := \{b \in S : b_{n-k+1} = a_{n-k+1}, ..., b_n = a_n\}$. By the induction hypothesis the probability that the first $i-1$ equations have a unique solution in S'' is at least $(2c(d-1)k+2)^{-i-1}$. Since $w_1, ..., w_n$ and $r_0, ..., r_{i-1}$ are chosen uniformly and independently the overall success probability is at least $(2c(d-1)k+2)^{-i}$. Hence, $\Pr_{\mathcal{L}}(|S \cap \mathrm{Sol}_d(\mathcal{L})| = 1)$ is greater or equal than

$$(2c(d-1)k+2)^{-n/k-1} \geq (4d^3k^2)^{-n/k-1} \geq 2^{-O(n\frac{\log(dk)}{k})}.$$

□

The next lemma states the simple but important fact that we can encode the random linear equations from the previous lemma as a (d,k)-constraint system.

Lemma 5. *Let $\mathcal{C}$ be a constraint system with domain size d over n variables. There exists a (d,k)-constraint system $\mathcal{C}'$ over n variables computable in time $d^k \cdot \mathrm{poly}(|\mathcal{C}|)$ such that if $\mathcal{C}$ is satisfiable, then $\mathcal{C} \cup \mathcal{C}'$ has exactly one satisfying assignment with probability at least $2^{-O(n \cdot \frac{\log(dk)}{k})}$. Moreover, $|\mathcal{C}'| \leq d^k \cdot (n/k+1)$.*

Proof. Let $\mathcal{L}$ be as in Lemma 4. For every equation in $\mathcal{L}$ we can enumerate all assignments of the k variables in d^k steps. Thus, we can encode a single equation as a (d,k)-constraint system in polynomial time. We define $\mathcal{C}'$ to be the set of these at most $d^k \cdot (n/k+1)$ constraints. Now, let $S := \mathrm{Sat}(\mathcal{C})$. If $\mathcal{C}$ is unsatisfiable, then $\mathcal{C} \cup \mathcal{C}'$ is unsatisfiable. Otherwise, $S \neq \{\}$. The probability that $\mathcal{C} \cup \mathcal{C}'$ has exactly one satisfying assignment is as in Lemma 4. □

We are now at a point where we can prove Theorem 2. It follows from the following corollary by taking the the limit $k \to \infty$.

Corollary 3. *For all $d \geq 2$ and $k \geq 2$, it holds that $c_{2,k} \cdot \lfloor \log(d) \rfloor \leq c_{d,k} \leq c^{UQ}_{d,k} + O(\frac{\log(dk)}{k})$.*

Proof. The relation $c_{2,k} \cdot \lfloor \log(d) \rfloor \leq c_{d,k}$ follows from Corollary 1. To prove $c_{d,k} \leq c^{\mathrm{UQ}}_{d,k} + O(\log(dk)/k)$ we apply Lemma 5 $2^{O(n \cdot (\log(dk)/k))}$ times and test every time if $\mathcal{C} \cup \mathcal{C}'$ is satisfiable using an algorithm A of time complexity $2^{(c_{d,k}+\delta) \cdot n}$, $\delta \geq 0$, for (d,k)-UNIQUE-CSP. If A once accepts, $\mathcal{C}$ gets accepted, otherwise rejected. □

In the proof of Lemma 5 we used the fact that we can encode the solutions of a random linear equation as a constraint system without changing the number of variables. The opposite is however not true. Therefore we may say that Lemma 4 is stronger than Lemma 5. In particular, if we want to obtain similar relations as in Corollary 3 for other problems, Lemma 4 is appropriate if the problem at hand allows a compact encoding of the solutions of a random linear equation. This is for example the case for Binary Integer Programming. We also remark here that for the proof of Theorem 1 it is not necessary that $\mathcal{C}'$ has constant constraint size k. For example, $k = \sqrt{n}$ suffices. To give an example of a situation where it is necessary that $\mathcal{C}'$ has constant constraint size k we prove Corollary 4.

Corollary 4. *ETH holds iff $c^{UQ}_{3,2} > 0$.*

Proof. Let $\mathcal{C}$ be a $(3,2)$-constraint system over n variables and $\varepsilon > 0$. Make $k = k(\varepsilon)$ large enough such that $O(\log(dk)/k) < \varepsilon$. Applying Lemma 5 we get a constraint system $\mathcal{C} \cup \mathcal{C}'$. The constraints in $\mathcal{C}'$ have size at most k and $|\mathcal{C}'| \leq n \cdot (3^k+k)/k$. By introducing $K \leq n \cdot (3^k+k)$ new variables we can transform $\mathcal{C}'$ into a $(3,2)$-constraint system $\mathcal{C}''$ with the same number of satisfying assignments. Set $\mathcal{C}''$ to $\mathcal{C}'$. Replace every $\{x_1 \neq s_1, ..., x_l \neq s_l\} \in \mathcal{C}''$ with $l > 2$ by $\{x_1 \neq s_1, y \neq 1\}$, $\{x_2 \neq s_2, y \neq 2\}$, $\{x_3 \neq s_3, ..., x_l \neq s_l, y \neq 3\}$. Here, y is a new variable not used before. We add constraints to $\mathcal{C}''$ which say that $x_1 \neq s_1$ implies $y \neq 2$ and that $x_1 \neq s_1$ implies $y \neq 3$. Hence, if the inequality $x_1 \neq s_1$ is satisfied y is forced to

be 1. The constraints are $\{x_1 \neq s_1^1, y \neq 2\}$, $\{x_1 \neq s_1^2, y \neq 2\}$, $\{x_1 \neq s_1^1, y \neq 3\}$ and $\{x_1 \neq s_1^2, y \neq 3\}$ where $\{s_1, s_1^1, s_1^2\} = \mathrm{Dom}(\mathcal{C})$. Next we add constraints to $\mathcal{C}''$ which say that $x_2 \neq s_2$ implies $y \neq 3$. In the case that the inequality $x_1 \neq s_1$ is not satisfied but $x_2 \neq s_2$ is y is forced to be 2. The constraints are $\{x_2 \neq s_2^1, y \neq 3\}$, and $\{x_2 \neq s_2^2, y \neq 3\}$ where $\{s_2, s_2^1, s_2^2\} = \mathrm{Dom}(\mathcal{C})$. In the last case that $x_1 \neq s_1$ and $x_2 \neq s_2$ are not satisfied y is forced to be 3. We repeat this step until every constraint has size at most 2. In every step the size of one constraint is reduced by one and exactly one variable is used. Hence, we need $K \leq n \cdot (3^k + k)$ new variables. We conclude that $c_{3,2} \leq (3^{k(\varepsilon)} + k(\varepsilon)) \cdot c_{3,2}^{\mathrm{UQ}} + \varepsilon$ for every $\varepsilon > 0$. If $c_{3,2}^{\mathrm{UQ}} = 0$, then $c_{3,2} \leq \varepsilon$ for every $\varepsilon > 0$. A contradiction to ETH. □

Acknowledgments

Thanks to Robert Berke for pointing out [5].

References

1. Andreas Björklund and Thore Husfeldt. Inclusion–exclusion algorithms for counting set partitions. In *Proc. of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 575–582, 2006.
2. Chris Calabro, Russell Impagliazzo, Valentine Kabanets, and Ramamohan Paturi. The complexity of unique k-SAT: An isolation lemma for k-CNFs. In *Proc. of the 18th Annual IEEE Conference on Computational Complexity*, pages 135–141, 2003.
3. David Eppstein. Improved algorithms for 3-coloring, 3-edge-coloring, and constraint satisfaction. In *Proc. of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 329–337, 2001.
4. Tomás Feder and Rajeev Motwani. Worst-case time bounds for coloring and satisfiability problems. *J. Algorithms*, 45(2):192–201, 2002.
5. Penny E. Haxell. A condition for matchability in hypergraphs. *Graphs and Combinatorics*, 11:245–248, 1995.
6. Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *J. Computer and System Sciences*, 62(2):367–375, 2001.
7. Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Computer and System Sciences*, 63(4):512–530, 2001.
8. Kazuo Iwama and Suguru Tamaki. Improved upper bounds for 3-SAT. In *Proc. of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 328–329, 2004.
9. David S. Johnson and Mario Szegedy. What are the least tractable instances of max independent set? In *Proc. of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 927–928, 1999.
10. Mikko Koivisto. An $\mathrm{O}(2^n)$ algorithm for graph coloring and other partitioning problems via inclusion–exclusion. In *Proc. of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 583–590, 2006.

11. Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987.
12. Uwe Schöning. A probabilistic algorithm for k-SAT and constraint satisfaction problems. In *Proc. of the 40th Annual Symposium on Foundations of Computer Science*, pages 410–414, 1999.
13. Alexander D. Scott and Gregory B. Sorkin. An LP-designed algorithm for constraint satisfaction. In *Proc. of the 14th Annual European Symposium on Algorithms*, pages 588–599, 2006.
14. L. G. Valiant and V. V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47(1):85–93, 1986.
15. Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Computer Science*, 348(2-3):357–365, 2005.