Algorithms for the Diverse-k-SAT Problem: The Geometry of Satisfying Assignments

Per Austrin
KTH Royal Institute of Technology, Stockholm, Sweden

Ioana O. Bercea
KTH Royal Institute of Technology, Stockholm, Sweden

Mayank Goswami
Queens College, City University of New York, NY, USA

Nutan Limaye
IT University of Copenhagen, Denmark

Adarsh Srinivasan
Rutgers University, Piscataway, NJ, USA

Abstract

Given a k-CNF formula and an integer $s \geq 2$, we study algorithms that obtain s solutions to the formula that are as dispersed as possible. For $s = 2$, this problem of computing the *diameter* of a k-CNF formula was initiated by Creszenzi and Rossi, who showed strong hardness results even for $k = 2$. The current best upper bound [Angelsmark and Thapper '04] goes to 4^n as $k \to \infty$. As our first result, we show that this quadratic blow up is not necessary by utilizing the Fast-Fourier transform (FFT) to give a $O^*(2^n)$ time exact algorithm for computing the diameter of any k-CNF formula.

For $s > 2$, the problem was raised in the SAT community (Nadel '11) and several heuristics have been proposed for it, but no algorithms with theoretical guarantees are known. We give exact algorithms using FFT and clique-finding that run in $O^*(2^{(s-1)n})$ and $O^*(s^2|\Omega_{\mathbf{F}}|^{\omega\lceil s/3\rceil})$ respectively, where $|\Omega_{\mathbf{F}}|$ is the size of the solutions space of the formula $\mathbf{F}$ and ω is the matrix multiplication exponent.

However, current SAT algorithms for *finding one solution* run in time $O^*(2^{\varepsilon_k n})$ for $\varepsilon_k \approx 1 - \Theta(1/k)$, which is much faster than all above run times. *As our main result*, we analyze two popular SAT algorithms - PPZ (Paturi, Pudlák, Zane '97) and Schöning's ('02) algorithms, and show that in time $\text{poly}(s)O^*(2^{\varepsilon_k n})$, they can be used to approximate diameter as well as the dispersion ($s > 2$) problem. While we need to modify Schöning's original algorithm for technical reasons, we show that the PPZ algorithm, without any modification, samples solutions in a geometric sense. We believe this geometric sampling property of PPZ may be of independent interest.

Finally, we focus on diverse solutions to NP-complete optimization problems, and give bi-approximations running in time $\text{poly}(s)O^*(2^{\varepsilon n})$ with $\varepsilon < 1$ for several problems such as Maximum Independent Set, Minimum Vertex Cover, Minimum Hitting Set, Feedback Vertex Set, Multicut on Trees and Interval Vertex Deletion. For all of these problems, all existing exact methods for finding optimal diverse solutions have a runtime with at least an exponential dependence on the number of solutions s. Our methods show that by relaxing to bi-approximations, this dependence on s can be made polynomial.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

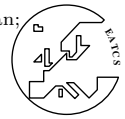
Keywords and phrases Exponential time algorithms, Satisfiability, k-SAT, PPZ, Schöning, Dispersion, Diversity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2025.14

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: https://arxiv.org/abs/2408.03465 [7]

52nd International Colloquium on Automata, Languages, and Programming (ICALP 2025).
Editors: Keren Censor-Hillel, Fabrizio Grandoni, Joël Ouaknine, and Gabriele Puppis
Article No. 14; pp. 14:1–14:17



Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Funding *Ioana O. Bercea*: Received funding from Basic Algorithms Research Copenhagen(BARC), supported by VILLUM Foundation Grants 16582 and 54451.
Mayank Goswami: Supported by NSF grant CCF-2503086.
Nutan Limaye: Received funding from the Independent Research Fund Denmark (grant agreement No. 10.46540/3103-00116B) and is also supported by the Basic Algorithms Research Copenhagen (BARC), funded by VILLUM Foundation Grants 16582 and 54451. Was also supported by Digital Research Centre Denmark, project P40.
Adarsh Srinivasan: Supported by the National Science Foundation under grants CCF-2313372 and CCF-2443697 and a grant from the Simons Foundation, Grant Number 825876, Awardee Thu D. Nguyen, and received funding from Basic Algorithms Research Copenhagen(BARC), supported by VILLUM Foundation Grants 16582 and 54451.

1 Introduction

In this work, we start by asking a simple question: what is the complexity of computing the diameter of a k-SAT solution space? That is, given a satisfiable k-CNF formula, we want to output two satisfying assignments with maximum Hamming distance between them. More generally, what if we want *multiple* satisfying assignments that are maximally far apart? One can also think of this as finding a binary code with optimal rate/distance tradeoff, where each codeword must satisfy the given k-CNF formula. We give exact and approximate exponential time algorithms for these problems and show that existing well-known algorithms for finding one solution can be leveraged to output multiple, reasonably far apart, solutions.

Crescenzi and Rossi [18] formulated the diameter computation problem for general Constraint Satisfaction Problems (CSPs), under the name Maximum Hamming Distance. They studied the approximability of the problem and gave a complete classification based on Schaefer's criteria for the satisfiability of CSPs [46]. In particular, they also showed that the diameter problem is NP-hard even for 2-SAT.[1] On the constructive side, Angelsmark and Thapper [4] gave an algorithm that outputs a diameter pair in polynomial space and $(2a_k)^n$ time, whenever there exists an $(a_k)^n$ time algorithm for finding one satisfying assignment. Under standard complexity assumptions (SETH), $a_k \to 2$ as $k \to \infty$, so the above approach is unlikely to run in time better than[2] $O^*(4^n)$.

This already raises the interesting question of the optimal running time needed for finding a diameter pair (i.e., its exponential complexity [13]). In the case of graphs, it is known that quadratic blow-up in time is unavoidable, assuming the Orthogonal Vectors Hypothesis [52, 3]. Should we also expect a quadratic blow-up in time for diameter of k-SAT? We first show that this is not the case: using a Fourier analytical approach, we show how to compute a diameter pair deterministically in $O^*(2^n)$ time (Theorem 6).

Dispersion. The problem of computing $s > 2$ diverse satisfying assignments to a k-CNF formula was explicitly raised by Nadel [40]. Generating diverse solutions has many applications [10, 1, 8], and several other works have focused on finding multiple solutions to either SAT or constraint programming [2, 43, 29, 44, 38, 23, 6]. However, all of the above works are heuristic in nature, and we could not find any algorithm for dispersed solutions to k-SAT with provable guarantees. Our work provides the first exact and approximate algorithms for computing diverse solutions to a k-CNF formula.

[1] They in fact show that it is PolyAPX-hard. Moreover, while not explicitly stated, their reduction immediately gives an optimal inapproximability of $O(n^{1-\epsilon})$ for the diameter of a 2-CNF formula.
[2] We use the O^* notation to hide polynomial factors in n.

There are many different ways to define the *dispersion* for a set of points (see Table 1 in [36]). We consider two most popular measures of dispersion: minimum pairwise distance and sum of pairwise distances (the latter is equivalent to average pairwise distance). We will use d_H to denote the Hamming distance. By the dispersion problem, we mean given a k-CNF formula $\mathbf{F}$ and an integer $s \geq 2$, return a set S of s satisfying assignments to $\mathbf{F}$ that maximize $\textsc{minPD}(S) := \min_{z_1,z_2 \in S} d_H(z_1, z_2)$ or $\textsc{sumPD}(S) := \frac{1}{2}\sum_{z_1,z_2 \in S} d_H(z_1, z_2)$. If the k-CNF formula does not have s distinct satisfying assignments, we allow the algorithm to return a multiset. Unless stated otherwise, our results will be for the minimum version of dispersion.

Exact algorithms. We show that we can extend our Fourier analytical approach for diameter to dispersion, obtaining an exact algorithm in time $O^*(2^{(s-1)n})$ (Theorem 9). Furthermore, for $s \geq 6$ we also get a faster algorithm based on clique finding (Theorem 10), that runs in time $O^*(s^2|\Omega_{\mathbf{F}}|^{\omega\lceil s/3\rceil})$, where $\Omega_{\mathbf{F}}$ is the set of satisfying assignments of the formula $\mathbf{F}$ and $\omega \leq 2.38$ is the matrix multiplication exponent [53].

Faster approximations. Even with our improvements, the above exact algorithms still run in $O^*(2^{csn})$ time for $c < 1$. What if we allow approximations? Two questions arise:

- Can one obtain a bound of the form $f(s)O^*(2^n)$? If so, must f have exponential dependence on s, or can f be made polynomial in s?
- The current fastest k-SAT algorithms for finding *one solution* run in time $O^*(2^{\varepsilon_k n})$ for $\varepsilon_k = 1 - \Theta(1/k)$. Can one get a bound of the form $f(s)O^*(2^{\varepsilon_k n})$? Thus, the best runtime for finding s dispersed solutions that one could hope for is $\text{poly}(s)O^*(2^{\varepsilon_k n})$, as this is roughly the time taken to find any set of s solutions. Can we achieve this?

Main result, informal. There exist randomized algorithms with a run time of $\text{poly}(s)O^*(2^{\varepsilon_k n})$ that, given a k-CNF formula $\mathbf{F}$ on n variables and a parameter s, return a set S of s many satisfying assignments that approximately maximize $\textsc{minPD}(S)$ and $\textsc{sumPD}(S)$. Moreover, for several *optimization* problems, there exist algorithms with a similar runtime that are bi-approximations, i.e., return approximately-optimal solutions that are also approximately-maximally-diverse.

In addition to these results being a step towards bridging the gap between the theory and practice of finding diverse solutions, what is surprising is that the way we arrive at them reveals novel interesting aspects of two extremely well-studied algorithms for finding one solution to a given formula: PPZ and Schöning's algorithm.

PPZ and Schöning's algorithms. The complexity of the k-SAT problem has a long and rich history [34, 35, 13, 21]. In a foundational work, Paturi, Pudlák, and Zane [42] presented a remarkably simple and elegant randomized algorithm for k-SAT. Their algorithm runs in time $O^*\left(2^{(1-1/k)n}\right)$ and outputs a satisfying solution with probability $1-o(1)$ if one exists. A few years after that, Schöning [51] developed another surprisingly simple random walk-based algorithm running in time $O^*\left(2^{(1-1/(k\ln 2))n}\right)$,[3] which runs faster than the PPZ algorithm for all k. With time, these approaches have been reanalyzed and sometimes improved in a variety of technically subtle and involved ways [33, 11, 41, 31, 30, 39, 49, 28, 47, 48], including the PPSZ algorithm by Paturi, Pudlák, Saks and Zane [41], which is the current fastest algorithm for k-SAT.

[3] The run-time of Schöning's algorithm is normally presented as $O^*\left((2(1-1/k))^n\right)$, which we have rewritten for ease of comparison with PPZ.

In our work, we ask whether PPZ and Schöning's can exploit the global geometry of the solution space and go beyond finding just one satisfying assignment. Namely, can they be used to *approximate* the diameter and the dispersion for k-SAT? We remark that the main result above is not a black-box result that uses *any SAT solver* - we only know how to use PPZ and Schöning's algorithms for this purpose. To familiarize the reader with these two algorithms, we provide their pseudocodes next.

Algorithm 1 PPZ.

Input: A k-CNF formula $\mathbf{F}$ over n variables
repeat $n^{O(1)} \cdot 2^{(1-1/k)n}$ **times**
  Sample $\pi \sim S_n$, $y \sim \{0,1\}^n$ u.a.r.
  for $i \in [n]$ **do**
    if $\mathbf{F}$ *contains the unit clause* $(x_{\pi(i)})$ **then**
      $u_{\pi(i)} \leftarrow 1$
    if $\mathbf{F}$ *contains the unit clause* $(\bar{x}_{\pi(i)})$ **then**
      $u_{\pi(i)} \leftarrow 0$
    else
      $u_{\pi(i)} \leftarrow y_{\pi(i)}$
    $\mathbf{F} \leftarrow \mathbf{F}|_{x_{\pi(i)}=u_{\pi(i)}}$
  if u *satisfies* $\mathbf{F}$ **then**
    return $u = (u_1, u_2, \ldots, u_n)$
return *"not satisfiable"*

Algorithm 2 Schöning.

Input: A k-CNF formula $\mathbf{F}$ over n variables
repeat $n^{O(1)} \cdot 2^{(1-\frac{1}{k \ln 2})n}$ **times**
  Sample $y \sim \{0,1\}^n$
  repeat $3n$ **times**
    if y *satisfies* $\mathbf{F}$ **then**
      return y
    else
      Let C be the first clause in $\mathbf{F}$ not satisfied by y, pick one of the k variables in C at random and flip the value that y assigns to that variable
return *"not satisfiable"*

Farthest Point Oracles. Gonzalez [24] proposed the farthest-insertion algorithm, and showed that it gives a 1/2 approximation to the minimum version of the dispersion problem: given a metric space of n points, find a set S of s points in it that maximize $\textsc{min}\mathrm{PD}(S)$. This was later extended to the sum version by [12]. The algorithm builds the set S iteratively; in the ith iteration it adds the point x_i that maximizes the minimum (resp. sum of) distance to all the points in the solution so far. Moreover, the factor 1/2 is tight assuming the Exponential Time Hypothesis (ETH), so in a sense, farthest insertion is the best possible (polynomial) algorithm for dispersion [22].

In our setting, a farthest point oracle takes as input a k-CNF formula $\mathbf{F}$ (with a set $\Omega_{\mathbf{F}}$ of satisfying assignments) and a set (or multiset) $S \subseteq \{0,1\}^n$, and outputs a satisfying assignment $z^* \in \Omega_{\mathbf{F}}$ that is "far away" from the assignments in S. Namely, for $x \in \{0,1\}^n$, $S \subseteq \{0,1\}^n$, we let $\textsc{min}\text{-}d_H(S,x) = \min_{y \in S} d_H(x,y)$ and $\textsc{sum}\text{-}d_H(S,x) = \sum_{y \in S} d_H(x,y)$. Then for some $\delta \in [0,1)$, the assignment z^* would either satisfy

$$\textsc{min}\text{-}d_H(z^*,S) \geq (1-\delta) \max_{z' \in \Omega_{\mathbf{F}}} \textsc{min}\text{-}d_H(z',S), \quad \textbf{or} \quad \textsc{sum}\text{-}d_H(z^*,S) \geq (1-\delta) \max_{z' \in \Omega_{\mathbf{F}}} \textsc{sum}\text{-}d_H(z',S),$$

for the $\textsc{min}\mathrm{PD}(S)$ and the $\textsc{sum}\mathrm{PD}(S)$ version, respectively.

In Section 2, we describe our main technical lemmas on PPZ and Schöning algorithms. This is followed by the algorithms for diameter and dispersion implied by these lemmas (Section 3). As mentioned in the informal result statement, our techniques extend to finding diverse solutions to optimization problems as well. These results are formally described in Section 4.

2 Main Technical Lemmata

Recall that we are aiming for a runtime of $\text{poly}(s)O^*(2^{\varepsilon_k n})$. The question therefore is: can we implement farthest point insertion in $O^*(2^{\varepsilon_k n})$ time? We now state the two main technical lemmas that form the core of our analysis.

▶ **Lemma 1** (PPZ performs geometric sampling). *For any $z_0 \in \{0,1\}^n$, with probability at least $\frac{1}{2n} \cdot 2^{-(1-1/k)n}$, each iteration of the PPZ algorithm outputs a satisfying assignment z^*, such that $d_H(z_0, z^*) \geq \left(1 - \frac{1}{k}\right) \cdot \max_{z' \in \Omega_{\mathbf{F}}} d_H(z_0, z')$. The iteration of PPZ does not depend on z_0.*

▶ **Lemma 2** (Modified Schöning's Algorithm is a farthest point oracle). *There exists an algorithm, running in time $O^*\left(2^{(1-1/(k \ln 2))n}\right)$ that takes a k-CNF formula $\mathbf{F}$ and $z_0 \in \{0,1\}^n$ as input and outputs a satisfying assignment z^* such that $d_H(z_0, z^*) \geq \left(1 - \frac{4(k-1)}{(k-2)^2}\right) \cdot \max_{z' \in \Omega_{\mathbf{F}}} \textsc{sum}\text{-}d_H(S, z')$. Here, z_0 is used explicitly inside the iteration.*

We sketch the proofs in Section 5. Three remarks are in order.

▶ Remark 3. Lemma 1 requires several insights into the behavior of PPZ. PPZ is not a traditional local search algorithm and it falls in the random restriction paradigm [48]. The analysis of PPZ [42] is local in nature: the authors bound the probability of arriving at a solution z that is j-isolated, meaning that exactly $n - j$ neighbors of z are also satisfying solution. This probability is then added over all satisfying assignments, resulting in the PPZ run time bound of $O^*(2^{(1-1/k)n})$. On the other hand, in Lemma 1 we are interested in bounding the probability that PPZ returns a solution that is far away from a given point z_0. The fact that PPZ, without any modifications based on z_0, returns such far-away solutions automatically was surprising to us. We leave it as an open question whether the PPZ-based, more involved, state-of-the-art algorithm of Paturi, Pudlák, Saks and Zane (PPSZ) [41], can also be shown to exhibit similar behavior.

▶ Remark 4. Unlike PPZ, we could not prove that Schöning's original algorithm works directly as an approximate farthest point oracle. Our modification of Schöning's algorithm controls both the region of starting assignments x and the length of the Schöning walk from x. Instead of Schöning's analysis that bounds the probability of finding any solution starting at a random point, we bound the probability that we find a solution far from z_0 and close to x. As a plus, in addition to giving us a farthest point oracle, this also allows us to obtain a tradeoff between runtime and approximation factors. More details can be found in Section 5.

▶ Remark 5. We investigate other promising candidate approaches for k-CNF dispersion that do not use PPZ or Schöning's algorithms. First, we show that the approach to solve dispersion problem via *uniform sampling algorithms* [50] does not necessarily give a good approximation compared to our approach, even for the diameter. Furthermore, we consider yet another promising approach via the Min-Ones problem. This problem asks for the minimum Hamming weight solution to a SAT formula [20]. While we note that the an

algorithm for the MIN-ONES problem can be used to give a 1/2 approximation of the diameter, we also observe that this approach is unlikely to be extended to finding more than two diverse solutions, as the reduction to diameter does not generalize. We refer the reader to the full version of this paper for more details [7].

Lemma 1 and Lemma 2 give us algorithms for computing a set S with maximum dispersion for both the MINPD(S) and the SUMPD(S) versions. These are stated formally in Section 3. Moreover, we get a variety of applications: diverse solutions to several optimization problems and CSPs, and reanalyzing SAT algorithms when the formula has many diverse assignments. These are presented in Section 4.

3 Results on Diameter and Dispersion

Throughout the paper, we let $\mathbf{F}$ denote a k-CNF formula on n variables (unless otherwise specified). Given such an $\mathbf{F}$, we let $\Omega_{\mathbf{F}} \subseteq \{0,1\}^n$ denote the set of satisfying assignments of $\mathbf{F}$. We start by formally defining the diameter problem. For a given formula $\mathbf{F}$, let DIAM($\mathbf{F}$) be defined as $\max_{z_1,z_2 \in \Omega_{\mathbf{F}}} \{d_H(z_1,z_2)\}$, where $\Omega_{\mathbf{F}}$ is non-empty. Note that when $\mathbf{F}$ has a unique satisfying assignment, then DIAM($\mathbf{F}$) is simply 0. On the other hand, if $\mathbf{F}$ is not satisfiable, we define DIAM($\mathbf{F}$) $=\perp$. For a set $S \subseteq \{0,1\}^n$, define MINPD(S) $:= \min_{z_1,z_2 \in S} d_H(z_1,z_2)$ and SUMPD(S) $:= \frac{1}{2}\sum_{z_1,z_2 \in S} d_H(z_1,z_2)$. We then define OPT-SUM($\mathbf{F},s$) as the maximum value of SUMPD(S) over all multisets S with s satisfying assignments (including multiplicities), and
OPT-MIN($\mathbf{F},s$) $= \max_{S \subseteq \Omega_{\mathbf{F}}, |S|=s}$ MINPD(S), i.e., the maximum such distance over all sets of s satisfying assignments. Further, we define OPT-SUM$_{\neq}$($\mathbf{F},s$) as the maximum value of SUMPD(S) over all *sets* S with s distinct satisfying assignments.

3.1 Computing diameter exactly and approximately

Computing diameter exactly. We first study the exponential complexity of computing DIAM($\mathbf{F}$). Specifically, we prove the following theorem.

▶ **Theorem 6** (Exact Diameter). *Let $\mathbf{F}$ be a k-CNF formula on n variables. There exists a deterministic algorithm that uses $O^*(2^n)$ time and $O^*(2^n)$ space, and outputs a pair of satisfying assignments $z_1, z_2 \in \Omega_{\mathbf{F}}$ with $d_H(z_1,z_2) =$ DIAM($\mathbf{F}$).*

Prior to our work, the best exact algorithm known was by Angelsmark and Thapper [5]. Their algorithm runs in time $O((2a_k)^n)$ and space poly(n), where $O(a_k^n)$ is the running time for solving the k-SAT problem. Our result significantly improves the running time of their algorithm (but uses substantially more space than their algorithm).

Our technique is also different from other techniques in the literature. Namely, this algorithm does not depend on any SAT algorithm. Our main observation is that DIAM($\mathbf{F}$) can be reduced to computing the *convolution* of the Boolean function represented by $\mathbf{F}$ with itself. We then use that such a convolution can be computed within the above stated time and space bounds using the Fast Fourier Transform.

Our technique for exact diameter is fairly general and does not depend on the fact that the solution space corresponds to a k-CNF formula. For any Boolean function $f : \{0,1\}^n \to \{0,1\}$ such that for a given $x \in \{0,1\}^n$, there is a polynomial time oracle to compute $f(x)$, our algorithm can be used to exactly compute the diameter of f with the above performance guarantees.

Approximating the diameter. Next, we give algorithms for approximating $\textsc{Diam}(\mathbf{F})$[4]. As a warm-up, here is a simple way to approximate $\textsc{Diam}(\mathbf{F})$. We can start by using the best known algorithm to find a single satisfying assignment for $\mathbf{F}$. Suppose that assignment is α. We can then (in polynomial time) change $\mathbf{F}$ to $\mathbf{F}'_\alpha$ by negating some of the variables such that 1^n becomes the satisfying assignment of $\mathbf{F}'_\alpha$. One can then use the best known algorithm for the Min-Ones problem to find a satisfying assignment for $\mathbf{F}'_\alpha$, which finds a satisfying assignment with minimum 1s in it, say β. It is easy to see that the Hamming distance between α, β gives a 0.5-approximation to the diameter of $\mathbf{F}$. By using the best known algorithms for k-SAT by Paturi, Pudlák, Saks, and Zane [41] and for Min-Ones by Fomin, Gaspers, Lokshtanov and Saurabh [20], it is easy to see that we can obtain (α, β) in time $O^*((2-\frac{1}{k})^n) = O^*\left(2^{(1-\frac{1}{(2\ln 2)\cdot k})n}\right)$.[5]

Here, we obtain better running time for $\textsc{Diam}(\mathbf{F})$ for $k \geq 3$ with a small loss in the approximation factor. From here on, we assume that $k \geq 3$ unless stated otherwise.

▶ **Theorem 7** (PPZ approximating $\textsc{Diam}(\mathbf{F})$)**.** *Let $\mathbf{F}$ be a k-CNF formula on n variables. There exists a randomized algorithm running in time $O^*\left(2^{(1-1/k)n}\right)$ that takes $\mathbf{F}$ as input and if $\mathbf{F}$ is satisfiable, outputs $z_1^*, z_2^* \in \Omega_{\mathbf{F}}$ with $d_H(z_1^*, z_2^*) \geq \frac{1}{2} \cdot \left(1 - \frac{1}{k}\right) \textsc{Diam}(\mathbf{F})$ with probability $1 - o(1)$.*

The running time of the algorithm here is exactly the same as the running time of the algorithm achieved in [42], which solves the k-SAT problem. Our result demonstrates that the diameter can be approximated in the same time used to compute a single satisfying assignment. In fact, the way we achieve this running time is by repeatedly invoking the PPZ algorithm. At the heart of the analysis of the PPZ algorithm lies the Satisfiability Coding Lemma from [42]. Informally speaking, the Satisfiability Coding Lemma says that if the solutions of a k-CNF instance are *well-separated* then they have a small description. In our proof, we generalise this lemma. We discuss our proof idea in detail in Section 5.

Next, we show how to approximate the diameter within the running time guarantees of Schöning's algorithm for k-SAT. Specifically, we prove the following theorem.

▶ **Theorem 8** (Schöning approximating $\textsc{Diam}(\mathbf{F})$.)**.** *Let $\mathbf{F}$ be a k-CNF formula on n variables. There exists a randomized algorithm running in time $O^*\left(2^{(1-\frac{1}{k\ln 2})n}\right)$ that takes $\mathbf{F}$ as input and if $\mathbf{F}$ is satisfiable, outputs $z_1^*, z_2^* \in \Omega_{\mathbf{F}}$ with $d_H(z_1^*, z_2^*) \geq \frac{1}{2}\left(1 - \frac{4(k-1)}{(k-2)^2}\right) \cdot \textsc{Diam}(\mathbf{F})$ with probability $1 - o(1)$.*

In fact, Theorem 8 is one instance of a smooth tradeoff between the approximation factor and the running time. Notice that the running time obtained here is better than the running time obtained using Theorem 7, which in turn is faster than the naive algorithm that uses Min-Ones. We incur some loss in the approximation factors to obtain these speedups. As stated, the result gives non-trivial approximation factors when $k \geq 7$. We refer the reader to the full version of this paper [7] for more details.

3.2 Computing dispersion exactly and approximately

We extend all the algorithms from Section 3.1 and obtain bounds for the dispersion problem.

[4] All approximation algorithms we present here use $\text{poly}(n)$ space.

[5] Note that, $O^*((2-\frac{1}{k})^n) = O^*(2(1-\frac{1}{2k}))^n \sim O^*(2^n \cdot e^{-\frac{n}{2k}}) = 2^{n(1-\frac{1}{(2\ln 2)\cdot k})}$.

Exact algorithms for dispersion. We start with the problem of exactly computing Opt-sum$(\mathbf{F}, s)$, Opt-min$(\mathbf{F}, s)$ and Opt-sum$_{\neq}(\mathbf{F}, s)$. The obvious algorithm for computing all these quantities would be to do a brute force search over all $z_1, z_2, \ldots, z_s \in \{0,1\}^n$, which would require $O^*(2^{sn})$ time. We observe that we can extend the Fourier analytical approach we used in Theorem 6 to do this in $O^*(2^{(s-1)n})$ time and $O^*(2^n)$ space. We also provide an alternate algorithm for dispersion based on clique-finding that runs in time $O(s^2 \cdot |\Omega_{\mathbf{F}}|^{\omega\lceil s/3\rceil})$ and uses space $O(|\Omega_{\mathbf{F}}|^{2\lceil s/3\rceil})$, where $\omega \leq 2.38$ denotes the matrix multiplication exponent [53]. As such, it is faster than the Fourier analysis-based algorithm for any $s \geq 6$, and can be much faster when the size of the solution set is less than 2^n. We formally state these results below.

▶ **Theorem 9** (Exact Dispersion using FFT)**.** *Let $f : \{0,1\}^n \to \{0,1\}$ be a function computable by a black box and let s be a given parameter. Then, there exist deterministic algorithms $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$ that make 2^n oracle calls to f and in addition to that, use $O^*(2^{(s-1)n})$ time and $O^*(2^n)$ space provide the following guarantees.*

1. *The output of $\mathcal{A}_1$ is a multiset $\{z_1, z_2, \ldots, z_s\} \subseteq f^{-1}(1)$ such that sumPD$(z_1, z_2, \ldots, z_s) =$ Opt-sum(f, s).*
2. *The output of $\mathcal{A}_2$ is a set $\{z_1, z_2, \ldots, z_s\} \subseteq f^{-1}(1)$ such that minPD$(z_1, z_2, \ldots, z_s) =$ Opt-min(f, s).*
3. *If $|f^{-1}(1)| \geq s$, the output of $\mathcal{A}_3$ is a set $\{z_1, z_2, \ldots, z_s\} \subseteq f^{-1}(1)$ such that sumPD$(z_1, z_2, \ldots, z_s) =$ Opt-sum$_{\neq}(f, s)$.*

▶ **Theorem 10** (Exact Dispersion using Clique Finding)**.** *There exist deterministic algorithms $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$ that given as input a non-empty set $X \subseteq \{0,1\}^n$ of size M and parameter s, runs in $O(\text{poly}(n, s) \cdot M^{\omega\lceil s/3\rceil})$ time, uses $O(M^{2\lceil s/3\rceil})$ space, and have the following behaviour.*

1. *The output of $\mathcal{A}_1$ is $z_1, z_2, \ldots, z_s \in X$ such that sumPD$(z_1, z_2, \ldots, z_s) =$ Opt-sum(X, s).*
2. *The output of $\mathcal{A}_2$ is $z_1, z_2, \ldots, z_s \in X$ such that minPD$(z_1, z_2, \ldots, z_s) =$ Opt-min(X, s).*
3. *And, as long as $|S| \geq s$, the output of $\mathcal{A}_3$ is a set $\{z_1, z_2, \ldots, z_s\} \in X$ such that sumPD$(z_1, z_2, \ldots, z_s) =$ Opt-sum$_{\neq}(X, s)$.*

Approximating dispersion. We now turn to approximation algorithms for dispersion. Our goal is to come up with approximation algorithms for all the versions of the dispersion problem as in the case of approximation algorithms for computing the diameter.

We saw that Min-Ones can be used to give a 0.5 approximation to Diam$(\mathbf{F})$. However, it is not clear how we can use it to approximate the dispersion. We elaborate in Section 5.

Approximating Opt-sum$(\mathbf{F}, s)$. We show that PPZ as well as Schöning's algorithms can be modified to compute Opt-sum$(\mathbf{F}, s)$. Formally,

▶ **Theorem 11** (PPZ approximating Opt-sum$(\mathbf{F}, s)$)**.** *Let $\mathbf{F}$ be a k-CNF formula on n variables. There exists a randomized algorithm running in time $O^*\left(s^4 \cdot 2^{(1-1/k)n}\right)$ that takes $\mathbf{F}$ and an integer $s \geq 1$ as input and if $\mathbf{F}$ is satisfiable, with probability at least $1 - o(1)$, outputs a multiset $S \subseteq \Omega_{\mathbf{F}}$ of size s such that*

$$\text{sumPD}(S) \geq \left(1 - \frac{4}{k-3}\right)\left(1 - \frac{2}{s+2}\right) \cdot \text{Opt-sum}(\mathbf{F}, s) \ .$$

▶ Remark 12. When $k \leq 6$, this algorithm achieves a better approximation ratios for smaller values of s than stated above. Note that as k and s become large, the approximation factor tends to 1. For more details, we refer the reader to the full version of this paper [7, Section 3].

We note that we can design algorithms for $\textsc{Opt-sum}_{\neq}(\mathbf{F}, s)$, with exactly the same approximation factors as for $\textsc{Opt-sum}(\mathbf{F}, s)$ for certain parameter regimes of s (As earlier, we refer the reader to the full version of this paper for more details).

Approximating Opt-min$(\mathbf{F}, s)$. Next, we show that our techniques can be used to approximate $\textsc{Opt-min}$ as well. Formally,

▶ **Theorem 13** (PPZ approximating $\textsc{Opt-min}(\mathbf{F}, s)$)**.** *Let $\mathbf{F}$ be a k-CNF formula on n variables. There exists a randomized algorithm running in time $O^*\left(s^3 \cdot 2^{(1-1/k)n}\right)$ that takes $\mathbf{F}$ and an integer $s \geq 1$ as input and if $\mathbf{F}$ is satisfiable and $|\Omega_{\mathbf{F}}| \geq s$, with probability at least $1 - o(1)$, outputs a set S of size s such that $\textsc{min}PD(S) \geq \frac{1}{2}\left(1 - \frac{1}{kH^{-1}(1-1/k)}\right) \cdot \textsc{Opt-min}(\mathbf{F}, s)$.*[6]

Note that, in the above statement, the approximation factor is non-trivial (> 0) only for $k \geq 5$. We note that we can also obtain Schöning-type running time bounds for dispersion for $k \geq 2$. We achieve this by extending Theorem 8.

Approximating Opt-min$(\mathbf{F}, s)$ for heavy-weight solutions. We now consider a heavy-weight variant of $\textsc{Opt-min}(\mathbf{F}, s)$. Formally, for a k-CNF formula $\mathbf{F}$, we let $\Omega_{\mathbf{F},\geq W}$ denote the set of satisfying assignments to $\mathbf{F}$ with Hamming weight at least W. We then define

$$\textsc{Opt-min}(\mathbf{F}, s, \geq W) = \max_{\substack{S \subseteq \Omega_{\mathbf{F},\geq W} \\ |S|=s}} \textsc{min}\mathrm{PD}(S), \textsc{Opt-min}(\mathbf{F}, s, \leq W) = \max_{\substack{S \subseteq \Omega_{\mathbf{F},\leq W} \\ |S|=s}} \textsc{min}\mathrm{PD}(S)\ .$$

and let $\textsc{min}\mathrm{W}(S)$ denote the minimum Hamming weight of assignments in S. We show that the approach developed for approximating $\textsc{Opt-min}$ via Schöning's algorithm can also be used to return dispersed satisfying assignments of heavy weight.

▶ **Theorem 14** (Schöning for weighted dispersion)**.** *Let $\mathbf{F}$ be a k-CNF formula on n variables, $W \in [n]$ and $s \in \mathbb{N}$. Let $\delta = \frac{4(k-1)}{(k-2)^2}$. There exist algorithms that take $\mathbf{F}, s, W$ as input and output with probability $1 - o(1)$ in time $O^*\left(s^3 \cdot 2^{n(1-\frac{1}{k \ln 2})}\right)$:*

1. *$S^* \subseteq \Omega_{\mathbf{F},\geq(1-\delta)W}$ of size s such that $\textsc{min}PD(S^*) \geq \frac{1}{2}(1-\delta)\,\textsc{Opt-min}(\mathbf{F}, s, \geq W)$ if $\mathbf{F}$ is satisfiable and $|\Omega_{\mathbf{F},\geq W}| \geq s$.*
2. *$S^* \subseteq \Omega_{\mathbf{F},\leq(1+\delta)W}$ of size s such that $\textsc{min}PD(S^*) \geq \frac{1}{2}(1-\delta)\,\textsc{Opt-min}(\mathbf{F}, s, \leq W)$ if $\mathbf{F}$ is satisfiable and $|\Omega_{\mathbf{F},\leq W}| \geq s$,*

▶ Remark 15. We note that when $W = 0$, this just reduces to an algorithm for approximating $\textsc{Opt-min}(\mathbf{F}, s)$. The approximation factors in Theorem 14 are non-trivial only for $k \geq 7$. However, just like the case of Theorem 8, Theorem 14 can be generalized, obtaining running time bounds for any k and for a larger range of approximation factors. Further, we can prove that an analogous result exists for the sum of distances dispersion measure.
We refer the reader to [7, Section 4] for the complete theorem statements and proofs of our Schöning type results.

4 Generalizations and applications

1. Isometric Reductions. Dispersion has also been studied when the space is induced by solutions to some NP-complete optimization problem [10, 9]. To address this optimization aspect, we first generalize our techniques to give dispersed solutions of high (or low) Hamming

[6] The function $H^{-1}(\cdot)$ denotes the inverse of the binary entropy function $H(x) = -x\log(x) - (1-x)\log(1-x)$ restricted to the domain $[0, 1/2]$. The domain of H^{-1} is $[0, 1]$ and its range is $[0, 1/2]$.

weight[7]. Namely, given $W \in [n]$, all of our solutions will have Hamming weight at least (or at most) approximately W, and their dispersion will be close to that of an optimally dispersed set wherein all solutions have weight at least (or at most) W. We then formalize a set of reductions, that preserve the size of the solution set and the distances between solutions. We call such reductions *isometric*. As a result, we can approximate dispersion for problems such as Maximum Independent Set, Minimum Vertex Cover and Minimum Hitting Set.

2. Using the monotone local search framework for diverse solutions. Our second application allows us to compute diverse solutions to optimization problems that perhaps do not allow isometric reductions to SAT. In this case, we show how to use the *monotone local search* framework by Fomin, Gaspers, Lokshtanov and Saurabh [20]. This allows us to extend our results to a variety of problems, including Feedback Vertex Set, Multicut on Trees, and Minimum d-Hitting Set (see Table 1 for a sample of the results that can be obtained using this technique[8]).

For all of these problems, any existing exact methods for finding a set of optimal, maximally diverse solutions has a runtime with at least an exponential dependence on the number of solutions s [10, 9]. Our methods show that by relaxing to bi-approximations, this dependence on s can be made polynomial.

Table 1 The second column contains the time taken to obtain one exact solution using methods in [20]. The third column contains the time taken to obtain 3/2-approx. optimal, 1/4-approx. maximally diverse solutions (except for Maximum Independent Set, where we obtain $(1/2, 1/4)$-bi-approx.)

Optimization Problem	One optimal solution [20]	Multiple approximately optimal, approximately dispersed solutions
Vertex cover	1.5^n	$s^3 \cdot 1.5486^n$
Maximum independent Set	1.5^n	$s^3 \cdot 1.5486^n$
Feedback Vertex Set	1.7217^n	$s^3 \cdot 1.6420^n$
Subset Feedback Vertex Set	1.7500^n	$s^3 \cdot 1.6598^n$
Feedback Vertex Set in Tournaments	1.3820^n	$s^3 \cdot 1.5162^n$
Group Feedback Vertex Set	1.7500^n	$s^3 \cdot 1.6598^n$
Vertex (r, ℓ)-Partization $(r, \ell \leq 2)$	1.6984^n	$s^3 \cdot 1.6289^n$
Interval Vertex Deletion	1.8750^n	$s^3 \cdot 1.7789^n$
Proper Interval Vertex Deletion	1.8334^n	$s^3 \cdot 1.7284^n$
Block Graph Vertex Deletion	1.7500^n	$s^3 \cdot 1.6598^n$
Cluster Vertex Deletion	1.4765^n	$s^3 \cdot 1.5415^n$
Thread Graph Vertex Deletion	1.8750^n	$s^3 \cdot 1.7789^n$
Multicut on Trees	1.3565^n	$s^3 \cdot 1.51^n$
3-Hitting Set	1.5182^n	$s^3 \cdot 1.5544^n$
4-Hitting Set	1.6750^n	$s^3 \cdot 1.6167^n$
Weighted Feedback Vertex Set	1.7237^n	$s^3 \cdot 1.6432^n$
Weighted 3-Hitting Set	1.5388^n	$s^3 \cdot 1.5612^n$

[7] In a recent work, Gurumukhani, Paturi, Pudlák, Saks, and Talebanfard [26] consider the problem of enumerating satisfying assignments with Hamming weight at least W for a given k-CNF formula (assuming that satisfying assignments of smaller weight do not exist). They show that this problem has interesting connections to circuit lower bounds.

[8] The table provides the running time guarantees to obtain 3/2-approx. optimal, 1/4-approx. maximally diverse solutions, by plugging in $\delta = 1/2$ into the run-time bounds in our generalized theorem [7, Theorem 36]

3. On faster SAT algorithms. Another compelling reason to study diversity of the solution space of a k-CNF formula is that the existence of far apart solutions might be used to study the computational complexity of k-SAT and its variants. Indeed, the geometry of the solution space has been studied extensively, both to obtain faster SAT solvers (parameterised by the number of solutions, such as in Hirsch [32] and Kane and Watanabe [37]) and in the random SAT setting, e.g., the diameter by Feige, Flaxman and Vilenchik [19] and the giant connected component by Chen, Mani, Moitra [17]).

Consider a formula $\mathbf{F}$ with $|\Omega_{\mathbf{F}}| = 2^{\delta n}$ for some $\delta > 0$. For such a formula, it is known that PPZ scales optimally, i.e., it finds one solution in time $2^{(1-1/k)(1-\delta)n}$ [14]. Cardinal, Nummenpalo and Welzl [15] proved a weaker result for Schöning, but nevertheless, both PPZ and Schöning run faster if the solution space is large. In fact, the same is true for PPSZ [47].

Taking this idea a step further, we investigate the runtime of PPZ and Schöning's algorithms when $\Omega_{\mathbf{F}}$ contains many well-dispersed solutions. For example, if $\Omega_{\mathbf{F}}$ contains a Hamming code that achieves the Gilbert Varshmov bound, we can show an exponential improvement in the runtime of Schöning's algorithm. Similarly, using the geometric sampling property of PPZ in Lemma 1, we obtain an improved runtime in this setting. In this sense, *if having more (solutions) is better [47], then our results formalize the intuition that more dispersed solutions are even better*. To be precise, for every $r \in [n]$, we define $N_r := \max\{|S| : S \subseteq \Omega_{\mathbf{F}}, \textsc{minPD}(S) \geq r\}$. Note that from the definition of N_r, for every $r \in [n]$, there exists a set $S_r \subseteq \Omega_{\mathbf{F}}$ of size N_r, such that the balls of radius $\lfloor \frac{r}{2} \rfloor$ around each $z^* \in S_r$ are disjoint. We also note that $N_0 \geq N_1 \geq \cdots \geq N_n$.

▶ **Theorem 16.** *Let $\mathbf{F}$ be a k-CNF formula. If $\mathbf{F}$ is satisfiable, Schöning's algorithm succeeds in finding a satisfying assignment within $O^*\left(\frac{2^n(1-1/k)^n}{N_{\lfloor 2n/k \rfloor}}\right)$ iterations.*

▶ **Theorem 17.** *Let $\mathbf{F}$ be a k-CNF formula. If $\mathbf{F}$ is satisfiable, the PPZ algorithm succeeds in finding a satisfiable assignment to $\mathbf{F}$ within $O^*\left(\frac{2^{n-n/k}}{N_{\lfloor 2n/k \rfloor}}\right)$ iterations.*

4. Relation to coding theory. We mention a connection that might be of independent interest. The dispersion problem can be restated in the language of coding theory, namely, we are looking for codewords that also satisfy a given k-CNF formula. If $\mathbf{F}(x) = 1$ for all $x \in \{0,1\}^n$, then it is known that a uniformly random code achieves the Gilbert-Varshamov bound [45]. When $\mathbf{F}$ is not trivial, the algorithms presented in this work provide such a code. Moreover, our result says that the code can be found in time proportional to the running times of PPZ and Schöning (when the size of the code is small). Additionally, in practice, one also wants codes that have succinct representations, e.g. linear codes [27, 25]. While our codes do not exhibit this property, it would indeed be interesting to extend our algorithms in this direction.

5. CSPs. Finally, since Schöning's algorithm for finding one solution generalizes to CSPs, we also give algorithms obtaining diverse solutions to CSPs.

5 Technical Overview: Proof sketches for Lemma 1 and Lemma 2

In this section we outlines the main techniques behind Lemma 1 and Lemma 2, that show that PPZ and Schöning algorithms can be employed as approximate farthest point oracles. Because of this approximation, slightly more work needs to be done in order to bound the overall approximation factors for dispersion. For the sum dispersion problem, we also adapt Cevallos, Eisenbrand, and Zenklusen's local search algorithm [16] for our setting to get an improved approximation factor.

Lemma 1: PPZ samples geometrically

The PPZ algorithm consists of repeating the following procedure $O^*(2^{(1-1/k)n})$ times: sample an assignment $y \in \{0,1\}^n$ and a permutation $\pi \in S_n$ uniformly and independently at random. Then call a deterministic subroutine $\mathsf{PPZ\text{-}Modify}(\mathbf{F}, y, \pi)$ that runs in $n^{O(1)}$ time and outputs another assignment u. The algorithm stops once $u \in \Omega_{\mathbf{F}}$.

The analysis is based on bounding the probability that, for a randomly chosen y and π, $\mathsf{PPZ\text{-}Modify}(\mathbf{F}, y, \pi)$ leads to some satisfying assignment $z \in \Omega_{\mathbf{F}}$. For any $z \in \Omega_{\mathbf{F}}$, let $\tau(\mathbf{F}, z)$ denote the probability that an iteration outputs z and for any set $A \subseteq \Omega_{\mathbf{F}}$, let $\tau(\mathbf{F}, A) = \sum_{z' \in A} \tau(\mathbf{F}, z')$ denote the probability that an iteration outputs a satisfying assignment in A.

The lower bound that PPZ gives on $\tau(\mathbf{F}, z)$ uses the the *local* geometry of $\Omega_{\mathbf{F}}$ around z in the following sense: we say that z is j-*isolated* if, out of the n neighboring assignments to z in the Boolean hypercube, at least j of them are not satisfying. The key observation in the analysis of the PPZ algorithm, called the *Satisfiability Coding Lemma* [42] states that for every j-isolated satisfying assignment z, it holds that $\tau(\mathbf{F}, z) \geq 2^{-n+j/k}$. Intuitively, the more isolated a solution z is, the more choices of y and π would lead to it through $\mathsf{PPZ\text{-}Modify}(\mathbf{F}, y, \pi)$.

Our renewed analysis of PPZ shows that, for any fixed assignment $z_0 \in \{0,1\}^n$, $\mathsf{PPZ\text{-}Modify}(\mathbf{F}, y, \pi)$ is also likely to output satisfying assignments that are far away from it. We state Lemma 1 formally in [7, Lemma 18] that shows that with probability at least $\frac{1}{2n} \cdot 2^{-n+n/k}$, each iteration of the PPZ algorithm outputs a satisfying assignment z^*, such that

$$d_H(z_0, z^*) \geq \left(1 - \frac{1}{k}\right) \cdot \max_{z' \in \Omega_{\mathbf{F}}} d_H(z_0, z') \ .$$

Thus, we get that PPZ is also an approximate farthest point oracle. More interestingly, the run of PPZ does not depend on z_0, and therefore we say that PPZ samples geometrically. We note that the original analysis does not take into account distances between solutions, i.e., the probability of finding a solution only depends on the number of its *immediate* neighbors that are non-solutions. This in itself is a local feature that does not capture global properties like the diameter/dispersion of the solution space. Indeed, our analysis differs from the original PPZ analysis in precisely the fact that it exploits this global information (which is needed for diameter/diversity, but not needed if we just want to find one solution).

In order to exploit global geometric properties of the solution space, we view $\Omega_{\mathbf{F}}$ as a subgraph $G_{\mathbf{F}}$ of the n-dimensional Hypercube graph. We then divide the vertices in $G_{\mathbf{F}}$ into n layers, where layer V_j consists of all the vertices at distance j from z_0 (in $G_{\mathbf{F}}$). We also define $U_j = \bigcup_{j' \geq j} V_{j'}$. Now, we want to show that assignments in higher layers will be reached by $\mathsf{PPZ\text{-}Modify}(\mathbf{F}, y, \pi)$ with good probability. We do this by proving that for large enough j, either $|U_j|$ is large or the number of cut edges between U_j and $\Omega_{\mathbf{F}} \setminus U_j$ is small in $G_{\mathbf{F}}$.

We then use the original Satisfiability Coding Lemma and the fact that an assignment is j-isolated if and only if its degree in $G_{\mathbf{F}}$ is $n - j$, to show that, for any subset A of the vertices in $G_{\mathbf{F}}$, it holds that

$$\tau(\mathbf{F}, A) \geq 2^{-n(1-1/k)} |A| 2^{-\left(\frac{2|E(A)|}{k|A|} + \frac{|S|}{k|A|}\right)} \ ,$$

where $E(A)$ denotes the edges in $G_{\mathbf{F}}$ between vertices in A and S denotes the edges in $G_{\mathbf{F}}$ between A and $\Omega_{\mathbf{F}} \setminus A$. We then use the edge isoperimetric lemma for subgraphs of the

hypercube which upper bounds the number of edges in the subgraph by a function of the number of vertices in the subgraph. To complete the proof of Lemma 1, we lower bound the probability $\tau(\mathbf{F}, A)$, where A are the assignments in $\Omega_{\mathbf{F}}$ that are far away from z_0.

We also show that the above analysis can be extended to prove that for any subset $S \subseteq \{0,1\}^n$, with probability at least $\frac{1}{2n} \cdot 2^{-n+n/k}$, each iteration of the PPZ algorithm outputs a satisfying assignment z^*, such that

$$\text{SUM-}d_H(S, z^*) \geq \left(1 - \frac{2}{k+1}\right) \cdot \max_{z' \in \Omega_{\mathbf{F}}} \text{SUM-}d_H(S, z') \ .$$

This directly implies the existence of a $\left(1 - \frac{2}{k+1}\right)$-approximate farthest point oracle that runs in the same time as the PPZ algorithm [7, Section 3]. However, we were not able to show a similar lower bound with respect to the MIN-d_H distance from S. Instead, we can use Lemma 1 to show that for every satisfying assignment $z \in \Omega_{\mathbf{F}}$, each iteration of the PPZ algorithm outputs a satisfying assignment within Hamming distance $\frac{n}{k}$ from z (invoke Lemma 1 on the antipode of z). We can also assume that we have a lower bound on $\max_{z' \in \Omega_{\mathbf{F}}} \text{MIN-}d_H(S, z')$ on the order of $n/\Theta(1)$ (just exhaustively search all the balls around assignments in S until you hit PPZ running time). Thus, we get an approximate farthest point oracle running in the same time as the PPZ algorithm for the min-dispersion problem as well.

Lemma 2: Modified Schöning's algorithm is a farthest point oracle

Our second approach for designing farthest point oracles uses Schöning's algorithm [51]. At its core, Schöning's algorithm is a local search algorithm that does a random walk from some starting assignment z_0. The main subroutine takes as input z_0 and, as long as there is a clause that is unsatisfied, picks one of its k literals at random an flips its value. Schöning showed that, if there exists a satisfying assignment within Hamming distance t from z_0, then within $3t$ steps, the above random walk outputs a satisfying assignment with probability at least $1/(k-1)^t$. By picking the starting point z_0 uniformly at random from $\{0,1\}^n$ and letting the random walk go for $3n$ steps, one can then show that the subroutine suceeds with probability at least $\left(\frac{1}{2} \cdot \left(1 + \frac{1}{k-1}\right)\right)^n$.

We modify Schöning's algorithm by picking the starting point z_0 and then setting the length of the random walk more carefully. Suppose we are promised that there exists a satisfying assignment z^* that is distance r (in max-sum or max-min) from some set S of assignments. We then restrict our starting points to be sampled such that they are also guaranteed to be approximately at distance r from S. From there, we perform a random walk of small length such that any satisfying assignment we find is also guaranteed to be far away from S. The probability that we succeed depends on bounding the set of good starting points: those that are close to the promised z^* (not just far from S), since these are the ones most likely to find a satisfying assignment within the length of the random walk. This is the most technically involved step of our analysis. We thus get a farthest point oracle for diameter and all versions of dispersion. Moreover, the Schöning strategy can also find heavy-weight assignments. This is done by artificially adding 0^n as part of the set S (thus, an assignment that is far from S in Hamming distance will also have a large weight).

References

1 Amir Abboud, Vincent Cohen-Addad, Euiwoong Lee, and Pasin Manurangsi. Improved approximation algorithms and lower bounds for search-diversification problems. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPIcs*, pages 7:1–7:18. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ICALP.2022.7.

2 Sabih Agbaria, Dan Carmi, Orly Cohen, Dmitry Korchemny, Michael Lifshits, and Alexander Nadel. Sat-based semiformal verification of hardware. In Roderick Bloem and Natasha Sharygina, editors, *Proceedings of 10th International Conference on Formal Methods in Computer-Aided Design, FMCAD 2010, Lugano, Switzerland, October 20-23*, pages 25–32. IEEE, IEEE, 2010. URL: https://ieeexplore.ieee.org/document/5770929/.

3 Josh Alman and Ryan Williams. Probabilistic polynomials and hamming nearest neighbors. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 136–150. IEEE, IEEE Computer Society, 2015. doi:10.1109/FOCS.2015.18.

4 Ola Angelsmark and Johan Thapper. Algorithms for the maximum Hamming distance problem. In Boi Faltings, Adrian Petcu, François Fages, and Francesca Rossi, editors, *International Workshop on Constraint Solving and Constraint Logic Programming*, volume 3419 of *Lecture Notes in Computer Science*, pages 128–141. Springer, 2004. doi:10.1007/11402763_10.

5 Ola Angelsmark and Johan Thapper. A microstructure based approach to constraint satisfaction optimisation problems. In Ingrid Russell and Zdravko Markov, editors, *Proceedings of the Eighteenth International Florida Artificial Intelligence Research Society Conference, Clearwater Beach, Florida, USA*, pages 155–160. AAAI Press, 2005. URL: http://www.aaai.org/Library/FLAIRS/2005/flairs05-026.php.

6 Andrea Arcuri and Lionel C. Briand. Formal analysis of the probability of interaction fault detection using random testing. *IEEE Trans. Software Eng.*, 38(5):1088–1099, 2012. doi:10.1109/TSE.2011.85.

7 Per Austrin, Ioana O Bercea, Mayank Goswami, Nutan Limaye, and Adarsh Srinivasan. Algorithms for the diverse-k-sat problem: the geometry of satisfying assignments. *arXiv preprint arXiv:2408.03465*, 2024. doi:10.48550/arXiv.2408.03465.

8 Nikhil Bansal, Kamal Jain, Anna Kazeykina, and Joseph Naor. Approximation algorithms for diversified search ranking. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6-10, 2010, Proceedings, Part II*, volume 6199 of *Lecture Notes in Computer Science*, pages 273–284. Springer, Springer, 2010. doi:10.1007/978-3-642-14162-1_23.

9 Julien Baste, Michael R. Fellows, Lars Jaffke, Tomás Masarík, Mateus de Oliveira Oliveira, Geevarghese Philip, and Frances A. Rosamond. Diversity of solutions: An exploration through the lens of fixed-parameter tractability theory. *Artif. Intell.*, 303:103644, 2022. doi:10.1016/j.artint.2021.103644.

10 Julien Baste, Lars Jaffke, Tomás Masarík, Geevarghese Philip, and Günter Rote. FPT algorithms for diverse collections of hitting sets. *Algorithms*, 12(12):254, 2019. doi:10.3390/a12120254.

11 Sven Baumer and Rainer Schuler. Improving a probabilistic 3-SAT algorithm by dynamic search and independent clause pairs. In Enrico Giunchiglia and Armando Tacchella, editors, *Theory and Applications of Satisfiability Testing, 6th International Conference, SAT 2003. Santa Margherita Ligure, Italy, May 5-8, 2003 Selected Revised Papers*, volume 2919 of *Lecture Notes in Computer Science*, pages 150–161. Springer, Springer, 2003. doi:10.1007/978-3-540-24605-3_12.

12 Allan Borodin, Hyun Chul Lee, and Yuli Ye. Max-sum diversification, monotone submodular functions and dynamic updates. In Michael Benedikt, Markus Krötzsch, and Maurizio Lenzerini, editors, *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2012, Scottsdale, AZ, USA, May 20-24, 2012*, pages 155–166. ACM, 2012. doi:10.1145/2213556.2213580.

13 Chris Calabro. *The exponential complexity of satisfiability problems*. PhD thesis, University of California, San Diego, USA, 2009. URL: http://www.escholarship.org/uc/item/0pk5w64k.

14 Chris Calabro, Russell Impagliazzo, Valentine Kabanets, and Ramamohan Paturi. The complexity of unique k-SAT: An isolation lemma for k-CNFs. *J. Comput. Syst. Sci.*, 74(3):386–393, 2008. doi:10.1016/j.jcss.2007.06.015.

15 Jean Cardinal, Jerri Nummenpalo, and Emo Welzl. Solving and sampling with many solutions: Satisfiability and other hard problems. In Daniel Lokshtanov and Naomi Nishimura, editors, *12th International Symposium on Parameterized and Exact Computation, IPEC 2017, September 6-8, 2017, Vienna, Austria*, volume 89 of *LIPIcs*, pages 11:1–11:12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.IPEC.2017.11.

16 Alfonso Cevallos, Friedrich Eisenbrand, and Rico Zenklusen. An improved analysis of local search for max-sum diversification. *Math. Oper. Res.*, 44(4):1494–1509, 2019. doi:10.1287/moor.2018.0982.

17 Zongchen Chen and Nitya Mani. From algorithms to connectivity and back: Finding a giant component in random k-sat. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 3437–3470. SIAM, SIAM, 2023. doi:10.1137/1.9781611977554.ch132.

18 Pierluigi Crescenzi and Gianluca Rossi. On the Hamming distance of constraint satisfaction problems. *Theor. Comput. Sci.*, 288(1):85–100, 2002. doi:10.1016/S0304-3975(01)00146-3.

19 Uriel Feige, Abraham D. Flaxman, and Dan Vilenchik. On the diameter of the set of satisfying assignments in random satisfiable k-cnf formulas. *SIAM J. Discret. Math.*, 25(2):736–749, 2011. doi:10.1137/090749323.

20 Fedor V. Fomin, Serge Gaspers, Daniel Lokshtanov, and Saket Saurabh. Exact algorithms via monotone local search. *J. ACM*, 66(2):8:1–8:23, March 2019. doi:10.1145/3284176.

21 Fedor V. Fomin and Petteri Kaski. Exact exponential algorithms. *Commun. ACM*, 56(3):80–88, 2013. doi:10.1145/2428556.2428575.

22 Jie Gao, Mayank Goswami, Karthik C. S., Meng-Tsung Tsai, Shih-Yu Tsai, and Hao-Tsung Yang. Obtaining approximately optimal and diverse solutions via dispersion. In Armando Castañeda and Francisco Rodríguez-Henríquez, editors, *LATIN 2022: Theoretical Informatics - 15th Latin American Symposium, Guanajuato, Mexico, November 7-11, 2022, Proceedings*, volume 13568 of *Lecture Notes in Computer Science*, pages 222–239. Springer, Springer, 2022. doi:10.1007/978-3-031-20624-5_14.

23 Carla P. Gomes, Ashish Sabharwal, and Bart Selman. Near-uniform sampling of combinatorial spaces using XOR constraints. In Bernhard Schölkopf, John C. Platt, and Thomas Hofmann, editors, *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006*, volume 19, pages 481–488. MIT Press, 2006. URL: https://proceedings.neurips.cc/paper/2006/hash/4110a1994471c595f7583ef1b74ba4cb-Abstract.html.

24 Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theor. Comput. Sci.*, 38:293–306, 1985. doi:10.1016/0304-3975(85)90224-5.

25 Elena Grigorescu, Tali Kaufman, and Madhu Sudan. Succinct representation of codes with applications to testing. *SIAM J. Discret. Math.*, 26(4):1618–1634, 2012. doi:10.1137/100818364.

26 Mohit Gurumukhani, Ramamohan Paturi, Pavel Pudlák, Michael E. Saks, and Navid Talebanfard. Local enumeration and majority lower bounds. *CoRR*, abs/2403.09134, 2024. doi:10.48550/arXiv.2403.09134.

27 Venkatesan Guruswami, Johan Håstad, and Swastik Kopparty. On the list-decodability of random linear codes. In Leonard J. Schulman, editor, *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 409–416. ACM, 2010. doi:10.1145/1806689.1806747.

28 Thomas Dueholm Hansen, Haim Kaplan, Or Zamir, and Uri Zwick. Faster k-sat algorithms using biased-ppsz. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 578–589. ACM, 2019. doi:10.1145/3313276.3316359.

29 Emmanuel Hebrard, Brahim Hnich, Barry O'Sullivan, and Toby Walsh. Finding diverse and similar solutions in constraint programming. In Manuela M. Veloso and Subbarao Kambhampati, editors, *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA*, volume 5, pages 372–377. AAAI Press / The MIT Press, 2005. URL: http://www.aaai.org/Library/AAAI/2005/aaai05-059.php.

30 Timon Hertli. Breaking the PPSZ barrier for unique 3-sat. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, volume 8572 of *Lecture Notes in Computer Science*, pages 600–611. Springer, Springer, 2014. doi:10.1007/978-3-662-43948-7_50.

31 Timon Hertli, Robin A. Moser, and Dominik Scheder. Improving PPSZ for 3-sat using critical variables. In Thomas Schwentick and Christoph Dürr, editors, *28th International Symposium on Theoretical Aspects of Computer Science, STACS 2011, March 10-12, 2011, Dortmund, Germany*, volume 9 of *LIPIcs*, pages 237–248. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2011. doi:10.4230/LIPIcs.STACS.2011.237.

32 Edward A. Hirsch. A fast deterministic algorithm for formulas that have many satisfying assignments. *Log. J. IGPL*, 6(1):59–71, 1998. doi:10.1093/jigpal/6.1.59.

33 Thomas Hofmeister, Uwe Schöning, Rainer Schuler, and Osamu Watanabe. A probabilistic 3-SAT algorithm further improved. In Helmut Alt and Afonso Ferreira, editors, *STACS 2002, 19th Annual Symposium on Theoretical Aspects of Computer Science, Antibes - Juan les Pins, France, March 14-16, 2002, Proceedings*, volume 2285 of *Lecture Notes in Computer Science*, pages 192–202. Springer, Springer, 2002. doi:10.1007/3-540-45841-7_15.

34 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

35 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.

36 Piotr Indyk, Sepideh Mahabadi, Mohammad Mahdian, and Vahab S. Mirrokni. Composable core-sets for diversity and coverage maximization. In Richard Hull and Martin Grohe, editors, *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS'14, Snowbird, UT, USA, June 22-27, 2014*, pages 100–108. ACM, 2014. doi:10.1145/2594538.2594560.

37 Daniel Kane and Osamu Watanabe. A short implicant of a CNF formula with many satisfying assignments. *Algorithmica*, 76(4):1203–1223, 2016. doi:10.1007/s00453-016-0125-z.

38 Nathan Kitchen and Andreas Kuehlmann. Stimulus generation for constrained random simulation. In Georges G. E. Gielen, editor, *2007 International Conference on Computer-Aided Design, ICCAD 2007, San Jose, CA, USA, November 5-8, 2007*, pages 258–265. IEEE, IEEE Computer Society, 2007. doi:10.1109/ICCAD.2007.4397275.

39 Sixue Liu. Chain, generalization of covering code, and deterministic algorithm for k-sat. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *LIPIcs*, pages 88:1–88:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ICALP.2018.88.

40 Alexander Nadel. Generating diverse solutions in SAT. In Karem A. Sakallah and Laurent Simon, editors, *Theory and Applications of Satisfiability Testing - SAT 2011 - 14th International Conference, SAT 2011, Ann Arbor, MI, USA, June 19-22, 2011. Proceedings*, volume 6695 of *Lecture Notes in Computer Science*, pages 287–301. Springer, Springer, 2011. doi:10.1007/978-3-642-21581-0_23.

41 Ramamohan Paturi, Pavel Pudlák, Michael E. Saks, and Francis Zane. An improved exponential-time algorithm for k-sat. *J. ACM*, 52(3):337–364, 2005. doi:10.1145/1066100.1066101.

42 Ramamohan Paturi, Pavel Pudlák, and Francis Zane. Satisfiability coding lemma. *Chic. J. Theor. Comput. Sci.*, 1999:566, 1999. URL: http://cjtcs.cs.uchicago.edu/articles/1999/11/contents.html.

43 Thierry Petit and Andrew C. Trapp. Enriching solutions to combinatorial problems via solution engineering. *INFORMS J. Comput.*, 31(3):429–444, 2019. doi:10.1287/ijoc.2018.0855.

44 Quentin Plazar, Mathieu Acher, Gilles Perrouin, Xavier Devroey, and Maxime Cordy. Uniform sampling of SAT solutions for configurable systems: Are we there yet? In *12th IEEE Conference on Software Testing, Validation and Verification, ICST 2019, Xi'an, China, April 22-27, 2019*, pages 240–251. IEEE, IEEE, 2019. doi:10.1109/ICST.2019.00032.

45 Ron M. Roth. *Introduction to Coding Theory*. Cambridge University Press, 2006.

46 Thomas J. Schaefer. The complexity of satisfiability problems. In Richard J. Lipton, Walter A. Burkhard, Walter J. Savitch, Emily P. Friedman, and Alfred V. Aho, editors, *Proceedings of the 10th Annual ACM Symposium on Theory of Computing, May 1-3, 1978, San Diego, California, USA*, pages 216–226. ACM, 1978. doi:10.1145/800133.804350.

47 Dominik Scheder. PPSZ for $k \geq 5$: More is better. *ACM Trans. Comput. Theory*, 11(4):25:1–25:22, 2019. doi:10.1145/3349613.

48 Dominik Scheder. PPSZ is better than you think. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 205–216. IEEE, IEEE, 2021. doi:10.1109/FOCS52979.2021.00028.

49 Dominik Scheder and John P. Steinberger. PPSZ for general k-sat - making hertli's analysis simpler and 3-sat faster. In Ryan O'Donnell, editor, *32nd Computational Complexity Conference, CCC 2017, July 6-9, 2017, Riga, Latvia*, volume 79 of *LIPIcs*, pages 9:1–9:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.CCC.2017.9.

50 Manuel Schmitt and Rolf Wanka. Exploiting independent subformulas: A faster approximation scheme for #k-sat. *Inf. Process. Lett.*, 113(9):337–344, 2013. doi:10.1016/j.ipl.2013.02.013.

51 Uwe Schöning. A probabilistic algorithm for k -sat based on limited local search and restart. *Algorithmica*, 32(4):615–623, 2002. doi:10.1007/s00453-001-0094-7.

52 Virginia Vassilevska Williams. Some open problems in fine-grained complexity. *SIGACT News*, 49(4):29–35, December 2018. doi:10.1145/3300150.3300158.

53 Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. New bounds for matrix multiplication: from alpha to omega. In David P. Woodruff, editor, *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*, pages 3792–3835. SIAM, SIAM, 2024. doi:10.1137/1.9781611977912.134.