Faster Algorithm for Unique $(k, 2)$-CSP

Or Zamir
Institute for Advanced Study, Princeton, NJ, USA

Abstract

In a $(k, 2)$-Constraint Satisfaction Problem we are given a set of arbitrary constraints on pairs of k-ary variables, and are asked to find an assignment of values to these variables such that all constraints are satisfied. The $(k, 2)$-CSP problem generalizes problems like k-coloring and k-list-coloring. In the Unique $(k, 2)$-CSP problem, we add the assumption that the input set of constraints has at most one satisfying assignment.

Beigel and Eppstein gave an algorithm for $(k, 2)$-CSP running in time $O\left((0.4518k)^n\right)$ for $k > 3$ and $O\left(1.356^n\right)$ for $k = 3$, where n is the number of variables. Feder and Motwani improved upon the Beigel-Eppstein algorithm for $k \geq 11$. Hertli, Hurbain, Millius, Moser, Scheder and Szedlák improved these bounds for Unique $(k, 2)$-CSP for every $k \geq 5$.

We improve the result of Hertli et al. and obtain better bounds for Unique $(k, 2)$-CSP for $k \geq 5$. In particular, we improve the running time of Unique $(5, 2)$-CSP from $O\left(2.254^n\right)$ to $O\left(2.232^n\right)$ and Unique $(6, 2)$-CSP from $O\left(2.652^n\right)$ to $O\left(2.641^n\right)$.

Recently, Li and Scheder also published an improvement over the algorithm of Hertli et al. in the same regime as ours. Their improvement does not include quantitative bounds, we compare the works in the paper.

2012 ACM Subject Classification Mathematics of computing → Combinatorial algorithms

Keywords and phrases Algorithms, Constraint Satisfaction Problem

Digital Object Identifier 10.4230/LIPIcs.ESA.2022.92

Funding Supported by NSF grant # CCF-1900460.

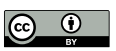
1 Introduction

The general Constraint Satisfaction Problem, in which we are asked to find an assignment to a set of variables that satisfies a list of arbitrary constraints, is NP-Complete. Furthermore, it is widely believed that a substantial improvement over the naive exhaustive search is unlikely for the general CSP problem and even for special cases of it like The *Boolean* Satisfiability Problem (SAT). Nevertheless, when the structure of the input is restricted in a certain manner, there are known improvements in the form of *moderately exponential* algorithms. These are algorithms that still have an exponential running time, yet achieve an exponential improvement over the exhaustive search bounds.

The study of moderately exponential algorithms for NP-Complete problems is extensive. In fact, exponential yet better-than-naive algorithms for NP-Complete problems were known for some problems, for example The Travelling Salesman Problem, long before the definition of NP. A survey of Woeginger [19] covers and refers to dozens of papers exploring such algorithms for many problems including satisfiability, graph coloring, knapsack, TSP, maximum independent sets and more. Subsequent review article of Fomin and Kaski [5] and book of Fomin and Kratsch [6] further cover the topic of exact exponential-time algorithms.

Two of the most notable problems for which the study of moderately exponential algorithms was fruitful are k-satisfiability (usually abbreviated as k-SAT) and graph coloring.

For satisfiability, the running time of the trivial algorithm enumerating over all possible assignments is $O^*(2^n)$. No algorithms solving SAT in time $O^*\left((2-\varepsilon)^n\right)$ for any $\varepsilon > 0$ are known, and a popular conjecture called The *Strong* Exponential Time Hypothesis [3] states that no such algorithm exists. On the other hand, for every fixed k there exists a constant



$\varepsilon_k > 0$ such that k-SAT (i.e., SAT on formulas in CNF form with at most k literals in every clause) can be solved in $O^*\left((2-\varepsilon_k)^n\right)$ time. A result of this type was first published by Monien and Speckenmeyer in 1985 [13]. A long list of improvements for the values of ε_k were published since, including the celebrated 1998 PPSZ algorithm of Paturi, Pudlák, Saks and Zane [15] and the recent improvement over it by Hansen, Kaplan, Zamir and Zwick [7]. The PPSZ bound was originally obtained only for the case of input formulas with a unique satisfying assignment, its analysis was extended to the general case more than a decade later by Hertli [8].

For graph coloring, i.e, the problem of deciding whether a graph is k-colorable, the naive exhaustive search algorithm takes $O(k^n)$ time. Nevertheless, it is known that computing the chromatic (or coloring) number of a graph (i.e., the smallest k for which the graph is k-colorable) can be done in exponential time that does not depend on k. The first such result was an $O^*(3^n)$ algorithm of Lawler [10]. A long line of works followed until finally an algorithm computing the chromatic number in $O^*(2^n)$ time was devised by Björklund, Husfeldt and Koivisto in 2009 [2]. This is conjectured to be optimal. For $k \leq 6$ there are algorithms solving k-colorability exponentially faster than $O(2^n)$ [1][20]. It is currently not known even if 7-coloring can be solved exponentially faster than the general $O(2^n)$ bound of computing the coloring number. One of the biggest open problems in this field is whether k-coloring can be solved in $O^*\left((2-\varepsilon_k)^n\right)$ time for every fixed k.

Both examples are special cases of the more general Constraint Satisfaction Problem. In an (a, b)-formula, we have n variables such that each of them can take a value in $[a] := \{1, \ldots, a\}$ and a list of constraints such that each constraint may depend on at most b variables. Every such constraint can be equivalently replaced by a disjunction of at most a^b constraints of the form $(x_1 \neq c_1 \vee x_2 \neq c_2 \vee \ldots \vee x_b \neq c_b)$ where $x_1, \ldots, x_b$ are (not necessarily distinct) variables and $c_1, \ldots, c_b \in [a]$ are possible values. Thus, we can think of every (a, b)-formula as a list of constraints of that form. In the (a, b)-CSP problem we are given a (a, b)-formula and need to decide whether or not there is an assignment to the variables that satisfies all constraints. In the Unique (a, b)-CSP problem we add the assumption that if there is such an assignment it is unique. Note that k-SAT is the same as $(2, k)$-CSP, and that k-coloring is a special case of $(k, 2)$-CSP. We later elaborate on the close relation between the $(k, 2)$-CSP and k-coloring problems.

In this paper we focus on obtaining better algorithms for Unique $(k, 2)$-CSP.

1.1 Possible running time

Denote by $c_{a,b}$ the infimum of constants c such that (a, b)-CSP on formulas with n variables can be solved in $O\left((c + o(1))^n\right)$ time. Naively, $c_{a,b} \leq a$ as we can simply try all a^n possible assignments to the variables. A simple improvement comes from the use of *down-sampling*. Given a (a, b)-formula we may randomly restrict each variable to $a' < a$ uniformly chosen values. Each satisfying assignment is not ruled out by the restriction with probability $\left(\frac{a'}{a}\right)^n$. After this down-sampling step, we are left with a (a', b)-formula. Thus, for every $a' < a$ we have $c_{a,b} \leq \frac{a}{a'} \cdot c_{a',b}$. In particular, $c_{a,b} \leq \frac{a}{2} c_{2,b}$. As we know that k-SAT can be solved exponentially faster than $O(2^n)$ for every fixed k, we have that $c_{2,b} < 2$ and in particular the strict inequality $c_{a,b} < a$ holds for every a, b.

On the other hand, (a, b)-CSP is clearly NP-Complete for every $a > 1$ except of the special case of $(a, b) = (2, 2)$ (which is the polynomial 2-SAT). The Exponential Time Hypothesis [3] states that there exists some constant $c > 0$ such that 3-SAT takes $\Omega(2^{cn})$ time to solve. Traxler [18] showed that assuming the Exponential Time Hypothesis, there exists some $c' > 0$ such that $c_{k,2} > k^{c'}$. Namely, even for $b = 2$ the $(k, 2)$-CSP problem becomes strictly more complex as k increases.

Table 1 Comparisons of the exponent base in Unique $(k, 2)$-CSP algorithms.

k	Downsampling+2SAT	PPZ FM [4]	BE [1]	PPSZ [9]	**Our algorithm**
3	1.5	1.818	*1.365*	1.434	-
4	2	2.214	*1.808*	1.849	-
5	2.5	2.606	2.259	*2.254*	**2.232**
6	3	2.994	2.711	*2.652*	**2.641**
7	3.5	3.381	3.163	*3.045*	**3.042**

1.2 $(k, 2)$-CSP and k-Coloring

Consider the following hierarchy of three problems:

- ***k*-Coloring:** given a graph with n vertices, determine whether it is k-colorable.
- ***k*-List-Coloring:** given a graph with n vertices and a list of at most k allowed colors (from a possibly larger universe of colors) for each vertex, determine if there is a proper coloring of the graph such that each vertex is colored with one of the allowed colors in its list.
- **$(k, 2)$-CSP:** given n variables that can admit values from $[k]$ and a list of arbitrary constraints involving one or two variables each, determine if there is an assignment of values to the variables that satisfies all constraints.

Each problem is a special case of the next one. Every instance of k-coloring is also an instance of k-list-coloring where all lists are simply $[k]$. Every instance of k-list-coloring is also an instance of $(k, 2)$-CSP. Nevertheless, while k-coloring and k-list-coloring can both be solved in $O^*(2^n)$ time regardless of k [2], Traxler's reduction [18] shows that there is some constant k_0 such that for every $k > k_0$ we have $c_{k,2} \geq 2$. Let k_0 be the minimal such constant. It is currently known that $c_{4,2} < 2$ and thus $k_0 \geq 4$.

In [20] it was recently shown that if k-list-coloring can be solved in $O^*\left((2-\varepsilon)^n\right)$ time for some $\varepsilon > 0$ then $(k+2)$-coloring can also be solved in $O^*\left((2-\varepsilon')^n\right)$ for some $\varepsilon' > 0$. In particular, (k_0+2)-coloring can be solved exponentially faster than $O(2^n)$. As $k_0 \geq 4$, this resulted in the first $O^*\left((2-\varepsilon)^n\right)$ time algorithms for 5-coloring and 6-coloring. This gives a strong motivation for improving upper bounds for $(k, 2)$-CSP, with the goal of improving the bound on k_0. In particular, showing that $(5, 2)$-CSP can be solved in $O^*\left((2-\varepsilon)^n\right)$ time would result in the first $O^*\left((2-\varepsilon)^n\right)$ time algorithm for 7-coloring.

1.3 Previous results

By the down-sampling argument we can reduce $(k, 2)$-CSP to the polynomial $(2, 2)$-CSP (i.e., 2-SAT) and get an expected running time of $O\left(\left(\frac{k}{2}\right)^n\right)$. Beigel and Eppstein [1] gave an algorithm for $(k, 2)$-CSP running in time $O\left((0.4518k)^n\right)$ for $k > 3$ and $O\left(1.356^n\right)$ for $k = 3$. Feder and Motwani [4] give a $(k, 2)$-CSP algorithm based on the k-SAT PPZ algorithm, which is the predecessor of the PPSZ one. They improve on the bound of Beigel and Eppstein only for $k \geq 11$. Hertli, Hurbain, Millius, Moser, Scheder and Szedlák [9] improved the bounds for Unique $(k, 2)$-CSP for every $k \geq 5$. Several other works [17] [11] [16] focus on the case where $b > 2$.

1.4 Our contribution

We present an algorithm that improves on the result of Hertli et al. and obtain better bounds for Unique $(k, 2)$-CSP for $k \geq 5$. In particular, we improve the running time of Unique $(5, 2)$-CSP from $O\left(2.254^n\right)$ to $O\left(2.232^n\right)$ and Unique $(6, 2)$-CSP from $O\left(2.652^n\right)$ to $O\left(2.641^n\right)$. Our result is compared to the previous ones in Table 1.

We obtain our result by combining the strengths of both PPSZ and the Beigel-Eppstein algorithms. Intuitively, we make the following insight regarding PPSZ-type algorithms. Throughout the run of the PPSZ algorithm, it slowly manipulates the CSP formula. For every $k' < k$ there is some time-point such that if we stop the algorithm at that point then the formula roughly looks like a $(k', 2)$-CSP formula. Furthermore, the rest of the algorithm run would have looked similar to running PPSZ on a $(k', 2)$-formula. Thus, for $k \geq 5$ we may stop the run of the PPSZ algorithm when the formula looks similar to a $(4, 2)$-CSP or $(3, 2)$-CSP formula, and then switch to using the Beigel-Eppstein algorithm which is faster than PPSZ for $k \leq 4$.

In Section 2 we give an extensive overview of the previous results we need to use. Then in Section 3 we introduce our algorithm and obtain the improved bounds. In Section 4 we sketch possible improvements to the analysis of Section 3 and show that even with a completely ideal analysis of our algorithm we would improve the bound for $k = 5$ from $O(2.232^n)$ only to $O(2.223^n)$. Thus, to prove that $k_0 \geq 5$, if true, additional algorithmic tools are necessary. In Section 5 we conclude our work, discuss more possible uses for the PPSZ-related observations, and present open problems.

1.5 Comparison with a recent work of Li and Scheder

Recently, Li and Scheder [12] also published an improvement for the PPSZ-type algorithm of Hertli et al. for Unique CSP. Their algorithm does not use the Beigel-Eppstein algorithm and just modifies the PPSZ-type algorithm itself in a different manner to ours. They prove that this modification gives an exponential improvement over the bounds of Hertli et al., but do not give a quantitative bound of this improvement. We could not find a way to compute such a bound from their paper, but suspect this improvement is very small.

Li and Scheder also observe that during the run of the PPSZ-type algorithm when the number of color choices of a variable is very low (in their algorithm, when it reaches 2), then there are better ways to settle the value of the variable than continuing the run of the PPSZ algorithm. In their case, they do so by randomly picking one of the two colors for the variable.

Our work can be seen as a refinement of this idea in two different ways. First, we cut off the PPSZ run with small color sets yet larger than two. Second, we resolve the remaining instance with a variant of the Beigel-Eppstein algorithm, which is much better than a random choice.

2 Relevant overview of previous work

In this section we give an overview of all previous results that are used in our algorithm. We repeat and refine some of the theorems used in these papers for their later use in Section 3.

2.1 The algorithm of Beigel and Eppstein

The algorithm of Beigel and Eppstein [1] solves a CSP by performing a series of local reductions that either reduce the number of variables in the CSP or the number of allowed values in some of the variables.

An example for such a local reduction that is of particular interest to us follows.

▶ **Lemma 1** (Lemma 2 of [1]). *Let (V, F) be a CSP in which each variable $x \in V$ has $k(x)$ allowed values. Let x be a variable with $k(x) = 2$, then there exists a set F' of additional constraints, each of size two, such that (V, F) is satisfiable if and only if $(V \setminus \{x\}, F \cup F')$ is satisfiable, with the same number of allowed values for each variable $y \neq x$.*

The result claimed in [1] is that their algorithm solves $(3,2)$-CSPs in time $O(1.3645^n)$ and $(k,2)$-CSPs for $k>3$ in time $O((0.4518k)^n)$. Note that $1.3645 > 1.3554 = 0.4518 \cdot 3$. In fact, the result proved in their paper is slightly stronger than that. The following Theorem follows from [1].

▶ **Theorem 2** (Section 5 of [1]). *Let (V,F) be a CSP with $|V| = n_3 + n_4$ variables such that n_3 variables have three allowed values and n_4 variables have four allowed values, then we can solve it in time $O(1.3645^{n_3} \cdot 1.8072^{n_4})$.*

The claimed results follow from Theorem 2 immediately. For $(3,2)$-CSPs we simply have $n_3 = n,\ n_4 = 0$ and for $(k,2)$-CSPs with $k>3$ we down-sample each variable to 4 out of its k possible values and then use the theorem with $n_3 = 0,\ n_4 = n$, with a total expected run-time of $O\left(\left(\frac{k}{4}\right)^n \cdot 1.8072^n\right) = O((0.4518k)^n)$. Nevertheless, for our use we need to fully use the power of Theorem 2 and we even slightly refine it with the following statement, from now on referred to as *the extended BE algorithm*.

▶ **Theorem 3.** *Denote by*

$$BE(i) := \begin{cases} 1 & \text{if } i \le 2 \\ 1.3645 & \text{if } i = 3 \\ 0.4518 \cdot i & \text{if } i \ge 4 \end{cases}.$$

Let (V,F) be a CSP in which each variable $x \in V$ has $k(x)$ allowed values. Let n_i be the number of variables x in V such that $k(x) = i$. Then, we can solve (V,F) in $O(\prod_i BE(i)^{n_i})$ expected time.

Proof. The n_1 variables with a single possible value can be ignored. The n_2 variables with two possible values can be eliminated using Lemma 1. For every $i > 4$ we use down-sampling to reduce the number of allowed values to four. We finally apply Theorem 2. ◀

2.2 The PPZ and PPSZ-type algorithms

In this section we present an overview of [4] and [9]. We state theorems of both papers and their variants that are useful for our analysis, and adapt some of their notation. Throughout the section we discuss only instances with a unique satisfying assignment.

▶ **Definition 4** (D-implication). *Let F be a $(k,2)$-CSP formula over a set V of variables, $x \in V$ be a variable, $c \in [k]$ be a possible value, α_0 a partial assignment, and $D \in \mathbb{N}$. We say that α_0 D-implies $x \neq c$ and write $\alpha_0 \models_D (x \neq c)$ if there is a subset of constraints $G \subseteq F$ of size $|G| \le D$ such that $G \wedge \alpha_0$ implies $(x \neq c)$.*

By enumeration, we can check whether $\alpha_0 \models_D (x \neq c)$ in $O\left(|F|^D \cdot poly(n)\right)$ time, which is polynomial in n,k if D is a constant and sub-exponential in n even if D is a slow-enough growing function of n. For the rest of the section we fix D.

▶ **Definition 5** (Eligible values). *Let F be a $(k,2)$-CSP formula over a set V of variables, α_0 a partial assignment, and $x \in V \setminus V(\alpha_0)$ a variable which value is not assigned in α_0. We denote by*

$$\mathcal{A}(x,\alpha_0) := \{c \in [k] \mid \alpha_0 \not\models_D (x \neq c)\}$$

the set of all possible values for x that are not ruled out by D-implication from α_0.

We can now describe the PPSZ algorithm (adapted from SAT [14] to CSP in [9]). Given a $(k, 2)$-CSP F, we begin with $\alpha_0 = \emptyset$ the empty assignment and incrementally add variables to it, hoping to finish with a satisfying assignment. In particular, we choose a permutation π of the variables V uniformly at random, and then choose an assignment for the variables of V one-by-one according to the order of π. When we reach a variable x, we compute $\mathcal{A}(x, \alpha_0)$ in sub-exponential time, pick a uniformly random $c \sim U(\mathcal{A}(x, \alpha_0))$, and extend α_0 by setting $\alpha_0(x) = c$.

Algorithm 1 The PPSZ algorithm.

Pick a uniform random permutation π of the set V of variables;
Set $\alpha_0 = \emptyset$;
for $x \in V$ *in the order dictated by* π **do**
  Draw $c \sim U(\mathcal{A}(x, \alpha_0))$ uniformly;
  Set $\alpha_0(x) := c$;
Return α_0;

Algorithm 1 runs in sub-exponential time and returns some assignment α_0 to all variables of V. It is clear that the probability of α_0 to satisfy F, assuming that F is satisfiable, is at least k^{-n}. We next prove that for formulas F with exactly one satisfying assignment α, the probability that Algorithm 1 produces the satisfying assignment $\alpha_0 = \alpha$ is in fact exponentially larger. For the rest of the section we assume that F has a unique satisfying assignment and denote it by α.

▶ **Definition 6** (Ultimately eligible values). *Let F be a $(k, 2)$-CSP formula uniquely satisfied by α, π be a permutation of its variables V and $x \in V$ some variable.*
We let $V_{\pi,x} := \{y \in V \mid \pi(y) < \pi(x)\}$ be the set of all variables appearing before x in π, $\alpha_{\pi,x} := \alpha\big|_{V_{\pi,x}}$ be the partial assignment resulting by restricting α to $V_{\pi,x}$ and then we denote by $\mathcal{A}(x, \pi) := \mathcal{A}(x, \alpha_{\pi,x})$ the set of all possible values for x that are not ruled out by D-implication when we reach x in a PPSZ iteration with permutation π, given that all previous variables were set correctly.

We observe that Algorithm 1 returns α if and only if it draws the correct value for *every* variable. In particular, for a specific permutation π the probability of success is exactly $\prod_{x \in V} \frac{1}{|\mathcal{A}(x,\pi)|}$. For a random permutation then, the probability of success is

$$\mathbb{E}_\pi \left[\prod_{x \in V} \frac{1}{|\mathcal{A}(x, \pi)|} \right] \geq k^{-\sum_{x \in V} \mathbb{E}_\pi [\log_k |\mathcal{A}(x,\pi)|]}$$

where we use Jensen's inequality. In particular, it is enough to give an upper bound on $\mathbb{E}_\pi [\log_k |\mathcal{A}(x, \pi)|]$ that holds for every variable x.

2.3 The PPZ-type algorithm of Feder and Motwani

In the PPZ-type variant of Feder and Motwani [4], a simpler variant where $D = 1$ is presented and analysed. Namely, a possible value c for a variable x is ruled out if and only if a variable y appeared before x in the permutation and was assigned a value c' such that the constraint $(x \neq c \vee y \neq c')$ appears in the list of constraints. When we reach the variable x, we uniformly guess a value for it out of all values that are not ruled out in that manner.

▶ **Lemma 7.** *Let (V, F) be the sets of variables and constraints in a Unique $(k, 2)$-CSP. Denote by φ the unique satisfying assignment of (V, F). For every variable $x \in V$ and every value $\varphi(x) \neq c' \in [k]$ other than the value x is assigned in φ, there exists a variable $y = y_{x,c'} \in V \setminus \{x\}$ such that $(x \neq c' \vee y \neq \varphi(y)) \in F$.*

Proof. Assume by contradiction that there exists a variable x and a value $c' \neq \varphi(x)$ for which $(x \neq c' \vee y \neq \varphi(y)) \notin F$ for every variable y. Consider the assignment φ' such that $\varphi'(x) = c'$ and $\varphi'(y) = \varphi(y)$ for every $y \neq x$. It is a satisfying assignment as well, and $\varphi' \neq \varphi$ which contradicts the uniqueness assumption. ◀

Instead of uniformly drawing a permutation $\pi \sim S_{|V|}$ of the variables, we (equivalently) independently draw a *time* value $\pi(x) \sim U([0,1])$ uniformly for every variable x, and let π be the permutation induced by the order of the time values $\pi(x)$ for $x \in V$.

We observe that if $\pi(y_{x,c'}) < \pi(x)$ then $c' \notin \mathcal{A}(x, \pi)$. In particular, if $\pi(x) = p \in [0,1]$ then for every $c' \neq \varphi(x)$ with probability at least p we have that $c' \notin \mathcal{A}(x, \pi)$.

▶ **Lemma 8.** *For every variable x, we have*

$$\mathbb{E}_\pi \left[\log_k |\mathcal{A}(x,\pi)| \mid \pi(x) = p \right] \leq \sum_{i=0}^{k-1} \binom{k-1}{i} (1-p)^i p^{k-1-i} \log_k (1+i).$$

Proof. For the analysis, we may assume that we rule out values only by the constraints involving x and one of the variables $y_{x,c'}$ for $c' \neq \varphi(x)$. This holds since ruling out more variables can only decrease the size of $\mathcal{A}(x, \pi)$.

If the variables $y_{x,c'}$ are distinct for all $c' \neq \varphi(x)$, then the right hand side of the lemma's statement is exactly the expected size of $\mathcal{A}(x, \pi)$, conditioned on $\pi(x) = p$. This is simply the expectation of $\log_k(1+i)$ where $i \sim Binomial(k-1, 1-p)$ is a binomial random variable.

Generally, let $A_{c'}$ be the indicator for the event that $\pi(y_{x,c'}) > p$. We need to upper bound $\mathbb{E}[\log_k \left(1 + \sum_{c' \neq \varphi(x)} A_{c'}\right)]$. Let $A'_{c'}$ be independent Bernoulli random variables with probability $(1-p)$ to be 1 and probability p to be 0. By concavity of the function $\log_k(1+z)$ and Jensen's inequality it follows that $\mathbb{E}[\log_k \left(1 + \sum_{c' \neq \varphi(x)} A_{c'}\right)] \leq \mathbb{E}[\log_k \left(1 + \sum_{c' \neq \varphi(x)} A'_{c'}\right)]$, which concludes our proof. The complete proof of the last statement appears in [9] as Lemma A.1. ◀

Denote by $S'_{k,2} := \int_0^1 \sum_{i=0}^{k-1} \binom{k-1}{i} (1-p)^i p^{k-1-i} \log_k(1+i)\, dp$. By Lemma 8, $\mathbb{E}[\log_k |\mathcal{A}(x,\pi)|] = \int_0^1 \mathbb{E}_\pi [\log_k |\mathcal{A}(x,\pi)| \mid \pi(x) = p]\, dp \leq S'_{k,2}$, this concludes the analysis of the Feder-Motwani PPZ-type algorithm.

▶ **Theorem 9** ([4])**.** *The success probability of a PPZ iteration is at least $k^{-S'_{k,2}}$.*

2.4 The PPSZ-type algorithm of Hertli et al.

In the PPSZ algorithm analysed in [9] more involved D-implications are considered. In the analysis for $D = 1$, we noticed that for every variable x and every value $c' \neq \varphi(x)$ there exists some variable $y_{x,c'}$ such that if $\pi(y_{x,c'}) < \pi(x)$ then $c' \notin \mathcal{A}(x, \pi)$.

We say that a variable y is *decided* with respect to some partial assignment α_0 if $|\mathcal{A}(y, \alpha_0)| = 1$, i.e., if α_0 already D-implies the correct value of y in φ. The main observation is that if in time $p := \pi(x)$ the variable $y_{x,c'}$ is decided then $c' \notin \mathcal{A}(x, \pi)$. The variable $y_{x,c'}$ is necessarily decided if $\pi(y_{x,c'}) < p$ but can also be decided if it is yet to appear in the permutation. Thus, the probability of $y_{x,c'}$ being decided at time p is strictly larger than p.

We give an intuitive reasoning for the probability of a variable being decided. Denote by $q_k(p)$ the probability that a variable x is decided by time p. The variable x is decided by time p if $\pi(x) < p$ or alternatively if for every $c' \neq \varphi(x)$ the variable $y_{x,c'}$ is by itself decided at time p. In particular, $q_k(p)$ is a solution to the recurrence $q_k(p) = p + (1-p)q_k(p)^{k-1}$. We thus denote by $q_k(p)$ the smallest non-negative real solution to that recurrence, it can be analytically computed for every k as it is simply a root of a polynomial.

This intuitive argument is of course not complete and lacks many technical details. Nevertheless, this statement does hold, and the following strengthening of Lemma 8 and Theorem 9 are proven in [9].

▶ **Lemma 10** ($A.1$ in [9]). *For every variable x, we have*

$$\mathbb{E}_\pi \left[\log_k |\mathcal{A}(x,\pi)| \;\middle|\; \pi(x) = p \right] \leq \sum_{i=0}^{k-1} \binom{k-1}{i} (1 - q_k(p))^i \, q_k(p)^{k-1-i} \log_k(1+i) .$$

Denote by $S_{k,2} := \int_0^1 \sum_{i=0}^{k-1} \binom{k-1}{i} (1 - q_k(p))^i \, q_k(p)^{k-1-i} \log_k(1+i) \, dp$.

▶ **Theorem 11** (Correctness of [9]). *Let F be a Unique $(k,2)$-CSP formula, then for every variable x it holds that $\mathbb{E}_\pi [\log_k |\mathcal{A}(x,\pi)|] \leq S_{k,2} + \varepsilon_D$, where ε_D is some error parameter that depends only on D and goes to 0 as D goes to infinity.*

3 Faster Unique $(k, 2)$-CSP algorithm

On a very high-level, our algorithm combines Hertli et al.'s PPSZ (Section 2.2) with the BE algorithm (Section 2.1). We begin by illustrating our idea intuitively (initially ignoring some crucial technical details to be discussed later). Consider a run of the PPSZ algorithm, as described in Section 2.2. For the early variables in the permutation π, it is very likely that $|\mathcal{A}(x,\pi)| = k$, since α_0 assigns values to very few variables. On the other hand, for the last variables in the permutation, it is very likely that $|\mathcal{A}(x,\pi)| = 1$. It turns out that in any point throughout the run of a PPSZ iteration, the sizes $|\mathcal{A}(x,\alpha_0)|$ for the remaining variables $x \in V \setminus V(\alpha_0)$ are quite concentrated. Furthermore, after most of the variables have $|\mathcal{A}(x,\alpha_0)| \approx k' < k$ the remaining portion of the PPSZ iteration strongly resembles a PPSZ algorithm for $(k',2)$-CSP formulas. As we see in Table 1, for $k < 5$ PPSZ behaves worse on $(k,2)$-CSP formulas than the BE algorithm.

Thus, in our algorithm, we begin with an iteration of PPSZ but halt it somewhere in the middle of the permutation when the sizes $|\mathcal{A}(x,\alpha_0)|$ are concentrated in $1, 2, 3, 4$. At that point, we use the extended BE algorithm shown in Section 2.1.

We set some parameter $t \in [0,1]$ to be chosen later and consider the following algorithm.

Algorithm 2 Our algorithm.

Pick a uniform random permutation π of the set V of variables;
Denote by $\pi_{<t}$ the prefix of π of size $t|V|$ and by $V_{<t}$ the variables appearing in it;
Set $\varphi = \emptyset$;
for $x \in V_{<t}$ *in the order dictated by* $\pi_{<t}$ **do**
    Draw $c \sim U(\mathcal{A}(x,\varphi))$ uniformly;
    Set $\varphi(x) := c$;
Run the extended BE algorithm on the remaining CSP F;
Return the solution φ;

Consider an iteration of Algorithm 2. Denote by R_i the number of variables that appeared in $V_{<t}$ and had $|\mathcal{A}(x,\pi)| = i$. Denote by φ' the partial assignment constructed by time t. Let B_i be the number of variables that did not appear in $V_{<t}$ and had $|\mathcal{A}(x,\varphi')| = i$.

▶ **Lemma 12.** *The success probability of Algorithm 2 is* $\prod_{i=1}^{k} i^{-R_i} \cdot \prod_{i=5}^{k} \left(\frac{4}{i}\right)^{B_i}$.

Proof. The probability of all PPSZ assignments to be correct is $\prod_{i=1}^{k} i^{-R_i}$, as follows from Section 2.2. The probability of the random down-sampling to not rule out the correct assignments is $\prod_{i=5}^{k} \left(\frac{4}{i}\right)^{B_i}$. ◀

▶ **Lemma 13.** *The running time of Algorithm 2 is* $O\left(1.3645^{B_3} \cdot 1.8072^{B_4+\ldots+B_k}\right)$.

Proof. The running time of the (partial) PPSZ iteration is polynomial. The running time of the BE algorithm is $O(1.3645^{n_3} \cdot 1.8072^{n_4})$. ◀

Note that all R_i and B_i are fully determined by the choice of π. Thus, for a specific choice of π, if we repeatedly run Algorithm 2 with π we expect finding a solution after $\left(\prod_{i=1}^{k} i^{-R_i} \cdot \prod_{i=5}^{k} \left(\frac{4}{i}\right)^{B_i}\right)^{-1}$ iterations. In particular, after

$$\left(\prod_{i=1}^{k} i^{-R_i} \cdot \prod_{i=5}^{k} \left(\frac{4}{i}\right)^{B_i}\right)^{-1} \cdot O\left(1.3645^{B_3} \cdot 1.8072^{B_4+\ldots+B_k}\right) = \prod_{i=1}^{k} i^{R_i} \cdot \prod_{i} BE(i)^{B_i}$$

computational steps. At this point, we would like to bound the expected running time when picking a random π with $\prod_{i=1}^{k} i^{\mathbb{E}[R_i]} \cdot \prod_i BE(i)^{\mathbb{E}[B_i]}$. Unfortunately, as we consider the running time and not a success probability (as in the PPSZ algorithm), we need an inequality of the opposite direction to Jensen's inequality. Fortunately, this inequality *essentially still holds* in this case.

▶ **Lemma 14** (Wrong direction Jensen's inequality is still kind-of right)**.** *Let* $\mathcal{A}$ *be an algorithm with expected running time* 2^X *conditioned on the value of a random variable* X*. There exists an algorithm* $\mathcal{A}'$ *that successfully executes* $\mathcal{A}$ *with probability at least* 0.99 *and has an expected running time of* $O\left(2^{\mathbb{E}[X]} \cdot \mathbb{E}[X]\right)$.

Proof. We apply Markov's inequality twice. First, to observe that

$$\Pr(X > \mathbb{E}[X] + 1) \leq \frac{1}{1 + \frac{1}{\mathbb{E}[X]}} = 1 - \frac{1}{\mathbb{E}[X] + 1}.$$

Hence, if we run $\mathcal{A}$ independently for $6(\mathbb{E}[X] + 1)$ times, then with probability at least $1 - e^{-6} > 1 - \frac{1}{200}$ at least one of these runs has $X \leq \mathbb{E}[X] + 1$. Second, conditioned on any value of X, with probability at least $1 - \frac{1}{200}$ algorithm $\mathcal{A}$ finishes in less than $200 \cdot 2^X$ computational steps. Thus, by union bound, if we run algorithm $\mathcal{A}$ for $6(\mathbb{E}[X] + 1)$ times, and terminate each run after $400 \cdot 2^{\mathbb{E}[X]}$ computational steps, then at least one run of $\mathcal{A}$ finishes with probability at least 0.99. ◀

▶ **Corollary 15.** *We find a satisfying assignment with probability greater than* 0.99 *in time*

$$O^\star\left(\prod_{i=1}^{k} i^{\mathbb{E}[R_i]} \cdot \prod_{i} BE(i)^{\mathbb{E}[B_i]}\right). \qquad (\star)$$

We give a simpler analysis leading to slightly sub-optimal bounds. In Section 4 we sketch the possible improvements to the analysis presented here, and also present a clear limit to the improvements that can be achieved by this algorithm.

We slightly abuse notation by equating the numbers R_i, B_i with the sets of variables they are counting. Let x be a variable. For the analysis we can assume that the algorithm rules out values for x only due to the constraints involving x and some $y_{x,c'}$. This holds as ruling out more values can only improve the success probability of each iteration.

We first consider the case in which the variables $y_{x,c'}$ for every $c' \neq \varphi(x)$ are all distinct.

▶ **Lemma 16.** *For each variable x, we have*

$$\mathbb{E}\left[\sum_{i=1}^{k} Pr\left(x \in R_i\right) \cdot \log_k i\right] \leq \int_0^t \sum_{i=0}^{k-1} \binom{k-1}{i} \left(1 - q_k\left(p\right)\right)^i q_k\left(p\right)^{k-1-i} \log_k\left(1+i\right) dp.$$

Proof. This follows immediately from Lemma 10. ◀

▶ **Lemma 17.** *Let x be a variable for which $y_{x,c'}$ are distinct for all $c' \neq \varphi(x)$. Then,*

$$\mathbb{E}\left[\sum_{i=3}^{k} Pr\left(x \in B_i\right) \cdot \log_k EB\left(i\right)\right] \leq \left(1-t\right) \cdot \sum_{i=3}^{k} \binom{k-1}{i} \left(1-t\right)^i t^{k-1-i} \cdot \log_k EB\left(i\right).$$

Proof. For simplicity, we analyse this part with a PPZ-type analysis (rather than PPSZ-type one), this is further discussed in Section 4. The probability that $x \notin V_{<t}$ is $(1-t)$, and the probability that exactly i out of the $(k-1)$ variables $y_{x,c'}$ do not appear in $V_{<t}$ is $\binom{k-1}{i}\left(1-t\right)^i t^{k-1-i}$. ◀

Denote by

$$\begin{aligned}\mathrm{cost}(k,t) := &\int_0^t \sum_{i=0}^{k-1} \binom{k-1}{i} \left(1 - q_k\left(p\right)\right)^i q_k\left(p\right)^{k-1-i} \log_k\left(1+i\right) dp \\ &+ \left(1-t\right) \cdot \sum_{i=3}^{k} \binom{k-1}{i} \left(1-t\right)^i t^{k-1-i} \cdot \log_k EB\left(i\right).\end{aligned}$$

If all variables had completely distinct $y_{x,c'}$'s, then by Lemma 16 and Lemma 17 we would have that $(\star)$ and in particular the running time of our algorithm is bounded by $O\left(k^{\mathrm{cost}(k,t)n}\right)$ for any choice of t. This would give us $O(2.22936^n)$ for $k = 5, t = 0.23$ and $O(2.64001^n)$ for $k = 6, t = 0.35$. We now deal with the case in which these are not distinct.

In the proof of Lemma 8 we faced the same problem and solved it by a simple application of Jensen's inequality to the concave function $\log_k(1+i)$. Unfortunately, the function $\log_k EB\left(i\right)$ is not concave (for $i = 1, \ldots, k$) due to its values on $i = 1, 2$. Indeed, the left-hand side of the inequality in Lemma 17 is higher than the right-hand side if these variables are not distinct. On the other hand, when these variables are not distinct then the term of Lemma 16 is much smaller.

Consider the expression

$$\mathbb{E}\left[\sum_{i=1}^{k} Pr\left(x \in R_i\right) \cdot \log_k i + \sum_{i=3}^{k} Pr\left(x \in B_i\right) \cdot \log_k EB\left(i\right)\right] \qquad (\star\star)$$

again. This time, we will assume that the variables $y_{x,c'}$ are not all distinct. Denote by $k' := |\{y_{x,c'} \mid c' \neq \varphi(x)\}| < k-1$ the number of such distinct variables, and by $j_1, \ldots, j_{k'}$ their cardinalities (note that $\sum_{i=1}^{k'} j_i = k-1$). Consider the following expression.

$$\mathbb{E}\Big[\int_0^t \sum_{b_1,\ldots,b_{k'}\in\{0,1\}} q_k(p)^{k'-\sum_{i=1}^{k'} b_i}\cdot(1-q_k(p))^{\sum_{i=1}^{k'} b_i}\cdot\log_k\left(1+\sum_{i=1}^{k'} j_i b_i\right)dp \quad (\star\star\star)$$
$$+(1-t)\cdot\sum_{b_1,\ldots,b_{k'}\in\{0,1\}} t^{k'-\sum_{i=1}^{k'} b_i}\cdot(1-t)^{\sum_{i=1}^{k'} b_i}\cdot\log_k EB\left(1+\sum_{i=1}^{k'} j_i b_i\right)\Big].$$

Expression $(\star\star\star)$ is a generalized form of $\text{cost}(k,t)$ and thus upper bounds $(\star\star)$ by the same arguments. Completely analysing the behaviour of Expression $(\star\star\star)$ for different partitions is rather technically involved and thus we simply enumerate over the few possible cases (for small values of k). In Section 4 we further discuss the possible improvements to the analysis of this section.

▶ **Theorem 18.** *We solve Unique* $(6,2)$*-CSP in* $O(2.641^n)$ *time.*

Proof. For the choice $t = 0.37$ we have that $\text{cost}(6,0.35) = \log_6(2.64001)$, this choice of t minimizes $\text{cost}(6,t)$. We verify that for $t = 0.35$ Expression $(\star\star\star)$ is always lower than $\text{cost}(6,0.35)$ and thus finish, as this implies that for every variable Expression $(\star\star)$ is bounded by $\text{cost}(6,0.35)$.

- For the partition $(j_1,j_2,j_3,j_4) = (2,1,1,1)$ the value of $(\star\star\star)$ in $t = 0.35$ is $\log_6(2.62023)$.
- For the partition $(j_1,j_2,j_3) = (2,2,1)$ the value of $(\star\star\star)$ in $t = 0.35$ is $\log_6(2.61171)$.
- For the partition $(j_1,j_2,j_3) = (3,1,1)$ the value of $(\star\star\star)$ in $t = 0.35$ is $\log_6(2.58391)$.
- For the partition $(j_1,j_2) = (3,2)$ the value of $(\star\star\star)$ in $t = 0.35$ is $\log_6(2.60366)$.
- For the partition $(j_1,j_2) = (4,1)$ the value of $(\star\star\star)$ in $t = 0.35$ is $\log_6(2.54819)$.
- For the partition $(j_1) = (5)$ the value of $(\star\star\star)$ in $t = 0.35$ is $\log_6(2.55566)$. ◀

▶ **Theorem 19.** *We solve Unique* $(5,2)$*-CSP in* $O(2.232^n)$ *time.*

Proof. For the choice $t = 0.23$ we have that $\text{cost}(5,0.23) = \log_5(2.22936)$, this choice of t minimizes $\text{cost}(5,t)$. This time, unfortunately, for $t = 0.23$ Expression $(\star\star\star)$ is not always lower than $\text{cost}(5,0.23)$. In particular, it is for every partition except of $(j_1) = 4$.

- For the partition $(j_1,j_2,j_3) = (2,1,1)$ the value of $(\star\star\star)$ in $t = 0.23$ is $\log_5(2.21658)$.
- For the partition $(j_1,j_2) = (2,2)$ the value of $(\star\star\star)$ in $t = 0.23$ is $\log_5(2.21983)$.
- For the partition $(j_1,j_2) = (3,1)$ the value of $(\star\star\star)$ in $t = 0.23$ is $\log_5(2.20499)$.
- For the partition $(j_1) = (4)$ the value of $(\star\star\star)$ in $t = 0.23$ is $\log_5(2.24925)$.

Denote by α the fraction of variables for which the $y_{x,c'}$ variables are all the same (i.e., variables with the only problematic partition). With the choice $t = 0.23$ the running time of our algorithm on a formula is $O\left(\left(2.22936^{(1-\alpha)}\cdot 2.24925^{\alpha}\right)^n\right)$. On the other hand, if α is large we can simply run the regular PPSZ algorithm and gain much. We see that by setting $t = 1$ and computing Expression $(\star\star\star)$ for the same partition $(j_1) = (4)$, gives a value of $\log_5(2.01077)$. Thus, running regular PPSZ would give us a running time of $O\left(\left(2.25303^{(1-\alpha)}\cdot 2.01077^{\alpha}\right)^n\right)$. We can therefore try both options ($t = 0.23$ or $t = 1$) simultaneously and thus get the running time of the faster one. Both expressions balance at $\alpha = 0.08612$, giving us a running time of $O(2.23107^n)$. ◀

Computation identical to this of Theorem 18 gives running time of $O(3.042^n)$ for $k = 7$ with $t = 0.44$.

4 Improvements and Limitations

In this section we sketch a possible improvement to the analysis of Section 3. The purpose of this section is not to tighten the upper bound but to explain the limitations of our algorithm and to convince that even with a tight analysis of Algorithm 2 it will achieve running times that are only slightly better than these we get in Section 3. In particular, if getting $O\left((2-\varepsilon)^n\right)$ time for $(5,2)$-CSP is possible, new algorithmic tools are likely required.

Consider the expression $\prod_{i=1}^{k} i^{\mathbb{E}[R_i]} \cdot \prod_i EB(i)^{\mathbb{E}[B_i]}$ $(\star)$ proven in Section 3 to upper bound the running time of our algorithm. In Lemma 16 we give a likely tight bound for the term involving $\mathbb{E}[R_i]$, yet in Lemma 17 we settle for a PPZ-type bound for $\mathbb{E}[B_i]$ in which we consider only the events in which the variables $y_{x,c'}$ themselves appear before time t and not the events in which they are decided by that time. The reason for this discrepancy becomes clear in the rest of the analysis. Due to the non-concave objective function in i, we can no longer assume independence between the events of each $y_{x,c'}$ being decided. Nevertheless, later in Theorem 18 and Theorem 19 we observe that while the term involving $\mathbb{E}[B_i]$ indeed gets worse with dependencies, the other term involving $\mathbb{E}[B_i]$ gets significantly better with them and thus can cover for those dependencies. Ideally, then, the simple PPZ-type bound of t in Lemma 17 can be replaced with the PPSZ-type bound of $q_k(t)$ in the total bound. This would result in the following tighter cost function.

$$\begin{aligned}\tilde{\text{cost}}(k,t) := &\int_0^t \sum_{i=0}^{k-1} \binom{k-1}{i} \left(1-q_k\left(p\right)\right)^i q_k\left(p\right)^{k-1-i} \log_k\left(1+i\right) dp \\ &+ \left(1-t\right) \cdot \sum_{i=3}^{k} \binom{k-1}{i} \left(1-q_k\left(t\right)\right)^i q_k\left(t\right)^{k-1-i} \cdot \log_k EB\left(i\right).\end{aligned}$$

With this ideal cost function, we would get running times of $O\left(2.223^n\right)$ for Unique $(5,2)$-CSP (for $t = 0.32$) and $O\left(2.628^n\right)$ for Unique $(6,2)$-CSP (for $t = 0.46$).

5 Conclusions and Open Problems

In Section 3 we presented an algorithm for Unique $(k,2)$-CSP with a running time of $O(2.232^n)$ for $k = 5$. In Section 4 we argued that even with an ideal analysis, the bound we get for $k = 5$ is only the slightly better $O(2.223^n)$. Thus, it remains open and would likely require new algorithmic tools to show that $(5,2)$-CSP can be solved in $O\left((2-\varepsilon)^n\right)$ time, or alternatively to rule out the existence of such algorithm by reductions to popular conjectures. More generally, we raise the following open problem.

▶ **Open Problem.** *What is the maximal k such that $(k,2)$-CSP can be solved in $O\left((2-\varepsilon)^n\right)$ time?*

The main algorithmic observation in this paper is in fact a general insight regarding the behaviour of PPSZ-type algorithms. The Beigel-Eppstein algorithm [1] only works for $(k,2)$-CSP. On the other hand, the PPSZ-type algorithm [9] generalizes to $b > 2$ and is in fact currently the fastest algorithm for Unique (a,b)-CSP with $b > 2$ and any a. Using the tools we introduced in this paper, it should be possible to turn any faster algorithm for (a,b)-CSP for a specific (a,b) into a faster (a',b)-CSP algorithm for all $a' > a$.

Another follow-up question is whether our algorithm can be generalized to the non-unique $(k,2)$-CSP case.

References

1. Richard Beigel and David Eppstein. 3-coloring in time $O(1.3289^n)$. *Journal of Algorithms*, 54(2):168–204, 2005.
2. Andreas Björklund, Thore Husfeldt, and Mikko Koivisto. Set partitioning via inclusion-exclusion. *SIAM Journal on Computing*, 39(2):546–563, 2009.
3. Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. The complexity of satisfiability of small depth circuits. In *International Workshop on Parameterized and Exact Computation*, pages 75–85. Springer, 2009.
4. Tomás Feder and Rajeev Motwani. Worst-case time bounds for coloring and satisfiability problems. *Journal of Algorithms*, 45(2):192–201, 2002.
5. Fedor V Fomin and Petteri Kaski. Exact exponential algorithms. *Communications of the ACM*, 56(3):80–88, 2013.
6. F.V. Fomin and D. Kratsch. *Exact Exponential Algorithms*. Texts in Theoretical Computer Science. An EATCS Series. Springer Berlin Heidelberg, 2010.
7. Thomas Dueholm Hansen, Haim Kaplan, Or Zamir, and Uri Zwick. Faster k-SAT algorithms using biased-PPSZ. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 578–589, 2019.
8. Timon Hertli. 3-SAT faster and simpler - unique-SAT bounds for PPSZ hold in general. *SIAM J. Comput.*, 43(2):718–729, 2014. Announced at FOCS'11.
9. Timon Hertli, Isabelle Hurbain, Sebastian Millius, Robin A Moser, Dominik Scheder, and May Szedlák. The ppsz algorithm for constraint satisfaction problems on more than two colors. In *International Conference on Principles and Practice of Constraint Programming*, pages 421–437. Springer, 2016.
10. Eugene L Lawler. A note on the complexity of the chromatic number problem, 1976.
11. Liang Li, Xin Li, Tian Liu, and Ke Xu. From k-sat to k-csp: Two generalized algorithms. *arXiv preprint*, 2008. `arXiv:0801.3147`.
12. Shibo Li and Dominik Scheder. Impatient ppsz–a faster algorithm for csp. *arXiv preprint*, 2021. `arXiv:2109.02795`.
13. Burkhard Monien and Ewald Speckenmeyer. Solving satisfiability in less than 2n steps. *Discrete Applied Mathematics*, 10(3):287–295, 1985.
14. Ramamohan Paturi, Pavel Pudlák, Michael E. Saks, and Francis Zane. An improved exponential-time algorithm for k-SAT. In *39th Annual Symposium on Foundations of Computer Science, FOCS '98*, pages 628–637, 1998. `doi:10.1109/SFCS.1998.743513`.
15. Ramamohan Paturi, Pavel Pudlák, Michael E Saks, and Francis Zane. An improved exponential-time algorithm for k-SAT. *Journal of the ACM (JACM)*, 52(3):337–364, 2005.
16. Dominik Scheder. Ppz for more than two truth values-an algorithm for constraint satisfaction problems. *arXiv preprint*, 2010. `arXiv:1010.5717`.
17. T Schoning. A probabilistic algorithm for k-SAT and constraint satisfaction problems. In *40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)*, pages 410–414. IEEE, 1999.
18. Patrick Traxler. The time complexity of constraint satisfaction. In *International Workshop on Parameterized and Exact Computation*, pages 190–201. Springer, 2008.
19. Gerhard J Woeginger. Exact algorithms for NP-hard problems: A survey. In *Combinatorial optimization – eureka, you shrink!*, pages 185–207. Springer, 2003.
20. Or Zamir. Breaking the 2^n barrier for 5-coloring and 6-coloring, 2020.