Algorithms for the Diverse-k-SAT problem: the geometry of satisfying assignments

Per Austrin[1], Ioana O. Bercea[*1], Mayank Goswami[†2], Nutan Limaye[‡3], and Adarsh Srinivasan [§4]

[1] KTH Royal Institute of Technology , `{austrin,bercea}@kth.se`
[2]Queens College, City University of New York, `mayank.goswami@qc.cuny.edu`
[3]IT University of Copenhagen, `nuli@itu.dk`
[4]Rutgers University, `adarsh.srinivasan@rutgers.edu`

Abstract

Given a k-CNF formula and an integer $s \geqslant 2$, 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.

[*]Received funding from Basic Algorithms Research Copenhagen(BARC), supported by VILLUM Foundation Grants 16582 and 54451

[†]Supported by NSF grant CCF-2503086

[‡]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

[§]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. Part of this work was done during a visit to ITU Copenhagen and BARC funded by 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 [CR02] 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 [Sch78]. In particular, they also showed that the diameter problem is NP-hard even for 2-SAT.[1] On the constructive side, Angelsmark and Thapper [AT04] 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 [Cal09]). In the case of graphs, it is known that quadratic blow-up in time is unavoidable, assuming the Orthogonal Vectors Hypothesis [Wil18, AW15]. 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 3).

Dispersion. The problem of computing $s > 2$ diverse satisfying assignments to a k-CNF formula was explicitly raised by Nadel [Nad11]. Generating diverse solutions has many applications [BJM+19, ACALM22, BJKN10], and several other works have focused on finding multiple solutions to either SAT or constraint programming [ACC+10, PT19, HHOW05, PAP+19, KK07, GSS06, AB11]. 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.

There are many different ways to define the *dispersion* for a set of points (see Table 1 in [IMMM14]). 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 $s \geqslant 2$, return a set S of s satisfying assignments to $\mathbf{F}$ that maximize $\textsc{min}\mathrm{PD}(S) := \min_{z_1,z_2 \in S} d_H(z_1, z_2)$ or $\textsc{sum}\mathrm{PD}(S) := \frac{1}{2}\sum_{z_1,z_2 \in S} d_H(z_1, z_2)$. 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 10). Furthermore, for $s \geqslant 6$ we also get a faster algorithm based on clique finding (Theorem 12), that runs in time

[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.

$O^*(s^2|\Omega_{\mathbf{F}}|^{\omega\lceil s/3\rceil})$, where $\Omega_{\mathbf{F}}$ is the set of satisfying assignments of the formula $\mathbf{F}$ and $\omega \leqslant 2.38$ is the matrix multiplication exponent [WXXZ24].

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 $\mathrm{poly}(s)O^*(2^{\varepsilon_k n})$, 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 $\mathrm{poly}(s)O^*(2^{\varepsilon_k n})$ 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 $\textsc{min}\mathrm{PD}(S)$ and $\textsc{sum}\mathrm{PD}(S)$. 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 [Ip01, IpZ01, Cal09, FK13]. In a foundational work, Paturi, Pudlák, and Zane [PPZ97] 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 [Sch02] 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 [HSSW02, BS04, PPSZ05, HMS11, Her14, Liu18, SS17, HKZZ19, Sch19, Sch22], including the PPSZ algorithm by Paturi, Pudlák, Saks and Zane [PPSZ05], which is the current fastest algorithm for k-SAT.

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.

[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.

Algorithm 1: PPZ

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

Algorithm 2: Schöning

```
Input: A $k$-CNF formula $\mathbf{F}$ over $n$ variables
1  repeat $n^{O(1)} \cdot 2^{(1-\frac{1}{k \ln 2})n}$ times
2      Sample $y \sim \{0,1\}^n$
3      repeat $3n$ times
4          if $y$ satisfies $\mathbf{F}$ then
5              return $y$
6          else
7              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
8  return "not satisfiable"
```

Farthest Point Oracles Gonzalez [Gon85] 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 $\text{MINPD}(S)$. This was later extended to the sum version by [BLY12]. 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 [GGK+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 $\text{MIN-}d_H(S,x) = \min_{y \in S} d_H(x,y)$ and $\text{SUM-}d_H(S,x) = \sum_{y \in S} d_H(x,y)$. Then for some $\delta \in [0,1)$, the assignment z^* would either satisfy

$$\text{MIN-}d_H(z^*,S) \geqslant (1-\delta) \max_{z' \in \Omega_\mathbf{F}} \text{MIN-}d_H(z',S), \text{ **or** } \text{SUM-}d_H(z^*,S) \geqslant (1-\delta) \max_{z' \in \Omega_\mathbf{F}} \text{SUM-}d_H(z',S),$$

for the $\text{MINPD}(S)$ and the $\text{SUMPD}(S)$ version, respectively.

In Section 1.1, 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 1.2). 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 1.3.

1.1 Main Technical Lemmata

Recall that we are aiming for a runtime of 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 $d_H(z_0, z^*) \geqslant \left(1 - \frac{1}{k}\right) \cdot \max_{z' \in \Omega_{\mathbf{F}}} d_H(z_0, z')$. 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 $d_H(z_0, z^*) \geqslant \left(1 - \frac{4(k-1)}{(k-2)^2}\right) \cdot \max_{z' \in \Omega_{\mathbf{F}}} \text{SUM-}d_H(S, z')$. Here, z_0 is used explicitly inside the iteration.*

We sketch the proofs in Section 1.4. Three remarks are in order.

Remark 1. 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 [Sch22]. The analysis of PPZ [PPZ97] 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) [PPSZ05], can also be shown to exhibit similar behavior.

Remark 2. 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 1.4.

Remark 3. 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* [SW13] does not necessarily give a good approximation compared to our approach, even for the diameter (Appendix E). 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 [FGLS19]. While we note that the an algorithm for the MIN-ONES problem can be used to give a 1/2 approximation of the diameter(Appendix D), 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.

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 1.2. 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 1.3.

1.2 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 $\textsc{Diam}(\mathbf{F})$ 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 $\textsc{Diam}(\mathbf{F})$ is simply 0. On the other hand, if $\mathbf{F}$ is not satisfiable, we define $\textsc{Diam}(\mathbf{F}) = \perp$. For a set $S \subseteq \{0,1\}^n$, define $\textsc{minPD}(S) := \min_{z_1,z_2 \in S} d_H(z_1,z_2)$ and $\textsc{sumPD}(S) := \frac{1}{2}\sum_{z_1,z_2 \in S} d_H(z_1,z_2)$. We then define $\textsc{Opt-sum}(\mathbf{F},s)$ as the maximum value of $\textsc{sumPD}(S)$ over all multisets S with s satisfying assignments (including multiplicities), and
$\textsc{Opt-min}(\mathbf{F},s) = \max_{S \subseteq \Omega_{\mathbf{F}}, |S|=s} \textsc{minPD}(S)$, i.e., the maximum such distance over all sets of s satisfying assignments. Further, we define $\textsc{Opt-sum}_{\neq}(\mathbf{F},s)$ as the maximum value of $\textsc{sumPD}(S)$ over all *sets* S with s distinct satisfying assignments.

1.2.1 Computing diameter exactly and approximately

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

Theorem 3. *[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 $d_H(z_1,z_2) = \textsc{Diam}(\mathbf{F})$.

Prior to our work, the best exact algorithm known was by Angelsmark and Thapper [AT05]. 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 $\textsc{Diam}(\mathbf{F})$ 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 $\textsc{Min-Ones}$ 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 $\mathbf{F}$. For more details on this reduction, we refer the reader to Appendix D. By

[4]All approximation algorithms we present here use poly(n) space.

using the best known algorithms for k-SAT by Paturi, Pudlák, Saks, and Zane [PPSZ05] and for Min-Ones by Fomin, Gaspers, Lokshtanov and Saurabh [FGLS19], 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 $k \geqslant 3$ with a small loss in the approximation factor. From here on, we assume that $k \geqslant 3$ unless stated otherwise.

Theorem 4 (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 $d_H(z_1^*, z_2^*) \geqslant \frac{1}{2}\cdot\left(1-\frac{1}{k}\right)\textsc{Diam}(\mathbf{F})$ 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 [PPZ97], 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 [PPZ97]. 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 1.4.

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 5 (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 $d_H(z_1^*, z_2^*) \geqslant \frac{1}{2}\left(1-\frac{4(k-1)}{(k-2)^2}\right)\cdot\textsc{Diam}(\mathbf{F})$ with probability $1-o(1)$.*

In fact, Theorem 5 is one instance of a smooth tradeoff between the approximation factor and the running time. We present the full tradeoff in Theorem 25 Section 4. Notice that the running time obtained here is better than the running time obtained using Theorem 4, 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 $k \geqslant 7$. Theorem 25 generalizes Theorem 5 to get non-trivial approximation factors for any k. In Theorem 41 Appendix A.1, we present another Schöning-type algorithm to approximate the diameter that outperforms the algorithm in Theorem 41 for small values of k and some regimes of the approximation factor.

1.2.2 Computing dispersion exactly and approximately

We extend all the algorithms from Section 1.2.1 and obtain bounds for the dispersion problem.

Exact algorithms for dispersion. We start with the problem of exactly computing $\textsc{Opt-sum}(\mathbf{F}, s)$, $\textsc{Opt-min}(\mathbf{F}, s)$ and $\textsc{Opt-sum}_{\neq}(\mathbf{F}, s)$. 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 3 to do this in $O^*(2^{(s-1)n})$ time and $O^*(2^n)$ space. We state and prove the formal statement in Section 2.

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

We also provide an alternate algorithm for dispersion in Theorem 12 in Section 2.3. The algorithm, based on clique-finding, 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 $\omega \leqslant 2.38$ denotes the matrix multiplication exponent [WXXZ24]. As such, it is faster than the Fourier analysis-based algorithm for any $s \geqslant 6$, and can be much faster when the size of the solution set is less than 2^n.

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. More about this in Section 1.4.

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 6 (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 $s \geqslant 1$ 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*

$$\text{SUM}PD(S) \geqslant \left(1 - \frac{4}{k-3}\right)\left(1 - \frac{2}{s+2}\right) \cdot \text{OPT-SUM}(\mathbf{F}, s) \; .$$

Remark 4. When $k \leqslant 6$, 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 to the reader to the full version of this theorem (Theorem 15) in Section 3).

For OPT-SUM$_{\neq}$($\mathbf{F}, s$), we can obtain exactly the same approximation factors as in Theorem 15 for certain parameter regimes of s (see Appendix C for more details).

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

Theorem 7 (PPZ approximating 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 $s \geqslant 1$ as input and if $\mathbf{F}$ is satisfiable and $|\Omega_{\mathbf{F}}| \geqslant s$, with probability at least $1 - o(1)$, outputs a set S of size s such that MIN$PD(S) \geqslant \frac{1}{2}\left(1 - \frac{1}{kH^{-1}(1-1/k)}\right) \cdot$ OPT-MIN($\mathbf{F}, s$)* [6]

Note that, in the above statement, the approximation factor is non-trivial (> 0) only for $k \geqslant 5$. We note that we can also obtain Schöning-type running time bounds for dispersion for $k \geqslant 2$. We achieve this by extending Theorem 5. The statements of our results and their proofs appear in Section 4.

Approximating OPT-MIN($\mathbf{F}, s$) for heavy-weight solutions. We now consider a heavy-weight variant of OPT-MIN($\mathbf{F}, s$). Formally, for a k-CNF formula $\mathbf{F}$, we let $\Omega_{\mathbf{F},\geqslant W}$ denote the set of

[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 $[0, 1/2]$.

satisfying assignments to $\mathbf{F}$ with Hamming weight at least W. We then define

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

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

Theorem 8 (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. *$S^* \subseteq \Omega_{\mathbf{F},\geqslant(1-\delta)W}$ of size s such that $\textsc{min}PD(S^*) \geqslant \frac{1}{2}(1-\delta)\,\textsc{Opt-min}(\mathbf{F}, s, \geqslant W)$ if $\mathbf{F}$ is satisfiable and $|\Omega_{\mathbf{F},\geqslant W}| \geqslant s$.*
2. *$S^* \subseteq \Omega_{\mathbf{F},\leqslant(1+\delta)W}$ of size s such that $\textsc{min}PD(S^*) \geqslant \frac{1}{2}(1-\delta)\,\textsc{Opt-min}(\mathbf{F}, s, \leqslant W)$ if $\mathbf{F}$ is satisfiable and $|\Omega_{\mathbf{F},\leqslant W}| \geqslant s$,*

Remark 5. 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 8 are non-trivial only for $k \geqslant 7$. However, just like the case of Theorem 5, Theorem 8 can be generalized, obtaining running time bounds for any k and for a larger range of approximation factors (Theorem 26). Further, we can prove that an analogous result exists for the sum of distances dispersion measure. We refer the reader to Section 4 for the complete theorem statements and proofs.

1.3 Generalizations and applications.

1. Isometric Reductions. Dispersion has also been studied when the space is induced by solutions to some NP-complete optimization problem [BJM$^+$19, BFJ$^+$22]. To address this optimization aspect, we first generalize our techniques to give dispersed solutions of high (or low) Hamming 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 [FGLS19]. This allows us to extend our results to

[7]In a recent work, Gurumukhani, Paturi, Pudlák, Saks, and Talebanfard [GpP$^+$24] 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.

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 [BJM+19, BFJ+22]. Our methods show that by relaxing to bi-approximations, this dependence on s can be made polynomial.

Optimization Problem	One optimal solution [FGLS19]	Multiple approximately optimal, approximately dispersed solutions
d-Hitting Set ($d \geqslant 3$)	$(2-\frac{1}{d})^n$	Theorem 35
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	$s^3 \cdot 1.6598^n$
Node Unique Label Cover	$(2-\frac{1}{\lvert\Sigma\rvert^2})^n$	Theorem 36
Vertex (r,ℓ)-Partization ($r,\ell \leqslant 2$)	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	$s^3 \cdot 1.6167^n$
d-Hitting Set ($d \geqslant 3$)	$(2-\frac{1}{d-0.9245})^n$	Theorem 36
Min-Ones 3-SAT	$s^3 \cdot 1.6097^n$	Theorem 26
Min-Ones d-SAT ($d \geqslant 4$)	$(2-\frac{1}{d})^n$	Theorem 26
Weighted d-SAT ($d \geqslant 3$)	$(2-\frac{1}{d})^n$	Theorem 26
Weighted Feedback Vertex Set	1.7237^n	$s^3 \cdot 1.6432^n$
Weighted 3-Hitting Set	1.5388^n	$s^3 \cdot 1.5612^n$
Weighted d-Hitting Set ($d \geqslant 4$)	$(2-\frac{1}{d-0.832})^n$	Theorem 36

Table 1: The second column contains the time taken to obtain one exact solution using methods in [FGLS19]. 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.)

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

[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 Theorem 36

solutions, such as in Hirsch [Hir98] and Kane and Watanabe [KW16]) and in the random SAT setting, e.g., the diameter by Feige, Flaxman and Vilenchik [FFV11] and the giant connected component by Chen, Mani, Moitra [CMM23]).

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 $2^{(1-1/k)(1-\delta)n}$ [CIKp08]. Cardinal, Nummenpalo and Welzl [CNW17] 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 [Sch19].

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 (Section 5.3). 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 [Sch19], then our results formalize the intuition that more dispersed solutions are even better.*

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 [Rot06]. 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 [GHK10, GKS12]. 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 (Section 5.4).

1.4 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. We include the technical details for this part of our analysis in Appendix B. There, we also show how to adapt Cevallos, Eisenbrand, and Zenklusen's local search algorithm [CEZ19] for our setting.

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* [PPZ97] states that for every j-isolated satisfying assignment z, it holds that $\tau(\mathbf{F}, z) \geqslant 2^{-n+j/k}$. 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 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

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

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 $U_j = \bigcup_{j' \geqslant j} V_{j'}$. 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 $\Omega_{\mathbf{F}} \backslash U_j$ 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

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

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 $\Omega_{\mathbf{F}} \backslash A$ (Lemma 17). 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 18, 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

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

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 (Lemma 19). 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 18 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 18 on the antipode of z). We can also assume that we have a lower bound on $\max_{z' \in \Omega_{\mathbf{F}}} \textsc{min}\text{-}d_H(S, z')$ 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 [Sch02]. 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 $((1/2 \cdot (1 + 1/(k-1)))^n$.

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).

1.5 Organization of the paper.

In Section 2, we present and analyse our algorithms for exact diameter and dispersion (Theorem 3, Theorem 10, and Theorem 12). In Section 3, we present our PPZ-based algorithms for approximately computing diameter and dispersion (Theorem 4, Theorem 15 and Theorem 7). In Section 4, we present our Schöning-based algorithms for diameter, dispersion and weighted dispersion (Theorem 25, Theorem 27, Theorem 26). In Section 5, we present our results on diversity preserving reductions and applications of parameterized local feasibility search and prove the results presented in Table 2.

2 Exact algorithms for diameter and dispersion

In this section, we present our algorithm for diameter (Theorem 3) and two algorithms for dispersion (Theorem 10 and Theorem 12). The problem of computing $\textsc{Diam}(\mathbf{F})$ has been studied by Angelsmark and Thapper [AT04]. They give an algorithm that runs in $O^*((2a_k)^n)$ time and $n^{O(1)}$ space, where $O^*(a_k^n)$ is the run-time of a k-SAT solver. Note that the strong exponential hypothesis implies that $\lim_{k \to \infty} a_k = 2$. We observe that there exists an algorithm to compute $\textsc{Diam}(\mathbf{F})$ exactly, using $O^*(2^n)$ time and $O^*(2^n)$ space. Then, we give two algorithms that compute $\textsc{Opt-min}(\mathbf{F}, s)$ and

Opt-sum$(\mathbf{F}, s)$ in time $O^*(2^{(s-1)n})$ and $O^*(2^{n\omega\lceil s/3\rceil})$, where $\omega \leqslant 2.38$ is the matrix multiplication exponent. In fact, these algorithms do not use the fact that $\mathbf{F}$ is a k-CNF formula. We formally define the setup below.

Preliminaries. Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function computable by an oracle. Our algorithms use Fourier analysis of Boolean functions, and we briefly recall some facts first.

Definition 1 (Fourier Transform)**.** Given any function $f : \{0,1\}^n \to \mathbb{R}$, the Fourier transform of f is defined as follows.

$$\hat{f}(y) := \sum_{x \in \{0,1\}^n} (-1)^{\langle x,y\rangle} f(x) ,$$

where $\langle x, y\rangle = \sum_{i=1}^n x_i y_i$.

Definition 2 (Convolution)**.** Given two functions $f, g : \{0,1\}^n \to \mathbb{R}$, we define their convolution to be

$$(f * g)(y) := \sum_{x \in \{0,1\}^n} f(x) g(x \oplus y) ,$$

where $\oplus$ represents bit-wise addition, modulo 2. Any function $f : \{0,1\}^n \to \mathbb{R}$ can be represented as a column vector $f \in \mathbb{R}^{2^n}$, by indexing the columns using $\{0, 1, \ldots, 2^n - 1\}$. It can be shown that $\hat{f} = H_{2^n} \cdot f$, where H_{2^n} is the $2^n \times 2^n$ Walsh-Hadamard matrix, which is inductively defined as follows:

$$H_1 = \begin{bmatrix}1\end{bmatrix}, H_{2^{m+1}} = \begin{bmatrix} H_{2^m} & H_{2^m} \\ H_{2^m} & -H_{2^m} \end{bmatrix} \text{ for all } m \geqslant 1 .$$

Given the vector f, the vector $\hat{f}$ can be computed by a divide and conquer algorithm called the fast Walsh-Hadamard transform that uses $O(n \cdot 2^n)$ operations. Also, note that $f(x) = \frac{1}{2^n}\sum_{y \in \{0,1\}^n} (-1)^{\langle x,y\rangle} \hat{f}(y)$. Further, for any two functions $f, g : \{0,1\}^n \to \mathbb{R}$, $\widehat{f * g}(x) = \hat{f}(x)\hat{g}(x)$, for every $x \in \{0,1\}^n$. This implies that given the vectors $f, g \in \mathbb{R}^{2^n}$, the vector $f * g \in \mathbb{R}^{2^n}$ can be computed in $O(n \cdot 2^n)$ time. For more details and proofs of the above facts, we refer the reader to [O'D21].

2.1 Computing the diameter of Boolean functions: the proof of Theorem 3

To define the exact diameter of f, we slightly abuse notation and define

$$\textsc{Diam}(f) = \max_{z_1, z_2 \in f^{-1}(1)} d_H(z_1, z_2) .$$

We relate computing $\textsc{Diam}(f)$ to evaluating the vector $(f * f)$.

Lemma 9. *For any $y \in \{0,1\}^n$, there exist $z_1, z_2 \in f^{-1}(1)$ with $z_1 \oplus z_2 = y$ if and only if $(f * f)(y) > 0$.*

Proof. TOPROVE 0 □

The above lemma motivates the following algorithm:

```
Algorithm 3: Exact diameter using Fourier transform
  Input: A black box computing a Boolean function $f : \{0,1\}^n \to \{0,1\}$
  Output: $z_1, z_2 \in f^{-1}(1)$ such that $d_H(z_1, z_2) = \textsc{Diam}(f)$ if $f^{-1}(1) \neq \varnothing$, $\perp$ if $f^{-1}(1) = \varnothing$
1 Compute the vector $f \in \mathbb{R}^{2^n}$ of values of $f$.
2 Using the fast Walsh-Hadamard transform, compute the vector $\hat{f} = H_{2^n} \cdot f \in \mathbb{R}^{2^n}$. Multiply
   each element of this vector with itself to obtain the vector $\hat{f}^2 \in \mathbb{R}^{2^n}$.
3 Compute the vector $(f * f) = \frac{1}{2^n} H_{2^n} \cdot \hat{f}^2$ using the fast Walsh-Hadamard transform. Let
   $z \in \{0,1\}^n$ be any of the vectors with largest Hamming weight such that $(f * f)(z) > 0$.
   Output $\perp$ if there is no such $z$, abort.
4 Find any $x \in \{0,1\}^n$ such that $f(x) = f(x \oplus z) = 1$ and output $x, x \oplus z$.
```

Each step of this algorithm uses $O^*(2^n)$ time and $O^*(2^n)$ space, which proves Theorem 3.

2.2 Exact algorithms for dispersion using Fourier transforms

We now generalize the above algorithm for diameter to dispersion, where our objectives are defined over the $f^{-1}(1)$ (similarly as in the diameter case). In the following section we present another algorithm with faster running time, but that algorithm works for $s \geqslant 6$. Our algorithm presented below can be used for all values of s.

Theorem 10. *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 $\textsc{sum}PD(z_1, z_2, \ldots, z_s) = \textsc{Opt-sum}(f, s)$.*
2. *The output of $\mathcal{A}_2$ is a set $\{z_1, z_2, \ldots, z_s\} \subseteq f^{-1}(1)$ such that $\textsc{min}PD(z_1, z_2, \ldots, z_s) = \textsc{Opt-min}(f, s)$.*
3. *If $|f^{-1}(1)| \geqslant s$, the output of $\mathcal{A}_3$ is a set $\{z_1, z_2, \ldots, z_s\} \subseteq f^{-1}(1)$ such that $\textsc{sum}PD(z_1, z_2, \ldots, z_s) = \textsc{Opt-sum}_{\neq}(f, s)$.*

We now prove Theorem 10. We begin by observing that for every $(z_0, z_1, \ldots, z_{s-1}) \in \{0,1\}^{sn}$, $\textsc{sum}\mathrm{PD}(z_0, z_1, \ldots, z_{s-1}) = \textsc{sum}\mathrm{PD}(0, z_1 \oplus z_0, \ldots, z_{s-1} \oplus z_0)$ and $\textsc{min}\mathrm{PD}(z_0, z_1, \ldots, z_{s-1}) = \textsc{min}\mathrm{PD}(0, z_1 \oplus z_0, \ldots, z_{s-1} \oplus z_0)$. Hence, the value of $\textsc{sum}\mathrm{PD}(z_0, z_1, \ldots, z_{s-1})$ and $\textsc{min}\mathrm{PD}(z_0, z_1, \ldots, z_{s-1})$ are determined entirely by $y_1, y_2, \ldots, y_{s-1}$, where $y_j = z_0 \oplus z_j$ for each $j \in \{1, 2, \ldots, s-1\}$. Next, we prove the following generalization of Lemma 9.

Lemma 11. *For every $w_1, w_2, \ldots, w_{s-2} \in \{0,1\}^n$, define the function $g_{(w_1, w_2, \cdots, w_{s-2})}(x) := f(x) f(x \oplus w_1) f(x \oplus w_2) \ldots f(x \oplus w_{s-2})$. For every $y, \in \{0,1\}^n$ and $w_1, w_2, \ldots, w_{s-2} \in \{0,1\}^n$, there exist $z_0, z_1, \ldots, z_{s-1} \in f^{-1}(1)$, such that $z_1 = z_0 \oplus y$ and $z_j = z_0 \oplus w_{j-1} \oplus y$ for each $j \in \{2, 3, \ldots, s-1\}$ if and only if $f * g_{(w_1, w_2, \cdots, w_{s-2})}(y) > 0$.*

Proof. TOPROVE 1 □

Hence, for each $w_1, w_2, \ldots, w_{s-2} \in \{0,1\}^n$, we can run the following procedure to compute an array containing the values of $f * g_{(w_1, w_2, \cdots, w_{s-2})}(y)$ for every $y \in \{0,1\}^n$.

```
Algorithm 4: Algorithm to compute convolution of $f$ and $g_{w_1,w_2,\ldots,w_{s-2}}$.
  Input: A black box computing $f : \{0,1\}^n \to \{0,1\}$, $w_1, w_2, \ldots, w_{s-2} \in \{0,1\}^n$.
  Output: An array $f * g_{(w_1,w_2,\cdots,w_{s-2})} \in \mathbb{R}^{2^n}$ containing the values of $f * g_{(w_1,w_2,\cdots,w_{s-2})}(y)$
          for every $y \in \{0,1\}^n$.
1 Compute the vectors $f, g_{(w_1,w_2,\cdots,w_{s-2})} \in \mathbb{R}^{2^n}$ with the values of $f(x)$ and $g_{(w_1,w_2,\cdots,w_{s-2})}(x)$
   for each $x \in \{0,1\}^n$.
2 Compute the vectors $\hat{f} = H_{2^n} \cdot f, \hat{g}_{(w_1,w_2,\cdots,w_{s-2})} = H_{2^n} \cdot g_{(w_1,w_2,\cdots,w_{s-2})}$ using the fast
   Walsh-Hadamard transform.
3 Compute the vector $\hat{f} \cdot \hat{g}_{(w_1,w_2,\cdots,w_{s-2})} \in \mathbb{R}^{2^n}$ by multiplying the elements of $\hat{f}$ and
   $\hat{g}_{(w_1,w_2,\cdots,w_{s-2})}$ element-wise.
4 Compute the vector $f * g_{(w_1,w_2,\cdots,w_{s-2})} = \frac{1}{2^n} H_{2^n} \cdot \left(\hat{f} \cdot \hat{g}_{(w_1,w_2,\cdots,w_{s-2})}\right)$ using the fast
   Walsh-Hadamard transform.
```

This implies that by iterating over all $(w_1, w_2, \cdots, w_{s-2}) \in \{0,1\}^{(s-2)n}$, we can compute Opt-sum$(f,s)$ and Opt-min(f,s) using $O^*(2^{(s-1)n})$ time and $O^*(2^n)$ space. We formally define the algorithm below. Note that we have defined it to compute Opt-sum(f,s), but the same algorithm with minor modifications can be used to compute Opt-min(f,s) and Opt-sum$_{\neq}(f,s)$.

```
Algorithm 5: Algorithm for exact dispersion using Fourier transforms
  Input: A black box computing a Boolean function $f : \{0,1\}^n \to \{0,1\}$
  Output: $z_1, z_2, \ldots z_s \in f^{-1}(1)$ such that sum-$d_H(z_1, z_2, \ldots, z_s)$ = Opt-sum$(f,s)$ if
          $f^{-1}(1) \neq \varnothing$, $\perp$ if $f^{-1}(1) = \varnothing$
1  Initialize $\mathcal{M} = \perp, y_1, y_2, \ldots, y_{s-1} = \perp$.
2  for $(w_1, w_2, \ldots, w_{s-2}) \in \{0,1\}^{(s-2)n}$ do
3      Compute an array containing the values of $f * g_{(w_1,w_2,\cdots,w_{s-2})}(y)$ for each $y \in \{0,1\}^n$
        using Algorithm 4.
4      for $y \in \{0,1\}^n$ do
5          if $f * g_{(w_1,w_2,\cdots,w_{s-2})}(y) > 0$ and sumPD$(0, y, y \oplus w_1, y \oplus w_2, \ldots, y \oplus w_{s-2}) > \mathcal{M}$
            then
6              set $\mathcal{M} :=$ sumPD$(0, y, y \oplus w_1, y \oplus w_2, \ldots, y \oplus w_{s-2})$ ,
                $y_1 = y, y_2 = y \oplus w_1, \ldots, y_{s-1} = y \oplus w_{s-2}$.
7  if $\mathcal{M} = \perp$ then
8      output $\perp$
9  else
10     If there exists $x \in \{0,1\}^n$ such that $f(x) = f(x \oplus y_1) = \ldots, f(x \oplus y_{s-1}) = 1$, output
        $z_0 = x, z_1 = x \oplus y_1, z_2 = x \oplus y_1, \ldots, z_{s-1} = x \oplus y_{s-1}$
```

Remark 6. To design an algorithm for Opt-min(f,s), we replace the comparison in line 5 of the algorithm with one using minPD instead of sumPD. An algorithm to compute Opt-sum$_{\neq}(f,s)$ would be identical, except that we would iterate over $w_1, w_2, \ldots, w_{s-2}$ such that they are all different, and in the inner loop, we would iterate over all $y \neq \mathbf{0}$.

Proof of correctness: Define the n-dimensional subspace $V \subseteq \{0,1\}^{(s-1)n}$ to be $\{(x, x, \ldots, x) \mid x \in \{0,1\}^n\}$, which partitions $\{0,1\}^{(s-1)n}$ into the $2^{(s-2)n}$ cosets $V_{(w_1,w_2,\cdots,w_{s-2})} = \{(x, x \oplus w_1, x \oplus$

$w_2, \ldots, x \oplus w_{s-2}) \mid x \in \{0,1\}^n\}$ for each s-tuple $(w_1, w_2, \cdots, w_{s-2}) \in \{0,1\}^{(s-2)n}$. Lemma 11 implies that for each $(y_1, y_2, \ldots, y_{s-1}) = (y, y \oplus w_1, y \oplus w_2, \ldots, y \oplus w_{s-2}) \in V_{(w_1, w_2, \cdots, w_{s-2})}$, there exists $z_0, z_1, \ldots, z_s \in f^{-1}(1)$ with $z_j = z_0 \oplus y_j$ for $j \in \{1, 2, \ldots, s-1\}$ if and only if $f * g_{(w_1, w_2, \cdots, w_{s-2})}(y) > 0$. This completes the proof of Theorem 10.

2.3 Exact Algorithms for Dispersion Using Clique-Finding

In this section, we discuss an alternate technique for exactly computing dispersion. The running time and space of the algorithm depend on the size of the solution space $\Omega_{\mathbf{F}}$. For any $s \geqslant 6$, the algorithm runs faster than the one in Section 2, but at the cost of potentially higher space.

We now formulate our results to work for dispersion over an arbitrary subset X of the hypercube, of size M. We thus slightly abuse notation and define Opt-sum(X, s), Opt-min(X, s) and Opt-sum$_{\neq}(X, s)$. In what follows, $\omega \leqslant 2.38$ denotes the matrix multiplication exponent [WXXZ24].

Theorem 12. *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 $O(\mathrm{poly}(n, s) \cdot M^{\omega \lceil s/3 \rceil})$ 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 $|S| \geqslant s$, 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$_{\neq}(X, s)$.*

Note that when applied with X being the set of satisfying assignments to a formula $\mathbf{F}$, the running time is at worst $O(2^{\omega \lceil s/3 \rceil n})$ but in general much faster depending on the number of satisfying assignments. Furthermore, these algorithms do not rely on the underlying space being $\{0,1\}^n$; they can be used on any M-point metric space.

The algorithms use the same idea as $O(n^{c_s s})$ time algorithms for finding a clique of size s in a graph, where $c_s \approx \omega/3$ with variations depending on s mod 3 [EG04]. In particular the Opt-min problem immediately reduces to the s-clique problem by creating a graph on X where $x, y \in X$ are connected by an edge if their distance is at least d (for some guess $d \in [0, n]$ for the value of Opt-min(X, s), which we can then binary search over). Similarly for the Opt-sum objective function, the problem reduces to finding an s-clique of maximum weight in an edge-weighted graph, which can be solved by similar methods. Similar ideas have been used before in for example [Wil05].

Let us describe the algorithms in more detail, starting with the case of Opt-min since it is easier. While in this case the reduction to s-clique described above could be used directly, let us still take a slightly longer route and reduce to triangle-finding, in order to provide a warm-up for the Opt-sum algorithm where this is needed.

To simplify notation we assume that s is divisible by 3. Given a guess $d \in [0, n]$ for the value of Opt-min(X, s), define a graph G_d where the vertex set is

$$V(G_d) = \left\{ (x_1, \ldots, x_{s/3}) \in X^{s/3} \mid d_H(x_i, x_j) \geqslant d \text{ for all } 1 \leqslant i < j \leqslant s/3 \right\}.$$

Two vertices $(x_1, \ldots, x_{s/3})$ and $(y_1, \ldots, y_{s/3})$ are connected by an edge if $d_H(x_i, y_j) \geqslant d$ for all i and j. Note that G_d has $O(M^{s/3})$ vertices and $O(M^{2s/3})$ edges, and can be constructed in $O(s^2 \cdot M^{2s/3})$ time.

Claim 13. *Three tuples $(x_1,\dots,x_{s/3})$, $(y_1,\dots,y_{s/3})$, and $(z_1,\dots,z_{s/3})$ form a triangle in G_d if and only if* $\textsc{min}PD(x_1,\dots,x_{s/3},y_1,\dots,y_{s/3},z_1,\dots,z_{s/3}) \geqslant d$.

This immediately gives us the algorithm $\mathcal{A}_2$ for $\textsc{Opt-min}(X,s)$: try all possible values of d, construct the graph G_d, and then search for a triangle in G_d, which can be done in $O(|V(G_d)|^{\omega}) = O(M^{\omega s/3})$ time [IR77].

Moving on to the $\textsc{Opt-sum}$ objective function, we change the above algorithm as follows. Given six values $\vec{d} = (d_1,d_2,d_3,d_{12},d_{23},d_{13}) \in [0,sn]^6$, we define the tri-partite graph $G_{\vec{d}}$ with vertex sets V_1,V_2,V_3 defined by

$$V_k(G_{\vec{d}}) = \left\{(x_1,\dots,x_{s/3}) \in X^{s/3} \mid \frac{1}{2}\sum_{i,j} d_H(x_i,x_j) \geqslant d_k\right\}.$$

Two vertices $(x_1,\dots,x_{s/3}) \in V_k$ and $(y_1,\dots,y_{s/3}) \in V_{k'}$ are connected by an edge if

$$\sum_{i,j} d_H(x_i,y_j) \geqslant d_{k,k'}.$$

We then have the following claim, which yields the algorithm $\mathcal{A}_1$ (by enumerating all $O((ns)^6)$ possible values of $\vec{d}$).[9]

Claim 14. *If three vertices $(x_1,\dots,x_{s/3}) \in V_1$, $(y_1,\dots,y_{s/3}) \in V_2$, and $(z_1,\dots,z_{s/3})$ form a triangle in $G_{\vec{d}}$ then* $\textsc{sum}PD(x_1,\dots,x_{s/3},y_1,\dots,y_{s/3},z_1,\dots,z_{s/3}) \geqslant d_1+d_2+d_3+d_{12}+d_{23}+d_{13}$. *Conversely, there exists a $\vec{d}$ such that $d_1+d_2+d_3+d_{12}+d_{23}+d_{13} \geqslant \textsc{Opt-sum}(S,s)$ and $G_{\vec{d}}$ has a triangle.*

Finally, to get the algorithm $\mathcal{A}_3$ for $\textsc{Opt-sum}_{\neq}(X,s)$, we simply change the definition of the vertices and edges of $G_{\vec{d}}$ to exclude any tuples with repeated strings.

3 The PPZ algorithm performs geometry-based sampling

This section is devoted to proving Theorem 4, Theorem 6 and Theorem 7, which we restate below. In fact, we prove a slightly stronger version of Theorem 6, which is stated here.

Theorem 4 (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 $d_H(z_1^*,z_2^*) \geqslant \frac{1}{2}\cdot\left(1-\frac{1}{k}\right)\textsc{Diam}(\mathbf{F})$ with probability $1-o(1)$.*

We now state the full version of Theorem 6.

Theorem 15 (PPZ approximating $\textsc{Opt-sum}(\mathbf{F},s)$). *Let $\mathbf{F}$ be a k-CNF formula on n variables. There exists a randomized algorithm running in time $O^*\left(s^4\cdot 2^{n-n/k}\right)$ that takes $\mathbf{F}$ and an integer $s \geqslant 1$ 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:*

1. $\textsc{sum}PD(S^*) \geqslant \frac{1}{2}\cdot\left(1-\frac{2}{k+1}\right)\cdot\textsc{Opt-sum}(\mathbf{F},s)$ *if* $s \leqslant 3+\left\lfloor\frac{4}{k-1}\right\rfloor$.

[9] Note that, if we reduced $\textsc{Opt-sum}$ to an s-clique problem instead of triangle finding, there would be $\binom{s}{2}$ distances to guess, which would lead to an extra runtime factor of roughly $n^{s^2/2}$. This is why we reduce to triangle-finding instead.

2. $\text{SUM}PD(S^*) \geqslant \frac{k-1}{k+3}\left(\frac{1-\frac{1}{s}}{1+\frac{k-1}{(k+3)}\cdot\frac{1}{s}}\right)\cdot \text{OPT-SUM}(\mathbf{F}, s)$ *if* $s \geqslant 3 + \left\lceil \frac{4}{k-1} \right\rceil$.

Theorem 7 (PPZ approximating 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* $s \geqslant 1$ *as input and if* $\mathbf{F}$ *is satisfiable and* $|\Omega_{\mathbf{F}}| \geqslant s$*, with probability at least* $1 - o(1)$*, outputs a set* S *of size* s *such that* $\text{MIN}PD(S) \geqslant \frac{1}{2}\left(1 - \frac{1}{kH^{-1}(1-1/k)}\right)\cdot \text{OPT-MIN}(\mathbf{F}, s)$ [10]

Proof organization: We prove the above three theorems in parallel using the following five step procedure.

1. In Section 3.1, we summarize the PPZ algorithm and state the satisfiability coding lemma.
2. In Section 3.2, we prove the *separator lemma*, that generalizes the satisfiablity coding lemma.
3. In Section 3.3, we prove *geometric sampling properties* of PPZ, with respect to DIAM and OPT-MIN in Lemma 18, and OPT-SUM in Lemma 19.
4. In Section 3.4, we use these geometric properties to develop farthest point oracles for OPT-MIN and OPT-SUM.
5. In Section 3.5, we describe our algorithms for finding dispersed solutions with respect to OPT-SUM and OPT-MIN. These algorithms use the farthest point oracles in the well known algorithms for dispersion studied by Gonzales [Gon85] and Cevallos, Eisenbrand and Zenklusen [CEZ19].

Notation. We use a graph theoretical framework to analyze the PPZ algorithm. Let $G_{\mathbf{F}}$ be the subgraph of the n-dimensional boolean hypercube induced by the set of satisfying assignments of $\mathbf{F}$. That is, the vertex set of $G_{\mathbf{F}}$ is $\Omega_{\mathbf{F}}$, and $z, z' \in \Omega_{\mathbf{F}}$ are connected in $G_{\mathbf{F}}$ if $d_H(z, z') = 1$. For any (z, z') connected in $G_{\mathbf{F}}$, $z' = z \oplus e_k$ for some $k \in [n]$, where $e_k \in \{0,1\}^n$ is the k-th standard basis vector. For any $z \in \Omega_{\mathbf{F}}$, we use $\deg(z)$ to denote its degree in the graph $G_{\mathbf{F}}$.

3.1 The PPZ algorithm

In this section, we formally define the subroutine used in the PPZ algorithm and recall its analysis.

PPZ-Modify. This subroutine takes as input a k-CNF formula $\mathbf{F}$, a string $y \in \{0,1\}^n$, and a permutation $\pi \in \mathcal{S}_n$ of length n. It iteratively computes a string $u \in \{0,1\}^n$ in n steps.
Let $\mathbf{F}_0 = \mathbf{F}$. In each step i, the algorithm computes $u_{\pi(i)}$ and updates the formula $\mathbf{F}_{i-1}$ to $\mathbf{F}_i$ as follows: if $\mathbf{F}_{i-1}$ has a clause $C = (x_{\pi(i)})$, then it sets $u_{\pi(i)}$ to 1; if it has a clause $C = (\overline{x_{\pi(i)}})$ then it sets $u_{\pi(i)}$ to 0, and if there is no such clause, i.e., any clause containing the variable $x_{\pi(i)}$ has two or more variables, then it sets $u_{\pi(i)}$ equal to $y_{\pi(i)}$. It updates $\mathbf{F}_{i-1}$ to $\mathbf{F}_i$ by setting all instances of the variable $x_{\pi(i)}$ as per $u_{\pi(i)}$ and simplifying the formula as needed (i.e., removing satisfied clauses and eliminating 0-valued literals from all clauses). After n steps, the algorithm outputs $u \in \{0,1\}^n$ as computed above.

For any $z \in \{0,1\}^n$, let $\tau(\mathbf{F}, z)$ denote the probability that PPZ-Modify$(\mathbf{F}, y, \pi)$ outputs z when y and π are chosen independently and uniformly at random from $\{0,1\}^n$ and $\mathcal{S}_n$, respectively. For

[10]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 $[0, 1/2]$.

any subset $A \subseteq \{0,1\}^n$, we use $\tau(\mathbf{F}, A)$ to denote the probability that PPZ-Modify$(\mathbf{F}, y, \pi)$ outputs an assignment in A over y and π chosen independently and uniformly at random. For any fixed π, y, the procedure PPZ-Modify outputs a fixed assignment that only depends on π and y, which implies that

$$\tau(\mathbf{F}, A) = \sum_{z \in A} \tau(\mathbf{F}, z) \ .$$

In their paper [PPZ97], Paturi, Pudlák and Zane proved the satisfiability coding lemma, which states that for a satisfying assignment z, $\tau(\mathbf{F}, z)$ depends on how *isolated* z is (i.e, its degree in $G_{\mathbf{F}}$).

Lemma 16 (Satisfiability Coding Lemma (Paturi, Pudlák, Zane [PPZ97]))**.** *Let $\mathbf{F}$ be a k-CNF formula on n variables. Let y be chosen uniformly at random from $\{0,1\}^n$ and π be chosen uniformly at random from $\mathcal{S}_n$. Let z be a satisfying assignment of $\mathbf{F}$ such that $\deg(z) = n - j$ for some $j \in [n]$. Then, the probability that PPZ-Modify$(\mathbf{F}, y, \pi)$ outputs z is at least $2^{-n+j/k}$.*

If $\Omega_{\mathbf{F}}$ is non-empty (i.e, $\mathbf{F}$ is satisfiable), they show that $\sum_{z \in \Omega_{\mathbf{F}}} 2^{-\deg(z)/k} \geqslant 1$, which implies the following lower bound on the probability that PPZ-Modify outputs any satisfying assignment to $\mathbf{F}$.

$$\tau(\mathbf{F}, \Omega_{\mathbf{F}}) = \sum_{z \in \Omega_{\mathbf{F}}} \tau(\mathbf{F}, z) = 2^{-n+n/k} \sum_{z \in \Omega_{\mathbf{F}}} 2^{-\deg(z)/k} \geqslant 2^{-n+n/k} \ .$$

This implies that repeating PPZ-Modify $O^*\left(2^{n(1-1/k)}\right)$ times is enough to output a satisfying assignment to $\mathbf{F}$ with probability $1 - o(1)$, if one exists.

3.2 The separator lemma

We first generalize Lemma 16 to lower bound $\tau(\mathbf{F}, A)$ for arbitrary sets A of satisfying assignments.

Lemma 17 (Separator Lemma)**.** *Let $A \subseteq \Omega_{\mathbf{F}}$, let S be the set of edges of $G_{\mathbf{F}}$ with one endpoint in A and the other endpoint in $\Omega_{\mathbf{F}} \backslash A$. Further, let $E(A)$ be the edges of $G_{\mathbf{F}}$ with both endpoints in A. Then,*

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

$$\geqslant 2^{-n(1-1/k)} |A|^{1-1/k} 2^{-\frac{|S|}{k|A|}} \tag{2}$$

Proof. TOPROVE 2 ☐

3.3 Geometric sampling properties of PPZ-Modify

In this section, we prove the dispersion properties of the PPZ-Modify subroutine. The goal is to show that PPZ-Modify is acts like an approximate farthest oracle: if a satisfying assignment exists that is "far away" from a set of already chosen solutions, then PPZ-Modify will output an approximately "far away" satisfying assignment with good probability.

In particular, let $z_0 \in \{0,1\}^n$ be any (not necessarily satisfying) assignment to $\mathbf{F}$. Let r denote the maximum distance from z_0 to any satisfying assignment in $\Omega_{\mathbf{F}}$. We show that PPZ-Modify will output, with probability at least $n^{-O(1)} \cdot 2^{-n+n/k}$, a satisfying assignment z such that $d_H(z, z_0) \geqslant$

$\left(1 - \frac{1}{k}\right) r$. As a corollary, this implies that for any satisfying assignment z, PPZ-Modify outputs a satisfying assignment to $\mathbf{F}$ within distance n/k of z with probability at least $n^{-O(1)} \cdot 2^{-n+n/k}$. Formally, we show that:

Lemma 18. *Let $\mathbf{F}$ be a satisfiable k-CNF formula, $z_0 \in \{0,1\}^n$, and $r = \max_{z \in \Omega_{\mathbf{F}}} d_H(z, z_0)$. Let y and π be chosen uniformly at random and independently from $\{0,1\}^n$ and $\mathcal{S}_n$ respectively. The probability that PPZ-Modify$(\mathbf{F}, y, \pi)$ outputs $z^* \in \Omega_{\mathbf{F}}$ with $d_H(z^*, z_0) \geqslant (1 - 1/k) \cdot r$ is at least $\frac{1}{2n} \cdot 2^{-n+n/k}$*

Proof. TOPROVE 3 □

The above lemma proves geometric sampling properties of PPZ for Diam and Opt-min. Now we consider Opt-sum: there exists a multi-set of assignments T, and our goal is to find a satisfying assignment z^* that maximises the sum of distances from the assignments in T, denoted as sum-$d_H(z^*, T)$. We show that with probability at least $\frac{1}{2n} \cdot 2^{-n+n/k}$, PPZ-Modify outputs such a satisfying assignment, with an approximation factor of $\left(1 - \frac{2}{k+1}\right)$. We employ the same strategy as in the proof of Lemma 18, dividing the vertex set of $G_{\mathbf{F}}$ into levels based on sum-$d_H(\cdot, T)$. However, in this case, we can no longer argue that a vertex $z \in V_{i*}$ neighbors in only V_{i*-1} and V_{i*+1}. This is because changing one coordinate in z does not necessarily decrease the objective function sum-$d_H(z, T)$ by just one. Hence, bounding the size of the separator S_{i*}, where S_{i*} is the set of edges between U_{i*} and $G_{\mathbf{F}} \backslash U_{i*}$ is more involved.

Lemma 19. *Let $\mathbf{F}$ be a satisfiable k-CNF formula, $T \subseteq \{0,1\}^n$ be a multiset of size t, and $r_{sum} = \max_{z \in \Omega_{\mathbf{F}}}$ sum-$d_H(z, T)$. Let y and π be chosen uniformly at random from $\{0,1\}^n$ and $\mathcal{S}_n$ respectively. The probability that PPZ-Modify$(\mathbf{F}, y, \pi)$ outputs $z^* \in \Omega_{\mathbf{F}}$ with sum-$d_H(z^*, T) \geqslant \frac{k-1}{k+1} \cdot r_{sum}$ is at least $\frac{1}{2n} \cdot 2^{-n+n/k}$.*

Proof. TOPROVE 4 □

3.4 Algorithmic Implications: farthest point oracles

We now use these dispersion properties to define farthest point oracles for the Diam, minPD and sumPD problems. To begin with, we show that we can use the PPZ algorithm to design an *approximate farthest point oracle.* An approximate farthest point oracle takes as input a k-CNF formula $\mathbf{F}$, an assignment z, and outputs a satisfying assignment z^* that is approximately the farthest satisfying assignment for $\mathbf{F}$ from z.

Lemma 20. *Let $\mathbf{F}$ be a k-CNF formula over n variables and $n^{O(1)}$ clauses and $z \in \{0,1\}^n$ be any assignment to $\mathbf{F}$. If $\mathbf{F}$ is satisfiable, there exists an algorithm that in time $O^*(2^{n-n/k})$ that outputs $z^* \in \Omega_{\mathbf{F}}$, with $d_H(z, z^*) \geqslant \left(1 - \frac{1}{k}\right) \max_{z' \in \Omega_{\mathbf{F}}} d_H(z, z')$ with probability at least $1 - 2^{-2n}$.*

Proof. TOPROVE 5 □

Next, we can define a farthest point oracle for sum-d_H.

Lemma 21. *Let $\mathbf{F}$ be a k-CNF formula over n variables and $n^{O(1)}$ clauses and $S \subseteq \{0,1\}^n$ be a multiset of size s. There exists an algorithm running in time $O^*(s \cdot 2^{n-n/k})$ that, if $\mathbf{F}$ is satisfiable, outputs $z^* \in \Omega_{\mathbf{F}}$, with sum-$d_H(S, z^*) \geqslant \left(\frac{k-1}{k+1}\right) \max_{z \in \Omega_{\mathbf{F}}}$ sum-$d_H(z, S)$ with probability $1 - 2^{-2n}$.*

Proof. TOPROVE 6 □

Next, we give a farthest point oracle for MIN-d_H.

Lemma 22. *Let* $\mathbf{F}$ *be a* k*-CNF formula over* n *variables and* $n^{O(1)}$ *clauses and* $S \subseteq \{0,1\}^n$ *be a set of size* s*. There exists an algorithm running in time* $O^*(s^2 \cdot 2^{n-n/k})$ *that, if* $\mathbf{F}$ *is satisfiable, there* PPZ-Farthest-Min$(\mathbf{F}, s)$ *outputs* $z^* \in \Omega_{\mathbf{F}}$*, with* MIN-$d_H(S, z^*) \geqslant \left(1 - \frac{1}{kH^{-1}(1-1/k)}\right) \max_{z \in \Omega_{\mathbf{F}}}$ MIN-$d_H(z, S)$ *with probability at least* $1 - 2^{-2n}$.

Proof. TOPROVE 7 □

3.5 PPZ-based algorithms for dispersion: Proofs of Theorem 4, Theorem 15 and Theorem 7

Proof of Theorem 4

Lemma 20 implies that the algorithm PPZ-Farthest behaves like a $(1 - 1/k)$-approximate farthest point oracle for k-SAT that runs in time $O^*(2^{n-n/k})$. That is, it takes as input a k-CNF formula $\mathbf{F}$ and $z \in \{0,1\}^n$, and with probability $1 - 2^{-2n}$, outputs $z^* \in \Omega_{\mathbf{F}}$ such that $d_H(z, z^*) \geqslant (1 - 1/k) \cdot \max_{z' \in \Omega_{\mathbf{F}}} d_H(z, z')$. Hence, we can use the following procedure to output a $\frac{1}{2}(1 - 1/k)$ approximation to $\mathbf{F}$: Use the PPZ algorithm to find one satisfying assignment z_1^* to $\mathbf{F}$, and then output $z_2^* =$ PPZ-Farthest$(\mathbf{F}, z_1^*)$. The triangle inequality then implies that z_1^* and z_2^*, will satisfy $d_H(z_1^*, z_2^*) \geqslant \frac{1}{2}(1 - 1/k) \cdot \textsc{Diam}(\mathbf{F})$.

Proof of Theorem 15

Lemma 23. *Suppose there exists a* $1 - \delta$*-approximate farthest point oracle,* $\mathcal{O}$ *that takes a* k*-CNF formula* $\mathbf{F}$ *and a multi-set* $S \subseteq \{0,1\}^n$ *and with probability* $1 - 2^{-2n}$*, outputs* $z^* \in \Omega_{\mathbf{F}}$ *such that* SUM-$d_H(S, z^*) \geqslant (1 - \delta) \cdot \max_{z' \in \Omega_{\mathbf{F}}}$ SUM-$d_H(S, z')$*. Then, there exists an algorithm taking* $\mathbf{F}$ *and* s *as input that uses* $s^3 n$ *calls to* $\mathcal{O}$ *(and an additional* $s^4 n^{O(1)}$ *overhead) that outputs a multi-set* $S^* \subseteq \Omega_{\mathbf{F}}$ *with* SUMPD$(S^*) \geqslant \max\{\frac{1}{2}(1 - \delta), \frac{(1-\delta)(s-1)}{(1+\delta)s + (1-\delta)}\} \cdot$ OPT-SUM$(\mathbf{F}, s)$ *with probability* $1 - o(1)$.

Proof. TOPROVE 8 □

We note that Lemma 21 implies that the algorithm PPZ-Farthest-Sum is a $1 - \delta$ approximate farthest point oracle, as defined in Lemma 23, for $\delta = \frac{2}{k+1}$. Hence, we can use PPZ-Farthest-Sum as a black box in the algorithm defined by Lemma 23. This completes the proof of Theorem 15.

Proof of Theorem 7

Lemma 24. *Suppose there exists a* $1 - \delta$*-approximate farthest point oracle,* $\mathcal{O}$ *that takes a* k*-CNF formula* $\mathbf{F}$ *and a set* $S \subseteq \{0,1\}^n$ *as input and with probability* $1 - 2^{-2n}$*, outputs* $z^* \in \Omega_{\mathbf{F}}$ *such that* MIN-$d_H(S, z^*) \geqslant (1 - \delta) \cdot \max_{z' \in \Omega_{\mathbf{F}}}$ MIN-$d_H(S, z')$*. Then, there exists an algorithm taking* $\mathbf{F}$ *and* s *as input that uses* s *calls to* $\mathcal{O}$ *(and an additional* $sn^{O(1)}$ *overhead) that outputs a set* $S^* \subseteq \Omega_{\mathbf{F}}$ *with* MINPD$(S^*) \geqslant \frac{1}{2}(1 - \delta) \cdot$ OPT-MIN$(\mathbf{F}, s)$ *with probability* $1 - o(1)$.

Proof. TOPROVE 9 □

We note that Lemma 22 implies that the algorithm PPZ-Farthest-Min is a $1-\delta$ approximate farthest point oracle as defined in Lemma 24, for $\delta = \frac{1}{kH^{-1}(1-1/k)}$. Hence, we can use PPZ-Farthest-Min as a black box in the algorithm defined by Lemma 24. This completes the proof of Theorem 7.

4 From approximate local search to dispersion – Schöning's algorithm

In this section we prove a generalization of Theorem 5 and we state and prove theorems with the same running time guarantees (up to a factor polynomial in s) to approximate $\textsc{Opt-min}(\mathbf{F}, s)$, $\textsc{Opt-min}(\mathbf{F}, s, \geqslant W)$, $\textsc{Opt-min}(\mathbf{F}, s, \leqslant W)$ as well as $\textsc{Opt-sum}(\mathbf{F}, s)$. We note that the algorithm for $\textsc{Opt-min}(\mathbf{F}, s)$ follows as special cases of the algorithms for $\textsc{Opt-min}(\mathbf{F}, s, \geqslant W)$, $\textsc{Opt-min}(\mathbf{F}, s, \leqslant W)$.

To start with, we define the quantity $\tau(\delta, k, n)$ to be $\frac{2^n (k-1)^R}{\binom{n}{R}}$ where $R = \left\lfloor \frac{\delta n}{2\left(2+\delta+\frac{2}{k-2}\right)} \right\rfloor$, for each $\delta \in \left(0, \min\left\{1, \frac{4(k-1)}{(k-2)^2}\right\}\right]$. From now on, we assume that $k \geqslant 3$ unless stated otherwise.

Theorem 25 (Schöning for $\textsc{Diam}$: Generalization of Theorem 5)**.** *Let $\mathbf{F}$ be a k-CNF formula on n variables. For each $0 < \delta \leqslant \min\{1, \frac{4(k-1)}{(k-2)^2}\}$, there exists an algorithm taking $\mathbf{F}$ as input and running in time $O^*(\tau(\delta, k, n))$ that outputs $z_1^*, z_2^* \in \Omega_{\mathbf{F}}$ such that $d_H(z_1^*, z_2^*) \geqslant \frac{1}{2} \cdot (1-\delta)\, \textsc{Diam}(\mathbf{F})$, if $\mathbf{F}$ is satisfiable.*

To make the above result more concrete, we first observe that we can define $a_{k,\delta}$, such that $\tau(\delta, k, n) = O^*(a_{k,\delta}^n)$. Now, for $k = 7$ and $k = 4$, we plot $a_{k,\delta}$ as a function of δ and compare it with what the PPZ algorithm achieves. Hence, this algorithm provides a smooth trade-off between the approximation factor (i.e., $(1-\delta)$) and running time. We note that for $k = 7$, we can achieve the Schöning running time for a non-trivial value of δ, but for $k = 4$, we cannot do so, even for δ very close to 1. Note that this algorithm still achieves non-trivial savings over a brute force search for all values of δ, and in particular, it can be faster than the PPZ algorithm (albeit with a worse approximation factor).

Remark 7. When $k \geqslant 7$, we can use $\delta = \frac{4(k-1)}{(k-2)^2}$ in this algorithm to get a running time of $O^*\left(\left(2 - \frac{2}{k}\right)^n\right)$ that matches the run-time of Schöning's algorithm for finding one satisfying assignment. Thus, Theorem 25 is a generalization of Theorem 5. For smaller k, while we cannot match the running time of Schöning's algorithm, we can still get better than brute force algorithms for the diameter and dispersion problems.

Weighted dispersion: For a k-CNF formula $\mathbf{F}$, let $\Omega_{\mathbf{F},=W}$,$\Omega_{\mathbf{F},\geqslant W}$, $\Omega_{\mathbf{F},\leqslant W}$ denote the set of satisfying assignments to $\mathbf{F}$ with Hamming weight W, at least W and at most W respectively. Let
$\textsc{Opt-min}(\mathbf{F}, s, \geqslant W) = \max_{S \subseteq \Omega_{\mathbf{F},\geqslant W}, |S|=s} \textsc{min}\mathrm{PD}(S)$, and
$\textsc{Opt-min}(\mathbf{F}, s, \leqslant W) = \max_{S \subseteq \Omega_{\mathbf{F},\leqslant W}, |S|=s} \textsc{min}\mathrm{PD}(S)$.

Theorem 26 (Weighted dispersion- Full version of Theorem 8)**.** *Let $\mathbf{F}$ be a k-CNF formula on n variables, $W \in [n]$ and $s \in \mathbb{N}$.*

1. *For each $0 < \delta \leqslant \min\left\{1, \frac{4(k-1)}{(k-2)^2}\right\}$, there exists an algorithm that takes $\mathbf{F}, W, s$ as input and runs in time $O^*\left(s^3 \cdot \tau(\delta, k, n)\right)$ and outputs a set $S^* \subseteq \Omega_{\mathbf{F},\geqslant(1-\delta)W}$ of size s such that $\textsc{min}PD(S^*) \geqslant \frac{1}{2}(1-\delta)\, \textsc{Opt-min}(\mathbf{F}, s, \geqslant W)$ with probability $1 - o(1)$.*

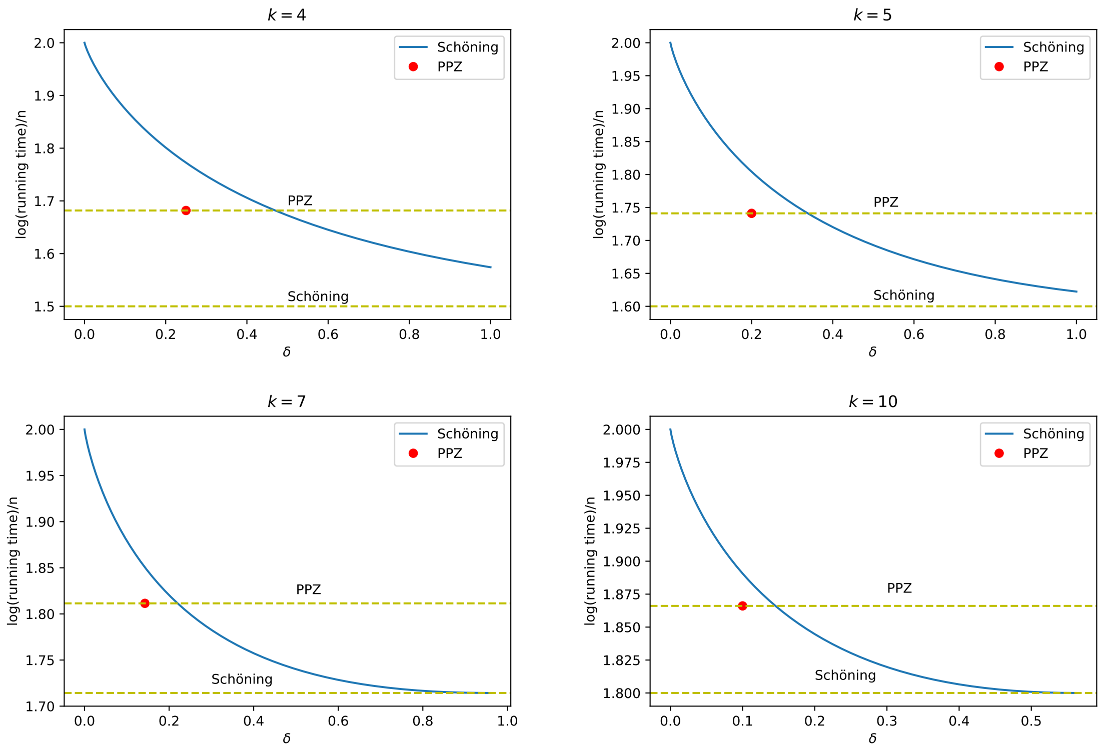

Figure 1: Plot of $a_{k,\delta}$ with respect to δ, with the PPZ running time and approximation factor and Schöning running time for comparison, for different values of k.

2. *For each* $0 < \delta \leqslant \min\left\{1, \frac{4(k-1)}{(k-2)^2}\right\}$*, there exists an algorithm that takes* $\mathbf{F}, W, s$ *as input and runs in time* $O^*\left(s^3 \cdot \tau(\delta, k, n)\right)$ *and outputs a set* $S^* \subseteq \Omega_{\mathbf{F}, \geqslant (1+\delta)W}$ *of size* s *such that* $\textsc{min}PD(S^*) \geqslant \frac{1}{2}(1-\delta)\,\textsc{Opt-min}(\mathbf{F}, s, \geqslant W)$ *with probability* $1 - o(1)$*.*

Note that as a special case, this theorem leads to an algorithm for $\textsc{Opt-min}(\mathbf{F}, s)$ with the same time bounds and approximation factors. In addition, we show that a slight modification of this algorithm can also be used for $\textsc{Opt-sum}(\mathbf{F}, s)$.

Theorem 27. *[Schöning approximating* $\textsc{Opt-sum}(\mathbf{F}, s)$*] Let* $\mathbf{F}$ *be a* k*-CNF formula on* n *variables and* $s \in \mathbb{N}$*. For each* $0 < \delta \leqslant \min\left\{1, \frac{4(k-1)}{(k-2)^2}\right\}$*, there exists an algorithm that takes* $\mathbf{F}, s$ *as input and runs in time* $O^*\left(s^3 \cdot \tau(\delta, k, n)\right)$ *that outputs, with probability* $1 - o(1)$*, a multi-set* $S^* \subseteq \Omega_{\mathbf{F}, \geqslant (1-\delta)W}$ *of size* s *such that*

$$\textsc{sum}PD(S^*) \geqslant \begin{cases} \frac{1}{2}(1-\delta)\,\textsc{Opt-sum}(\mathbf{F}, s) & \text{if } s \leqslant 3 + \left\lfloor \frac{2\delta}{1-\delta} \right\rfloor \\ \frac{1-\delta}{1+\delta}\left(\frac{1-\frac{1}{s}}{1+\frac{1-\delta}{1+\delta}\cdot\frac{1}{s}}\right)\textsc{Opt-sum}(\mathbf{F}, s) & \text{if } s > 3 + \left\lfloor \frac{2\delta}{1-\delta} \right\rfloor \end{cases}$$

The case of 2-SAT and other small k: We design different algorithms to handle the case of 2-SAT, which also outperform the algorithms presented here in some regimes of δ for larger k. For example, for $k = 3$, it outperforms the algorithm in Theorem 25 for all values of δ, and for $k \geqslant 4$, it outperforms Theorem 25 for smaller values of δ. This is presented in Appendix A.1.

Proof organization: We prove the above three theorems in parallel using the following three step procedure.

1. In Section 4.1, we recall Schöning's algorithm and the key observations used to analyse it.
2. In Section 4.2, we develop and analyze farthest point oracles for DIAM, SUM-d_H and MIN-d_H using Schöning's algorithm.
3. In Section 4.3, we describe and analyse our algorithms for finding dispersed solutions with respect to OPT-MIN, completing the proofs of Theorem 25 and Theorem 26. Just like for PPZ, these algorithms use the farthest point oracles in the algorithms for dispersion studied by Gonzales [Gon85]. In Appendix A.2, we describe and analyse an algorithm for finding dispersed solutions with respect to OPT-SUM, completing the proof of Theorem 27.
4. In Appendix A.1, we describe another algorithm that handles the case of 2-SAT and 3-SAT and also outperforms the algorithms described in this section for some regimes of δ for larger values of k.

4.1 Parameterized local search

The Schöning walk. Schöning's algorithm consists of repeatedly invoking the following procedure, which we call a *Schöning walk*. Formally, a Schöning walk of length one, denoted by $\mathsf{SW}_1(\mathbf{F}, z)$, takes as input a formula $\mathbf{F}$ and an assignment $z \in \{0,1\}^n$, and returns another assignment $z' \in \{0,1\}^n$ constructed as follows: if z is a satisfying assignment, then $z' = z$. Otherwise, let C be a clause in $\mathbf{F}$ that is not satisfied by z. Pick one of its k literals uniformly at random and flip its value in z, thus obtaining z'. For $t \geqslant 2$, a Schöning walk of length t can be recursively defined as $\mathsf{SW}_t(\mathbf{F}, z) = \mathsf{SW}_1(\mathbf{F}, \mathsf{SW}_{t-1}(\mathbf{F}, z))$. We refer to z as the starting point of the Schöning walk of length t.

We note the following key observation about the Schöning walk. We refer the reader to Schöning's original paper for a proof [Sch99].

Lemma 28. *For any starting assignment $z \in \{0,1\}^n$, if there exists a satisfying assignment $z^* \in \{0,1\}^n$ such that $d_H(z, z^*) \leqslant t$, then $\mathsf{SW}_t(\mathbf{F}, z)$ outputs a satisfying assignment with probability at least k^{-t}. Furthermore, $\mathsf{SW}_{\lceil(1+2/(k-2))t\rceil}(\mathbf{F}, z)$ outputs a satisfying assignment with probability at least $(k-1)^{-t}$.*

Remark 8. In Schöning's original paper, the statement proved is that $\mathsf{SW}_{3t}(\mathbf{F}, z)$ outputs a satisfying assignment with probability at least $(k-1)^{-t}$. However, looking at the analysis more carefully, we can prove that a shorter Schöning walk of length $(1 + 2/(k-2))\,t$ suffices (for $k = 3$, these two quantities are equal). This fact is irrelevant to the performance of the original algorithm, but is helpful for our purpose of finding dispersed satisfying assignments to $\mathbf{F}$.

Schöning's local search: Lemma 28 gives a *parameterized local search* algorithm for k-SAT. Formally for some values $\alpha \geqslant 1, c > 1$, a local search procedure $\mathsf{LS}_{\alpha,c}$ takes as input a k-CNF formula $\mathbf{F}$, a starting assignment $z \in \{0,1\}^n$, and $t \in [n]$, such that if there exists a satisfying assignment z_0, with $d_H(z, z_0) \leqslant t$, then, in time $n^{O(1)}c^t$, $\mathsf{LS}_{\alpha,c}$ outputs a satisfying assignment $z^* \in \Omega_{\mathbf{F}}$, with $d_H(z, z^*) \leqslant \lceil \alpha t \rceil$. [11]. Hence, there exist two versions of parameterized local search for k-SAT.

1. $\mathsf{LS}_{1,k}$: This involves repeating the Schöning walk starting at z for t steps $n^{O(1)} \cdot k^t$ times.
2. $\mathsf{LS}_{(1+2/(k-2)),k-1}$: This involves repeating the Schöning walk of $\lceil (1 + 2/(k-2))\, t \rceil$ steps starting at z $n^{O(1)} \cdot (k-1)^t$ times.

Consider the following algorithm for solving k-SAT. Given a local search procedure $\mathsf{LS}_{\alpha,c}$, set $t = \left\lfloor \frac{n}{c+1} \right\rfloor$, sample $z \in \{0,1\}^n$ uniformly at random, and run $\mathsf{LS}_{\alpha,c}$ with z and t as input. If there exists a satisfying assignment z_0, z will be within distance t of z_0 with probability at least $\frac{\binom{n}{t}}{2^n}$. To succeed in finding a satisfying assignment with probability $1 - o(1)$, it is sufficient to repeat this procedure $n^{O(1)} \cdot \frac{2^n}{\binom{n}{t}}$ times. The entire algorithm runs in time $n^{O(1)} \cdot \frac{2^n}{\binom{n}{t}c^{-t}} = O^*\left(\left(\frac{2}{1+1/c}\right)^n\right)$. Schöning uses $\mathsf{LS}_{(1+2/(k-1))t,k-1}$, which gives a running time of $O^*\left(\left(2\left(1 - \frac{1}{k}\right)\right)^n\right)$. We refer the reader to Appendix F for a proof of this statement.

The case of 2-SAT and other small k: Our algorithms for approximating dispersion use the procedure $\mathsf{LS}_{(1+2/(k-2)),k-1}$. For the case of small k and small δ, it is useful to use the local search procedure $\mathsf{LS}_{1,k}$ instead. It turns out that this algorithm gives a better trade-off with δ. We present more details in Appendix A.1.[12]

4.2 Anchored local search and farthest point oracles

Next, we show that we can carefully control the length of the Schöning walk to come up with farthest point oracles. We call this procedure "anchoring". This technique is general and can be used with any $\mathsf{LS}_{\alpha,c}$ procedure for a "subset problem". We will see more examples in Section 5.

Lemma 29. *Consider a local search algorithm $\mathsf{LS}_{\alpha,c}$. Then, for every $0 < \delta \leqslant \frac{2(1+\alpha)}{c-1}$, there exists an algorithm running in time $\frac{2^n c^R}{\binom{n}{R}}$, where $R = \left\lfloor \frac{\delta n}{2(1+\alpha+\delta)} \right\rfloor$, that takes as input $\mathbf{F}$ and $z \in \{0,1\}^n$, and if $\mathbf{F}$ is satisfiable, outputs $z^* \in \Omega_{\mathbf{F}}$ such that $d_H(z^*, z) \geqslant (1-\delta) \cdot \max_{z' \in \Omega_{\mathbf{F}}} d_H(z, z')$ with probability at least $1 - 2^{-n}$.*

Proof. TOPROVE 10 □

[11] We have defined an "approximate" version of local search. The traditional definition does not use α

[12] Before Schöning's algorithm for k-SAT was discovered, a very similar (polynomial time) algorithm was developed for 2-SAT by Papadimitriou [Pap91]. It picks a starting assignment $z \in \{0,1\}^n$ at random and performs a Schöning walk for $O(n^2)$ steps starting at z. If the 2-CNF formula is indeed satisfiabile, this algorithm finds a satisfying assignment with probability $1 - o(1)$. Schöning's main innovation in extending this algorithm to get a better than brute force algorithm was in restarting the local search process with a new randomly chosen starting assignment after $3n$ steps. However, computing the diameter of a 2-CNF formula is an NP-complete problem, and Schöning's paradigm is useful here as well.

Lemma 30. *Let* $0 < \beta \leqslant \frac{2}{c-1}$*. Then,*

$$\max_{r \in \{0,1,\ldots,\left\lfloor \frac{n}{2(1+\beta)} \right\rfloor\}} \frac{\binom{n}{r+\lfloor \beta r \rfloor} c^{\lfloor \beta r \rfloor}}{\binom{n}{\lfloor \beta r \rfloor}} \leqslant n^{O(1)} \cdot \frac{2^n}{\binom{n}{\left\lfloor \frac{\beta n}{2(1+1\beta)} \right\rfloor} c^{-\left\lfloor \frac{\beta n}{2+2\beta} \right\rfloor}}$$

In order to prove Lemma 30, we need some observations.

Observation 31. For integers n and $m \leqslant n/2$, $\frac{2}{n} \cdot \binom{n}{m+1} \leqslant \binom{n}{m} \leqslant \binom{n}{m+1}$.

Observation 32 ([MS77])**.**

$$\frac{1}{n^{O(1)}} \cdot \left(\mu^{-\mu}(1-\mu)^{\mu-1}\right)^n \leqslant \binom{n}{\mu n} \leqslant \left(\mu^{-\mu}(1-\mu)^{\mu-1}\right)^n$$

Observation 33. The derivative of the function $f(\mu) = \mu^{-\mu}(1-\mu)^{\mu-1}$ with respect to μ is $f'(\mu) = f(\mu)\left(\ln(1-\mu) - \ln \mu\right)$.

Proof of Lemma 30 Let $r = \mu n$. let $f(\mu) = \mu^{-\mu}(1-\mu)^{\mu-1}$ We use Observation 32 to show that

$$\frac{\binom{n}{r+\lfloor \beta r \rfloor}}{\binom{n}{\lfloor \beta r \rfloor} c^{-\lfloor \beta r \rfloor}} = n^{O(1)} \cdot (g(\mu))^n \ ,$$

where $g(\mu) = \frac{f((1+\beta)\mu))}{c^{-\beta\mu} f(\beta\mu)}$. We next show that g is an increasing function of μ, which means that the maximum value of $g(\mu)$ is obtained at $\mu = \frac{1}{2(1+\beta)}$. Using the quotient, product and chain rules for differentiation and Observation 33, we can show that

$$g'(\mu) = \frac{\beta \ln(c) c^{\beta\mu} f(\beta\mu) f((1+\beta)\mu) + (1+\beta) c^{-\beta\mu} f(\beta\mu) f'((1+\beta)\mu) - \beta c^{\beta\mu} f'(\beta\mu) f((1+\beta)\mu)}{f(\beta\mu)^2}$$

$$= \frac{\begin{array}{r}\beta \ln(c) c^{\beta\mu} f(\beta\mu) f((1+\beta)\mu) - \beta \ln\left(\frac{1-\beta\mu}{\beta\mu}\right) c^{\beta\mu} f(\beta\mu) f((1+\beta)\mu) \\ + (1+\beta)\left(\ln\left(\frac{1-(1+\beta)\mu}{(1+\beta)\mu}\right)\right) c^{-\beta\mu} f(\beta\mu) f\left((1+\beta)\mu\right)\end{array}}{f(\beta\mu)^2}$$

$$= g(\mu)\left(\beta \ln(c) - \beta \ln\left(\frac{1-\beta\mu}{\beta\mu}\right) + (1+\beta)\ln\left(\frac{1-(1+\beta)\mu}{(1+\beta)\mu}\right)\right)$$

Let $h(\mu) = \frac{g'(\mu)}{g(\mu)}$. If we show that the $h(\mu)$ is a decreasing function of μ, when $0 \leqslant \mu \leqslant \frac{1}{2(1+\beta)}$, that is enough to show that $h(\mu) \geqslant 0$ for all $0 \leqslant \mu \leqslant \frac{1}{2(1+\beta)}$. We now compute $h'(\mu)$.

$$h'(\mu) = \frac{\beta}{(1-\beta\mu)\mu} - \frac{1+\beta}{(1-(1+\beta)\mu)\mu} \ ,$$

which is negative for all $0 < \mu \leqslant \frac{1}{2(1+\beta)}$. Hence, $g\left(\frac{1}{2(1+\beta)}\right)$ is an upper bound for all $g(\mu)$ for $0 \leqslant \mu \leqslant \frac{1}{2(1+\beta)}$.

We now generalize this approach to come up with a farthest point oracle for the MIN-d_H dispersion measure.

Heavy and low weight dispersion: We now show that this approach can also be used to return dispersed satisfying assignments of large or small Hamming weight. For a k-CNF formula $\mathbf{F}$, recall that $\Omega_{\mathbf{F},=W}$,$\Omega_{\mathbf{F},\geqslant W}$, $\Omega_{\mathbf{F},\leqslant W}$ denote the set of satisfying assignments to $\mathbf{F}$ with Hamming weight W, at least W and at most W respectively. Let $\textsc{Opt-min}(\mathbf{F}, s, \geqslant W) = \max_{S\subseteq\Omega_{\mathbf{F},\geqslant W},|S|=s} \textsc{min}\mathrm{PD}(S)$, and $\textsc{Opt-min}(\mathbf{F}, s, \leqslant W) = \max_{S\subseteq\Omega_{\mathbf{F},\leqslant W},|S|=s} \textsc{min}\mathrm{PD}(S)$.

Lemma 34 (Farthest Point Oracle)**.** *Consider a local search algorithm $\mathsf{LS}_{\alpha,c}$. Then, for every $0 < \delta \leqslant \frac{2(1+\alpha)}{c-1}$, there exists an algorithm that takes as input a k-CNF formula $\mathbf{F}$, a set $S \subseteq \{0,1\}^n$ of size s and $W \in [n]$. If $\Omega_{\mathbf{F},=W}$ is non-empty, with probability at least $1 - 2^{-2n}$, it outputs $z^* \in \Omega_{\mathbf{F}}$ such that $(1-\delta)W \leqslant |z^*| \leqslant (1+\delta)W$ and $\textsc{min}\text{-}d_H(z^*, S) \geqslant \max_{z'\in\Omega_{\mathbf{F},=W}} \textsc{min}\text{-}d_H(z', S)$. The algorithm runs in time $s^2 \cdot n^{O(1)} \cdot \frac{2^n c^R}{\binom{n}{R}}$, where $R = \left\lfloor \frac{\delta n}{2(1+\alpha+\delta)} \right\rfloor$.*

Proof. TOPROVE 11 □

4.3 Schöning-based algorithms for dispersion: Proofs of Theorem 25 and Theorem 26

Proof of Theorem 25: Diameter

The proof of Theorem 25 is similar to that of Theorem 4. Lemma 29 implies that there exists a $1-\delta$-approximate farthest point oracle that takes as input a k-CNF formula $\mathbf{F}$ and $z \in \{0,1\}^n$, and with probability $1 - 2^{-2n}$, outputs $z^* \in \Omega_{\mathbf{F}}$ such that $d_H(z, z^*) \geqslant (1-\delta)\max_{z'\in\Omega_{\mathbf{F}}} d_H(z, z')$. We first use Schöning's algorithm for k-SAT to find one satisfying assignment z_1^* to $\mathbf{F}$. Let z_2^* be the satisfying assignment output by the $1-\delta$-approximate farthest point oracle with z_1^* and $\mathbf{F}$ as input. The triangle inequality then implies that z_1^* and z_2^*, will satisfy $d_H(z_1^*, z_2^*) \geqslant \frac{1}{2}(1-\delta)$. The running time guarantee for the first and second algorithms come from using $c = k-1, \alpha = 1 + \frac{2}{k-2}$ that we described in Section 4.1.

Proof of Theorem 26: Weighted min-dispersion

Firstly, it is easy to observe that for any set $S \subseteq \{0,1\}^n$ and $W \in [n]$, we can use the algorithm in Lemma 34 to output $z^* \in \Omega_{\mathbf{F},\geqslant(1-\delta)W}$ such that $\textsc{min}\text{-}d_H(z^*, S) \geqslant (1-\delta)\max_{z\in\Omega_{\mathbf{F},\geqslant W}} \textsc{min}\text{-}d_H(z, S)$. We do so by iterating over all $W' \in \{W, W+1, \ldots, n\}$, using $\mathsf{Schöning\text{-}Farthest\text{-}Weighted}(\mathbf{F}, S, W')$, and returning z^* with maximum value of $\textsc{min}\text{-}d_H(z^*, S)$. This can be used along with Lemma 24 to prove Theorem 26.

Proof of Theorem 27: Sum-dispersion

We refer the reader to Appendix A.2 for the proof.

5 Applications and generalisations

In Sections 5.1 and 5.2 we demonstrate that the techniques we developed in Section 4 are fairly general and can be also used to obtain diverse solutions to several NP-complete optimisation problems. Following this, Section 5.3 shows how an improvement in runtime of Schöning's and PPZ algorithms (for finding one solution) can be obtained if $\Omega_{\mathbf{F}}$ has many dispersed solutions. Finally, Section 5.4 shows how to extend our Schöning result to finding diverse solutions to CSPs.

For simplicity, we focus on the Opt-min diversity measure in this section. It is easy to generalize the results to the Opt-sum diversity measure as well.

Optimization Problems and Bi-Approximations

We show that our techniques can be used in a black-box as well as white-box manner for a broad class of optimization problems called *subset problems*. A subset problem consists of an implicitly defined family $\mathcal{F}$ of subsets of $[n]$, and the problem is to find $A \in \mathcal{F}$ of minimum (or maximum) size. We start with describing a framework that captures all these problems. This framework will also help us to abstract the notion of *an isometric reduction*, which we will define formally in Section 5.1.

Implicit set systems: An implicit set system Φ is a function that takes a string $I \in \{0,1\}^*$ (called the instance) and outputs an integer $n \in \mathbb{N}$ and $\mathcal{F}_I \subseteq \{0,1\}^n$, called the feasible set of Φ. Elements in $\{0,1\}^n$ outside $\mathcal{F}_I$ are called infeasible. Many natural computational problems we consider can be defined using implicit set systems. For an implicit set system Φ, we define the computational problem Φ-Subset.

Problem 1 (Φ-Subset)**. Input:** An instance $I \in \{0,1\}^*$ to Φ.
Output: $A \in \mathcal{F}_I$, if $\mathcal{F}_I$ is non-empty.

An example of an implicit set system is one generated by k-CNF formulas. If the input instance I is a k-CNF formula $\mathbf{F}$ over n variables (using some canonical encoding of formulas as strings), then $\Phi(\mathbf{F}) = (n, \Omega_{\mathbf{F}})$ ($\mathcal{F}_I$ is defined to be empty for all other strings for consistency). In this case, the problem Φ-Subset is $\mathcal{NP}$-complete. Other examples of implicit set systems are those generated by graphs, where the input string I encodes a graph G, $n_I = |V(G)|$, and $\mathcal{F}_I$ is the set of all independent sets of G, or the set of all vertex covers of G, etc. For such problems, sets are identified with the corresponding bit-vectors. Throughout this section, we will interchangeably use strings in $\{0,1\}^n$ to denote subsets of $[n]$ and vice versa.

For the graph problems posed above, the problem Φ-Subset is in $\mathcal{P}$, and for an instance I we are interested in finding the element (set) in $\mathcal{F}_I$ that has minimum (or maximum) weight (size).

Problem 2 (Φ-Min)**. Input:** An instance $I \in \{0,1\}^*$ to Φ.
Output: $A \in \mathcal{F}_I$, of minimum weight if $\mathcal{F}_I$ is non-empty.

Problem 3 (Φ-Max)**. Input:** An instance $I \in \{0,1\}^*$ to Φ.
Output: $A \in \mathcal{F}_I$, of maximum weight if $\mathcal{F}_I$ is non-empty.

An example of Φ-Min is Minimum Vertex Cover and an example of Φ-Max is Maximum Independent Set. For an instance I for these problems, we use $\mathsf{OPT}_{\Phi\text{-Max}}(I)$ and $\mathsf{OPT}_{\Phi\text{-Min}}(I)$ to denote the size of the sizes of the largest and smallest sets in $\mathcal{F}_I$ respectively (if $\mathcal{F}_I$ is non-empty). We also use $\mathcal{F}_{I,\min}$ and $\mathcal{F}_{I,\max}$ to denote the subsets of $\mathcal{F}_I$ consisting of the elements of smallest and largest weight respectively.

Now, we are interested in finding approximately maximally diverse solutions to the Φ-Min and Φ-Max problems, that are also approximately optimal. In the following definition of bi-approximation, let $C_1 \geqslant 1$ and $C_2 \leqslant 1$.

Problem 4 ((C_1, C_2)-Diverse-Φ-Min)**. Input:** An instance I to the implicit set system Φ, $s \in \mathbb{N}$
Output: $S^* \subseteq \mathcal{F}_I$ with s elements such that every $z \in S^*$ has weight at most $C_1 \cdot \mathsf{OPT}_{\Phi\text{-Min}}(I)$, and $\text{minPD}(S^*) \geqslant C_2 \cdot \max_{S \subseteq \mathcal{F}_{\mathcal{I},\min}, |S|=s} \text{minPD}(S)$

For the next definition, let $C_1 \leqslant 1$ and $C_2 \leqslant 1$.

Problem 5 ((C_1, C_2)-Diverse-Φ-Max)**.** **Input:** An instance I to the implicit set system Φ, $s \in \mathbb{N}$
Output: $S^* \subseteq \mathcal{F}_I$ of s elements such that every $z \in S^*$ has weight at least $C_1 \cdot \mathsf{OPT}_{\Phi\text{-Max}}(I)$, and $\textsc{min}\mathrm{PD}(S^*) \geqslant C_2 \cdot \max_{S \subseteq \mathcal{F}_{\mathcal{I},\max}, |S|=s} \textsc{min}\mathrm{PD}(S)$

5.1 Isometric reductions

Our first set of applications results from Theorem 26 on finding diverse satisfying assignments for a k-CNF formula that has Hamming weight at least (or at most) a prescribed value $W \in [n]$. Using "isometric" reductions between popular NP-complete optimization problems and SAT, we obtain bi-criteria approximation algorithms for diverse solutions of many NP-complete optimization problems. We formally define such reductions first.

Definition 3 (Isometric Reduction)**.** Consider two implicit set systems Φ_1 and Φ_2. A isometric reduction from Φ_1 to Φ_2 is given by a computable function f and a family of computable functions $\{g_I\}$ for every instance I of Φ_2. The function f takes as input an instance $I_1 \in \{0,1\}^*$ of Φ_1 with $\Phi(I_1) = (n_1, \mathcal{F}_1)$ and outputs an instance I_2 of Φ_2 with $\Phi(I_2) = (n_2, \mathcal{F}_2)$ such that $n_2 = n_1$ and $|\mathcal{F}_2| = |\mathcal{F}_1|$. The function g_{I_2} is a bijective function $g_{I_2} : \mathcal{F}_2 \to \mathcal{F}_1$, that has the following properties.

- For each $A \in \mathcal{F}_2$, $|A| = |g_{I_2}(A)|$.
- For any $A_1, A_2 \in \mathcal{F}_2$, $d_H(A_1, A_2) = d_H(g_{I_2}(A_1), g_{I_2}(A_2))$.

An isometric reduction preserves the geometry of the solution space. This implies the following theorem.

Theorem 35. *Consider two implicit set systems Φ_1 and Φ_2 such that there exists an isometric reduction $(f, \{g_I\})$ from Φ_1 to Φ_2. Suppose there exists an algorithm that solves the (C_1, C_2)-Diverse-Φ_2-Min problem with input instance I and $s \in \mathbb{N}$, running in time $\tau(n, s, |I|)$. Then, given an instance I_1 for Φ_1, and $s \in \mathbb{N}$, there exists an algorithm for (C_1, C_2)-Diverse-Φ_1-Min running in time $\tau_f + \tau(n, s, |f(I_1)|) + \tau_{g_{I_2}}$. Here, τ_f and τ_{g_I} denote the time it takes to compute the functions f and g_I.*

Clearly, an analogous theorem holds for (C_1, C_2)-Diverse-Φ_2-Max also. We now demonstrate some simple examples of isometric reductions, which imply the results in the first three rows of Table 1. We leave the task of finding more interesting isometric reductions to future work.

Maximum Independent Set: We begin by noting that an independent set instance can be written as a 2-CNF formula $\mathbf{F}_{IS}$: for every $v \in V$, we let $x_v \in \{0,1\}$ such that $x_v = 1$ if and only if v is chosen in the independent set. For every edge $e = (u,v) \in E$, we define the constraint $\neg x_u \vee \neg x_v$. Note that this constraint is satisfied if and only if at most one vertex participating in the edge is chosen in the independent set. The formula $\mathbf{F}_{IS}$ is a conjunction of all the constraints corresponding to the edges in E. Then an independent set of G corresponds to a satisfying assignment of $\mathbf{F}_{IS}$ and vice versa. Moreover, the Hamming weight of a satisfying assignment of $\mathbf{F}_{IS}$ is equal to the size of the corresponding independent set. Finding an independent set of maximum size is therefore equivalent to finding a satisfying assignment of $\mathbf{F}_{IS}$ of maximum Hamming weight. Moreover, the Hamming distance between two satisfying assignments z_1, z_2 corresponding to two independent sets I_1 and I_2 are preserved, in the sense that $d_H(z_1, z_2) = |I_1 \Delta I_2|$, where Δ denotes the symmetric difference between sets.

Minimum Vertex Cover: Every vertex cover instance can be written as a 2-CNF formula $\mathbf{F}_{VC}$: For every edge $e = (u, v) \in E$, we define the constraint $x_u \vee x_v$. Note that this constraint is satisfied if and only if at least one vertex participating in the edge is chosen in the vertex cover. The formula $\mathbf{F}_{VC}$ is a conjunction of all the constraints corresponding to the edges in E, which implies that a vertex cover of G corresponds to a satisfying assignment of $\mathbf{F}_{VC}$ and vice versa, and the Hamming weight of a satisfying assignment of $\mathbf{F}_{IS}$ is equal to the size of the vertex cover. Finding a vertex cover of minimum size is therefore equivalent to finding a satisfying assignment of $\mathbf{F}_{VC}$ of minimum Hamming weight, and it can be seen that this reduction is isometric.

Minimum d-hitting set: Recall that an instance of the d-hitting set problem consists of a family $\mathcal{S}$ of subsets of $[n]$ of size d, with the output being a subset of $[n]$ of minimum size that has a non-empty intersection with each subset in $\mathcal{S}$. This can easily be written as a d-CNF formula $\mathbf{F}$ as follows. For every set $S \in \mathcal{S}$, we define a clause C_S which is a disjunction of all the non-negated literals corresponding to the elements in S, with the formula $\mathbf{F}$ being the conjunction of the clauses corresponding to each $S \in \mathcal{S}$. Finding hitting set of minimum size corresponds to finding a satisfying assignment to this formula of minimum Hamming weight and it can be seen that this reduction is isometric as well.

Remark 9. We note that the problems of diverse vertex cover and diverse hitting set have been studied in the setting of parameterized complexity by [BJM$^+$19, BFJ$^+$22]. However, in these works the focus is on obtaining optimal solutions with optimal diversity and their results are not directly comparable to ours. A typical runtime from the existing results is of the type $2^{s\ell}$ where s is the number of solutions required and ℓ is the size of a solution (e.g., the size of the minimum vertex cover). Note that in some settings, $s\ell = \Omega(n^\alpha)$, for some $\alpha > 1$, rendering the above running time of 2^{n^α}. Our results in Theorem 35 state that at the cost of relaxing both the quality of the solutions obtained and for approximating the maximum dispersion, the running time can be reduced to $\mathrm{poly(s)} \cdot o(2^n)$.

5.2 Local feasibility search

What about problems for which we cannot define an isometric reduction to k-SAT? For several of those problems, we point out that the techniques developed in Section 4 are very general and can be adapted to deal with several optimization problems. For the applications in this section, we restrict our attention to minimization problems. We start with defining a version of local search for subset problems similar to Schöning's local search for k-SAT.

Definition 4. (Parameterized approximately-local feasibility search - (α, c)-PLFS) An (α, c)-PLFS algorithm for an implicit set system Φ takes as input an instance I for Φ, $A \in \{0,1\}^n$, and $t \in \mathbb{N}$, and if there exists a feasible solution $A' \in \mathcal{F}_I$ such that $d_H(A, A') \leqslant t$, outputs an $A^* \in \mathcal{F}_I$ such that $d_H(A, A^*) \leqslant \alpha t$ in time $c^t \cdot n^{O(1)}$.

When $\alpha = 1$ we just call the algorithm a PLFS algorithm. Note that there are several examples of problems admitting PLFS algorithms. For example, the algorithms $\mathsf{LS}_{1,k}$ and $\mathsf{LS}_{3,k-1}$ described in Section 4 for k-SAT. We also note that this is the exact same definition of a local search used in Section 4, generalized to subset problems.

Remark 10. Notice that a PLFS algorithm only searches for any feasible solution in $B(A, t)$, where $B(A, t)$ is the Hamming ball of radius t around A. We note that this is potentially easier than searching for a solution of minimum weight in $B(A,t)$. Indeed, for several graph problems, the existence of an algorithm running in time $f(t)\cdot n^{O(1)}$ that finds a solution of minimum weight in $B(A,t)$ is unlikely [FFL+12].

Theorem 36 (From PLFS to Dispersion)**.** *Let Φ be an implicit set system that admits an (α,c)-PLFS. Then, for every $0<\delta\leqslant\frac{2(1+\alpha)}{c-1}$, there exists an algorithm that takes as input an instance I to Φ, $s\in\mathbb{N}$, and, if $|\mathcal{F}_{I,min}|\geqslant s$, outputs $S^*\subseteq\mathcal{F}_I$ of size s such that $|A|\leqslant(1+\delta)\mathsf{OPT}_{\Phi\text{-}\textsc{Min}}(I)$ for all $A\in S^*$, and $\textsc{min}PD(S^*)\geqslant\frac{1}{2}(1-\delta)\max_{S\subseteq\mathcal{F}_{\mathcal{I},\textsc{min}},|S|=s}\textsc{min}PD(S)$. This algorithm runs in time $s^3\cdot n^{O(1)}\cdot\frac{2^nc^R}{\binom{n}{R}}$, where $R=\left\lfloor\frac{\delta n}{2(1+\alpha+\delta)}\right\rfloor$. In particular, when $\delta=\frac{2(1+\alpha)}{c-1}$, this algorithm runs in time $O^*\left(s^3\cdot\left(\frac{2}{1+1/c}\right)^n\right)$.*

Proof. TOPROVE 12 ☐

The question now is, which problems admit a PLFS algorithm? In the field of parametrized complexity, there is a huge body of work on FPT algorithms parametrized by the solution size. While this does not directly imply PLFS algorithms, the framework of monotone local search by Fomin, Gaspers, Lokshtanov, and Saurabh [FGLS19] provide a bridge connecting PLFS to FPT algorithms.

Monotone local search: For an implicit set system Φ, the cone of length t starting at a set $A\in\{0,1\}^n$ is defined to be $C(A,t):=\{A'\in\{0,1\}^n : A\subseteq A' \text{ and } |A\Delta A'|\leqslant t\}$. Φ admits a parameterized local monotone search algorithm if there exists an algorithm taking an instance I of Φ, a set $A\in\{0,1\}^n$ and $t\in[n]$ as input, and if $C(A,t)\bigcap\mathcal{F}_I$ is non-empty, outputs some $A^*\in C(A,t)\bigcap\mathcal{F}_I$ in time $c^t\cdot n^{O(1)}$ for some constant $c>1$.

Now, we prove that for the class of *hereditary* problems, the concepts of parameterized local feasibility search and parameterized local monotone search are in fact, equivalent. We remind the reader that we are dealing with minimization problems only.

Definition 5. An implicit set system Φ is called hereditary if for all instances I of Φ such that $\Phi(I)=(n,\mathcal{F})$, $\mathcal{F}$ satisfies the property that for any $A\subseteq B\subseteq[n]$, $A\in\mathcal{F}$ implies that $B\in\mathcal{F}$.

Lemma 37. *If a hereditary implicit set system Φ admits a parameterized monotone local search algorithm, then it also admits a PLFS algorithm that runs in the same time, and vice versa.*

Proof. TOPROVE 13 ☐

Lemma 37, along with Theorem 36 implies the existence of a c-PLFS for many combinatorial problems that were studied in [FGLS19]. We select the same problems and present them in the table below (instantiated at $C_1=3/2, C_2=1/4$), along with our results on isometric reductions.

We remark that both the isometric reduction and the PLFS approaches give s-dispersion algorithms for d-Hitting Set. However, the second approach yields an algorithm with better guarantees because the monotone search for d-hitting set is faster than the local search for d-SAT [FGK+10].

Problem	Extension [FGLS19]	MinOnes [FGLS19] One exact solution	s-Dispersion Bi-approx
d-Hitting Set ($d \geqslant 3$)	d^k	$(2-\frac{1}{d})^n$	Theorem 35
Vertex cover	2^k	1.5^n	$s^3 \cdot 1.5486^n$
Maximum independent Set	2^k	1.5^n	$s^3 \cdot 1.5486^n$
Feedback Vertex Set	3.592^k	1.7217^n	$s^3 \cdot 1.6420^n$
Subset Feedback Vertex Set	4^k	1.7500^n	$s^3 \cdot 1.6598^n$
Feedback Vertex Set in Tournaments	1.6181^k	1.3820^n	$s^3 \cdot 1.5162^n$
Group Feedback Vertex Set	4^k	1.7500^n	$s^3 \cdot 1.6598^n$
Node Unique Label Cover	$\lvert\Sigma\rvert^{2k}$	$(2-\frac{1}{\lvert\Sigma\rvert^2})^n$	Theorem 36
Vertex (r,ℓ)-Partization ($r,\ell \leqslant 2$)	3.3146^k	1.6984^n	$s^3 \cdot 1.6289^n$
Interval Vertex Deletion	8^k	1.8750^n	$s^3 \cdot 1.7789^n$
Proper Interval Vertex Deletion	6^k	1.8334^n	$s^3 \cdot 1.7284^n$
Block Graph Vertex Deletion	4^k	1.7500^n	$s^3 \cdot 1.6598^n$
Cluster Vertex Deletion	1.9102^k	1.4765^n	$s^3 \cdot 1.5415^n$
Thread Graph Vertex Deletion	8^k	1.8750^n	$s^3 \cdot 1.7789^n$
Multicut on Trees	1.5538^k	1.3565^n	$s^3 \cdot 1.51^n$
3-Hitting Set	2.0755^k	1.5182^n	$s^3 \cdot 1.5544^n$
4-Hitting Set	3.0755^k	1.6750^n	$s^3 \cdot 1.6167^n$
d-Hitting Set ($d \geqslant 3$)	$(d-0.9245)^k$	$(2-\frac{1}{d-0.9245})^n$	Theorem 36
Min-Ones 3-SAT	2.562^k	$s^3 \cdot 1.6097^n$	Theorem 26
Min-Ones d-SAT ($d \geqslant 4$)	d^k	$(2-\frac{1}{d})^n$	Theorem 26
Weighted d-SAT ($d \geqslant 3$)	d^k	$(2-\frac{1}{d})^n$	Theorem 26
Weighted Feedback Vertex Set	3.6181^k	1.7237^n	$s^3 \cdot 1.6432^n$
Weighted 3-Hitting Set	2.168^k	1.5388^n	$s^3 \cdot 1.5612^n$
Weighted d-Hitting Set ($d \geqslant 4$)	$(d-0.832)^k$	$(2-\frac{1}{d-0.832})^n$	Theorem 36

Table 2: The second column contains the time taken to obtain one exact solution using methods in [FGLS19]. The third Column contains the time taken to solve $(3/2, 1/4)$-Diverse-Φ-Min (except for Maximum Independent Set, where $(1/2, 1/4)$-Diverse-Φ-Max is solved)

5.3 Schöning's and PPZ algorithms run faster if $\Omega_{\mathbf{F}}$ contains dispersed solutions

In this section, we show that if $\Omega_{\mathbf{F}}$ contains a dispersed subset, then Schöning's algorithm as well as the PPZ algorithm find a satisfying assignment to $\mathbf{F}$ faster. Let $\Omega_{\mathbf{F}}$ denote the set of satisfying assignments to $\mathbf{F}$.

For every $r \in [n]$, we define

$$N_r := \max\{|S| : S \subseteq \Omega_{\mathbf{F}}, \textsc{minPD}(S) \geqslant r\}$$

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 $N_{0,\mathbf{F}} \geqslant N_{1,\mathbf{F}} \geqslant \cdots \geqslant N_{n,\mathbf{F}}$.

Theorem 38. *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.*

If the solution space $\Omega_{\mathbf{F}}$ contains a code of minimum distance $2r = 2n/k$, with $N_{2r} \geqslant 2^{n(1-H((2r-1)/n))}$ (using the Gilbert Varshamov bound), which is equal to $2^{n(1-H(2/k-1/n))}$. When $k \geqslant 6$, this gives an exponential improvement.

To prove this, recall Lemma 28 and Schöning's algorithm as described in Section 4.1.

Lemma 28. *For any starting assignment $z \in \{0,1\}^n$, if there exists a satisfying assignment $z^* \in \{0,1\}^n$ such that $d_H(z,z^*) \leqslant t$, then $\mathsf{SW}_t(\mathbf{F},z)$ outputs a satisfying assignment with probability at least k^{-t}. Furthermore, $\mathsf{SW}_{\lceil(1+2/(k-2))t\rceil}(\mathbf{F},z)$ outputs a satisfying assignment with probability at least $(k-1)^{-t}$.*

It consists of sampling z uniformly at random from $\{0,1\}^n$ and performing a Schöning walk for $3n$ steps starting from z. If $\mathbf{F}$ is satisfiable, for each $r \in [n]$, with probability $\frac{1}{2^n} \cdot \binom{n}{r}$, z is at Hamming distance $\leqslant r$ from a satisfying assignment, and we can calculate the probability that the Schöning walk ends in a satisfying assignment to be at least $\frac{1}{2^n}\binom{n}{r}\frac{1}{(k-1)^r}$. Hence, setting $r = \lfloor n/k \rfloor$, we can lower bound this probability by $\left(\frac{1}{2}\left(1+\frac{1}{k-1}\right)\right)^n$, using Appendix F.

However, we now note that due to the definition of N_r, for each $0 \leqslant r \leqslant \lfloor n/2 \rfloor$, there exist N_{2r} satisfying assignments to $\mathbf{F}$, with the Hamming balls of radius r around then being disjoint. Hence, for each $r \in [\lfloor n/2 \rfloor]$, with probability at least $\frac{N_{2r}\binom{n}{r}}{2^n}$, z is at distance r from a satisfying assignment, when chosen uniformly at random from $\{0,1\}^n$. This means that the success probability of the Schöning walk can be calculated to be at least $N_{\lfloor 2n/k \rfloor} \cdot \left(\frac{1}{2}\left(1+\frac{1}{k-1}\right)\right)^n$. This probability is clearly better than the probability of success for Schöning's algorithm. Hence, we obtain that the running time of Schöning's algorithm with a dispersion guarantee equals

$$\frac{2^n\,(1-1/k)^n}{N_{\lfloor 2n/k \rfloor}}\,.$$

Now we note that we can prove a similar statement for the PPZ algorithm.

Theorem 39. *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.*

Proof. TOPROVE 14 □

5.4 Approximating Opt-min for CSPs

It is not hard to see that Schöning's parametrized local search algorithm can be used to find diverse solutions for k-ary CSP's as well, that is, Lemma 28 generalizes to CSPs [Sch99][Section 3]. Formally, we prove the following theorem.

Theorem 40 (Schöning approximating Opt-min for CSPs)**.** *Let Ψ be a any constraint satisfaction problem over the alphabet $\{0,1\}$, and $s \in \mathbb{N}$. with the maximum arity of the constraints being k. For $0 < \delta \leqslant \min\{1, \frac{4(k-1)}{(k-2)^2}\}$, there exists an algorithm taking Ψ and s as input and, if Ψ has at least s distinct satisfying assignments, outputs a set S^* of s of satisfying assignments to Ψ such that $\textsc{min}PD(S^*) \geqslant \frac{1}{2}(1-\delta)\,\textsc{Opt-min}(\Psi,s)$. It runs in time $O^*\left(s^3 \cdot \frac{2^n(k-1)^R}{\binom{n}{R}}\right)$, where $R = \left\lfloor \frac{\delta n}{2(2+\delta+\frac{2}{k-2})} \right\rfloor$.*

References

[AB11] Andrea Arcuri and Lionel Briand. Formal analysis of the probability of interaction fault detection using random testing. *IEEE Transactions on Software Engineering*, 38(5):1088–1099, 2011.

[ACALM22] Amir Abboud, Vincent Cohen-Addad, Euiwoong Lee, and Pasin Manurangsi. Improved approximation algorithms and lower bounds for search-diversification problems. In *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2022.

[ACC$^+$10] Sabih Agbaria, Dan Carmi, Orly Cohen, Dmitry Korchemny, Michael Lifshits, and Alexander Nadel. SAT-based semiformal verification of hardware. In *Formal Methods in Computer Aided Design*, pages 25–32. IEEE, 20we10.

[AT04] Ola Angelsmark and Johan Thapper. Algorithms for the maximum Hamming distance problem. In *International Workshop on Constraint Solving and Constraint Logic Programming*, pages 128–141. Springer, 2004.

[AT05] 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.

[AW15] Josh Alman and Ryan Williams. Probabilistic polynomials and hamming nearest neighbors. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 136–150. IEEE, 2015.

[BFJ$^+$22] Julien Baste, Michael R Fellows, Lars Jaffke, Tomáš Masařík, Mateus de Oliveira Oliveira, Geevarghese Philip, and Frances A Rosamond. Diversity of solutions: An exploration through the lens of fixed-parameter tractability theory. *Artificial Intelligence*, 303:103644, 2022.

[BJKN10] Nikhil Bansal, Kamal Jain, Anna Kazeykina, and Joseph Naor. Approximation algorithms for diversified search ranking. In *Automata, Languages and Programming: 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6-10, 2010, Proceedings, Part II 37*, pages 273–284. Springer, 2010.

[BJM$^+$19] Julien Baste, Lars Jaffke, Tomáš Masařík, Geevarghese Philip, and Günter Rote. Fpt algorithms for diverse collections of hitting sets. *Algorithms*, 12(12):254, 2019.

[BLY12] Allan Borodin, Hyun Chul Lee, and Yuli Ye. Max-sum diversification, monotone submodular functions and dynamic updates. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI symposium on Principles of Database Systems*, pages 155–166, 2012.

[Bol86] Béla Bollobás. *Combinatorics: set systems, hypergraphs, families of vectors, and combinatorial probability.* Cambridge University Press, 1986.

[BS04] Sven Baumer and Rainer Schuler. Improving a probabilistic 3-SAT algorithm by dynamic search and independent clause pairs. In *Theory and Applications of Satisfiability Testing: 6th International Conference, SAT 2003, Santa Margherita Ligure, Italy, May 5-8, 2003, Selected Revised Papers 6*, pages 150–161. Springer, 2004.

[Cal09] Chris Calabro. *The exponential complexity of satisfiability problems.* University of California, San Diego, 2009.

[CEZ19] Alfonso Cevallos, Friedrich Eisenbrand, and Rico Zenklusen. An improved analysis of local search for max-sum diversification. *Mathematics of Operations Research*, 44(4):1494–1509, 2019.

[CIKp08] Chris Calabro, Russell Impagliazzo, Valentine Kabanets, and Ramamohan paturii. The complexity of unique k-SAT: An isolation lemma for k-CNFs. *Journal of Computer and System Sciences*, 74(3):386–393, 2008.

[CMM23] Zongchen Chen, Nitya Mani, and Ankur Moitra. From algorithms to connectivity and back: finding a giant component in random k-sat. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3437–3470. SIAM, 2023.

[CNW17] Jean Cardinal, Jerri Nummenpalo, and Emo Welzl. Solving and sampling with many solutions: Satisfiability and other hard problems. *arXiv preprint arXiv:1708.01122*, 2017.

[CR02] Pierluigi Crescenzi and Gianluca Rossi. On the Hamming distance of constraint satisfaction problems. *Theoretical Computer Science*, 288(1):85–100, 2002.

[EG04] Friedrich Eisenbrand and Fabrizio Grandoni. On the complexity of fixed parameter clique and dominating set. *Theoret. Comput. Sci.*, 326(1-3):57–67, 2004.

[FFL$^+$12] Michael R Fellows, Fedor V Fomin, Daniel Lokshtanov, Frances Rosamond, Saket Saurabh, and Yngve Villanger. Local search: Is brute-force avoidable? *Journal of Computer and System Sciences*, 78(3):707–719, 2012.

[FFV11] Uriel Feige, Abraham D Flaxman, and Dan Vilenchik. On the diameter of the set of satisfying assignments in random satisfiable k-cnf formulas. *SIAM Journal on Discrete Mathematics*, 25(2):736–749, 2011.

[FGK$^+$10] Fedor V Fomin, Serge Gaspers, Dieter Kratsch, Mathieu Liedloff, and Saket Saurabh. Iterative compression and exact algorithms. *Theoretical Computer Science*, 411(7-9):1045–1053, 2010.

[FGLS19] Fedor V. Fomin, Serge Gaspers, Daniel Lokshtanov, and Saket Saurabh. Exact algorithms via monotone local search. *J. ACM*, 66(2), mar 2019.

[FK13] Fedor V Fomin and Petteri Kaski. Exact exponential algorithms. *Communications of the ACM*, 56(3):80–88, 2013.

[GGK$^+$22] Jie Gao, Mayank Goswami, CS Karthik, Meng-Tsung Tsai, Shih-Yu Tsai, and Hao-Tsung Yang. Obtaining approximately optimal and diverse solutions via dispersion. In *Latin American Symposium on Theoretical Informatics*, pages 222–239. Springer, 2022.

[GHK10] Venkatesan Guruswami, Johan Hastad, and Swastik Kopparty. On the list-decodability of random linear codes. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 409–416, 2010.

[GKS12] Elena Grigorescu, Tali Kaufman, and Madhu Sudan. Succinct representation of codes with applications to testing. *SIAM Journal on Discrete Mathematics*, 26(4):1618–1634, 2012.

[Gon85] Teofilo F Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical computer science*, 38:293–306, 1985.

[GpP$^+$24] Mohit Gurumukhani, Ramamohan paturii, Pavel Pudlák, Michael Saks, and Navid Talebanfard. Local enumeration and majority lower bounds, 2024.

[GSS06] Carla P Gomes, Ashish Sabharwal, and Bart Selman. Near-uniform sampling of combinatorial spaces using XOR constraints. *Advances In Neural Information Processing Systems*, 19, 2006.

[Her14] Timon Hertli. Breaking the PPSZ barrier for unique 3-SAT. In *International Colloquium on Automata, Languages, and Programming*, pages 600–611. Springer, 2014.

[HHOW05] Emmanuel Hebrard, Brahim Hnich, Barry O'Sullivan, and Toby Walsh. Finding diverse and similar solutions in constraint programming. In *AAAI*, volume 5, pages 372–377, 2005.

[Hir98] Edward A Hirsch. A fast deterministic algorithm for formulas that have many satisfying assignments. *Logic Journal of IGPL*, 6(1):59–71, 1998.

[HKZZ19] 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.

[HMS11] Timon Hertli, Robin A Moser, and Dominik Scheder. Improving PPSZ for 3-SAT using critical variables. In *28th International Symposium on Theoretical Aspects of Computer Science*, page 237, 2011.

[HSSW02] Thomas Hofmeister, Uwe Schöning, Rainer Schuler, and Osamu Watanabe. A probabilistic 3-SAT algorithm further improved. In *STACS 2002: 19th Annual Symposium on Theoretical Aspects of Computer Science Antibes-Juan les Pins, France, March 14–16, 2002 Proceedings 19*, pages 192–202. Springer, 2002.

[IMMM14] Piotr Indyk, Sepideh Mahabadi, Mohammad Mahdian, and Vahab S Mirrokni. Composable core-sets for diversity and coverage maximization. In *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 100–108, 2014.

[Ip01] Russell Impagliazzo and Ramamohan paturii. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.

[IpZ01] Russell Impagliazzo, Ramamohan paturii, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

[IR77] Alon Itai and Michael Rodeh. Finding a minimum circuit in a graph. In *Proceedings of the ninth annual ACM symposium on Theory of computing*, pages 1–10, 1977.

[KK07] Nathan Kitchen and Andreas Kuehlmann. Stimulus generation for constrained random simulation. In *2007 IEEE/ACM International Conference on Computer-Aided Design*, pages 258–265. IEEE, 2007.

[KW16] Daniel Kane and Osamu Watanabe. A short implicant of a CNF formula with many satisfying assignments. *Algorithmica*, 76:1203–1223, 2016.

[Liu18] Sixue Liu. Chain, generalization of covering code, and deterministic algorithm for k-SAT. In *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[MS77] Florence Jessie MacWilliams and Neil James Alexander Sloane. *The theory of error-correcting codes*, volume 16. Elsevier, 1977.

[Nad11] Alexander Nadel. Generating diverse solutions in SAT. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 287–301. Springer, 2011.

[O'D21] Ryan O'Donnell. Analysis of boolean functions, 2021.

[Pap91] Christos H Papadimitriou. On selecting a satisfying truth assignment. In *[1991] Proceedings 32nd Annual Symposium of Foundations of Computer Science*, pages 163–169. IEEE Computer Society, 1991.

[PAP$^+$19] Quentin Plazar, Mathieu Acher, Gilles Perrouin, Xavier Devroey, and Maxime Cordy. Uniform sampling of SAT solutions for configurable systems: Are we there yet? In *2019 12th IEEE Conference on Software Testing, Validation and Verification (ICST)*, pages 240–251. IEEE, 2019.

[PPSZ05] 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.

[PPZ97] R. Paturi, P. Pudlak, and F. Zane. Satisfiability coding lemma. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, FOCS '97, page 566, USA, 1997. IEEE Computer Society.

[PT19] Thierry Petit and Andrew C Trapp. Enriching solutions to combinatorial problems via solution engineering. *INFORMS Journal on Computing*, 31(3):429–444, 2019.

[Rot06] Ron Roth. *Introduction to Coding Theory.* Cambridge University Press, 2006.

[RRT94] Sekharipuram S Ravi, Daniel J Rosenkrantz, and Giri Kumar Tayi. Heuristic and special case algorithms for dispersion problems. *Operations research*, 42(2):299–310, 1994.

[Sch78] Thomas J Schaefer. The complexity of satisfiability problems. In *Proceedings of the tenth annual ACM symposium on Theory of computing*, pages 216–226, 1978.

[Sch99] T Schöning. 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.

[Sch02] T Schöning. A probabilistic algorithm for k-SAT based on limited local search and restart. *Algorithmica*, 32:615–623, 2002.

[Sch19] Dominik Scheder. PPSZ for k $\geqslant$ 5: More is better. *ACM Trans. Comput. Theory*, 11(4), 2019.

[Sch22] Dominik Scheder. PPSZ is better than you think. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 205–216. IEEE, 2022.

[SJ89] Alistair Sinclair and Mark Jerrum. Approximate counting, uniform generation and rapidly mixing markov chains. *Information and Computation*, 82(1):93–133, 1989.

[SS17] 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.

[SW13] Manuel Schmitt and Rolf Wanka. Exploiting independent subformulas: A faster approximation scheme for# k-SAT. *Information Processing Letters*, 113(9):337–344, 2013.

[Wil05] Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Computer Science*, 348(2-3):357–365, 2005.

[Wil18] Virginia Vassilevska Williams. Some open problems in fine-grained complexity. *SIGACT News*, 49(4):29–35, dec 2018.

[WXXZ24] Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. New bounds for matrix multiplication: from alpha to omega. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3792–3835. SIAM, 2024.

A More Schöning-type algorithms

A.1 The case of small k and small δ.

In this section, we start by showing that by using $\mathsf{LS}_{1,k}$ instead of $\mathsf{LS}_{(1+2/(k-2)),k-1}$, we can handle the case of $k = 2$, and beat Theorem 25 for some δ. We state the theorem to begin with. We define $\tau_1(\delta, k, n)$ to be $\frac{2^n k^{R_1}}{\binom{n}{R_1}}$ where $R_1 = \left\lfloor \frac{\delta n}{2(2+\delta)} \right\rfloor$ for each $\delta \in \left(0, \min\{1, \frac{4}{k-1}\}\right]$

Theorem 41 (Schöning for Diam). *Let* $\mathbf{F}$ *be a* k*-CNF formula on* n *variables, for any* $k \geqslant 2$*. For each* $0 < \delta \leqslant \min\{1, \frac{4}{k-1}\}$*, there exists an algorithm taking* $\mathbf{F}$ *as input and running in time* $O^*(\tau_1(\delta, k, n))$ *that outputs* $z_1^*, z_2^* \in \Omega_{\mathbf{F}}$ *such that* $d_H(z_1^*, z_2^*) \geqslant \frac{1}{2} \cdot (1-\delta)\,\textsc{Diam}(\mathbf{F})$*, if* $\mathbf{F}$ *is satisfiable.*

Proof. TOPROVE 15 □

Remark 11. We can also use $\mathsf{LS}_{1,k}$ instead of $\mathsf{LS}_{(1+2/(k-2)),k-1}$ in all the algorithms for dispersion as well, to get identical theorems, with the running times using $\tau_1(\delta, k, n)$ instead of $\tau(\delta, k, n)$ and δ can be tuned between 0 and $\frac{4}{k-1}$ instead of $\frac{4}{k-1}\left(1 + \frac{2}{k-2}\right)^2$. We do not restate all of them for the sake of brevity.

Comparison between Theorem 25 and Theorem 41. Not only does Theorem 41 handle the case of $k = 2$, it also outperforms Theorem 25 for some cases of δ. As before, we define $a_{k,\delta}$ such that $a_{k,\delta}^n = \tau(\delta, k, n)$ and $b_{k,\delta}^n = \tau_1(\delta, k, n)$. We plot $a_{k,\delta}$ and $b_{k,\delta}$ together, for different values of k to illustrate the comparison. It can be seen that for $k = 3$, the algorithm in Theorem 41 always outperforms the algorithm in Theorem 25, and for larger values of k, it outperforms for smaller valuse of δ.

A.2 Schöning-based algorithm for the sum dispersion measure: Proof of Theorem 27.

To start with, we restate Theorem 27.

Theorem 27. *[Schöning approximating* $\textsc{Opt-sum}(\mathbf{F}, s)$*] Let* $\mathbf{F}$ *be a* k*-CNF formula on* n *variables and* $s \in \mathbb{N}$*. For each* $0 < \delta \leqslant \min\left\{1, \frac{4(k-1)}{(k-2)^2}\right\}$*, there exists an algorithm that takes* $\mathbf{F}, s$ *as input and runs in time* $O^*\left(s^3 \cdot \tau(\delta, k, n)\right)$ *that outputs, with probability* $1 - o(1)$*, a multi-set* $S^* \subseteq \Omega_{\mathbf{F}, \geqslant(1-\delta)W}$ *of size* s *such that*

$$\textsc{sum}PD(S^*) \geqslant \begin{cases} \frac{1}{2}(1-\delta)\,\textsc{Opt-sum}(\mathbf{F}, s) & \text{if } s \leqslant 3 + \left\lfloor \frac{2\delta}{1-\delta} \right\rfloor \\ \frac{1-\delta}{1+\delta}\left(\frac{1-\frac{1}{s}}{1+\frac{1-\delta}{1+\delta}\cdot\frac{1}{s}}\right)\textsc{Opt-sum}(\mathbf{F}, s) & \text{if } s > 3 + \left\lfloor \frac{2\delta}{1-\delta} \right\rfloor \end{cases}$$

To prove this theorem, we show that Schöning's algorithm can be modified to be a farthest point oracle for $\textsc{sum}\text{-}d_H$.

Lemma 42. *Consider a local search algorithm* $\mathsf{LS}_{\alpha,c}$*. Then, for every* $0 < \delta \leqslant \frac{2(1+\alpha)}{c-1}$*, there exists an algorithm running in time* $s^2 \cdot n^{O(1)} \cdot \frac{2^n c^R}{\binom{n}{R}}$*, where* $R = \left\lfloor \frac{\delta n}{2(1+\alpha+\delta)} \right\rfloor$ *that takes as input* $\mathbf{F}$ *and a multi-set* $S \subseteq \{0,1\}^n$ *of size* s *and if* $\mathbf{F}$ *is satisfiable, outputs* $z^* \in \Omega_{\mathbf{F}}$ *such that* $\textsc{sum}\text{-}d_H(z^*, S) \geqslant (1-\delta) \cdot \max_{z' \in \Omega_{\mathbf{F}}} \textsc{sum}\text{-}d_H(z', S)$ *with probability at least* $1 - 2^{-2n}$*.*

Proof. TOPROVE 16 □

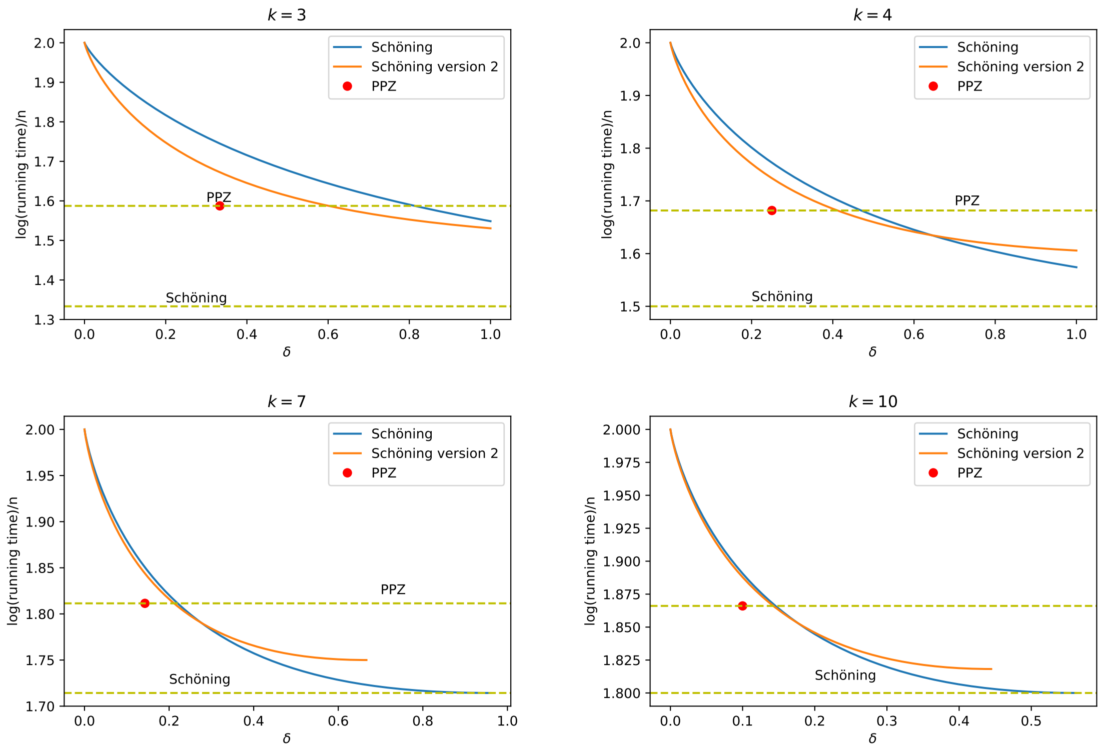

Figure 2: Plots of $a_{k,\delta}$ (labeled Schöning) and $b_{k,\delta}$ (labeled Schöning version 2)with respect to δ (on x-axis) for $k = 3, 4, 5$.

Proof of Theorem 27

The proof of Theorem 27 is similar to that of Theorem 15. We use the algorithm Schöning-Farthest-Sum (Lemma 42) as a $(1 - \delta)$-farthest point oracle, in the algorithm defined by Lemma 23. The approximation, running time guarantees and the range of δ that the two algorithms handle follows from the bounds stated in Lemma 42 for $c = k, \alpha = 1$ and $c = k - 1, \alpha = 1 + \frac{2}{k-2}$.

B Technical lemmas using approximate farthest point oracles

In this section, we design approximation algorithms for computing $\text{Opt-sum}(\mathbf{F}, s)$ and $\text{Opt-min}(\mathbf{F}, s)$, proving Theorem 15, Theorem 7, Theorem 27 and Theorem 26.

B.1 Sum dispersion: the proof of Lemma 23

Lemma 23. *Suppose there exists a $1 - \delta$-approximate farthest point oracle, $\mathcal{O}$ that takes a k-CNF formula $\mathbf{F}$ and a multi-set $S \subseteq \{0,1\}^n$ and with probability $1 - 2^{-2n}$, outputs $z^* \in \Omega_{\mathbf{F}}$ such that $\text{sum-}d_H(S, z^*) \geqslant (1 - \delta) \cdot \max_{z' \in \Omega_{\mathbf{F}}} \text{sum-}d_H(S, z')$. Then, there exists an algorithm taking $\mathbf{F}$ and s as input that uses $s^3 n$ calls to $\mathcal{O}$ (and an additional $s^4 n^{O(1)}$ overhead) that outputs a multi-set $S^* \subseteq \Omega_{\mathbf{F}}$ with $\text{sum}PD(S^*) \geqslant \max\{\frac{1}{2}(1 - \delta), \frac{(1-\delta)(s-1)}{(1+\delta)s+(1-\delta)}\} \cdot \text{Opt-sum}(\mathbf{F}, s)$ with probability $1 - o(1)$.*

To prove this lemma, consider the following algorithm, which is the same as the algorithm studied in [CEZ19], with the small difference being that we deal with multi-sets instead of sets.

Algorithm 6: Algorithm for Sum Dispersion

```
Input: A $k$-CNF formula $\mathbf{F}$, a number $s$, the oracle $\mathcal{O}$
Output: $S \in \Omega_{\mathbf{F}}^s$ with $\textsc{sum}\mathrm{PD}(S) \geqslant \max\{\frac{1}{2}(1-\delta), \frac{(1-\delta)(s-1)}{(1+\delta)s+(1-\delta)}\} \cdot \textsc{Opt-sum}(\mathbf{F}, s)$ if $\mathbf{F}$ is
        satisfiable, $\perp$ otherwise.
1  Use the PPZ algorithm (or Schöning's algorithm) to find a satisfying assignment $z_1^*$ to $\mathbf{F}$. ;
2  Set $S \leftarrow \{z_1^*\}$ ;
3  for $i \in \{2, 3, \ldots, s\}$ do
4      $z^* := \mathcal{O}(\mathbf{F}, S)$
5      $S \leftarrow S \bigcup \{z^*\}$
6  repeat $s^2 n$ times:
7      for $z \in S$ do
8          $z^* := \mathcal{O}(\mathbf{F}, S \backslash \{z\})$
9          if $\textsc{sum}\text{-}d_H(S\backslash\{z\}, z^*) > \textsc{sum}\text{-}d_H(S\backslash\{z\}, z)$ then
10             $S \leftarrow S\backslash\{z\} \bigcup \{z^*\}$
11 Output $S$
```

Because $\mathcal{O}$ is invoked at most $s^2 n$ times during the whole duration of the algorithm, the union bound implies that with probability at most $1 - o(1)$, $\mathcal{O}$ behaves as a $1-\delta$-approximate farthest point oracle in every iteration (because $\mathcal{O}$ behaves as a $1-\delta$-approximate farthest point oracle in every iteration with probability $1 - 2^{-2n}$).

The algorithm described above combines $\mathcal{O}$ with the well-known farthest point insertion algorithm [RRT94] for dispersion in metric spaces to get an algorithm that outputs a multiset $S \subseteq \Omega_{\mathbf{F}}$ with $|S| = s$ with the property that $\textsc{sum}\mathrm{PD}(S) \geqslant \frac{1-\delta}{2} \cdot \textsc{Opt-sum}(\mathbf{F}, s)$.

If s is large, we can further improve the approximation factor by repeatedly employing the following natural local search procedure on the set S. For each $z \in S$, we use the farthest point oracle with $\mathbf{F}$ and $S\backslash\{z\}$ as input. If z^*, the satisfying assignment output by the farthest point oracle satisfies $\textsc{sum}\text{-}d_H(z^*, S\backslash\{z\}) > \textsc{sum}\text{-}d_H(z, S\backslash\{z\})$ (which is equivalent to the condition that $\textsc{sum}\mathrm{PD}(S\backslash\{z\} \bigcup \{z^*\})) > \textsc{sum}\mathrm{PD}(S)$), we replace z by z^* in S. We show that at the end of $s^2 n$ iterations, $\textsc{sum}\mathrm{PD}(S) \geqslant \frac{(1-\delta)(s-1)}{(1+\delta)s+(1-\delta)} \cdot \textsc{Opt-sum}(\mathbf{F}, s)$. Because this local search procedure only increases the value of $\textsc{sum}\mathrm{PD}(S)$, this would complete the proof of Lemma 23.

We start with lower bounding $\textsc{sum}\mathrm{PD}(S)$ at the end of the farthest point insertion procedure. We start with proving the following lemma. For a multiset S, denote $|S|$ to be its cardinality counting multiplicities, and for two multisets A and B, we use $d_H(A, B) = \sum_{a \in A, b \in B} d_H(a, b)$

Observation 43. Let $A, B \subseteq \{0,1\}^n$ be two multisets. There exists $b \in B$ such that $\textsc{sum}\text{-}d_H(A, b) \geqslant \frac{|A|}{|B|(|B|-1)} \cdot \textsc{sum}\mathrm{PD}(B)$.

Proof. TOPROVE 17 ☐

Now, let $S_{\mathrm{OPT}} \subseteq \Omega_{\mathbf{F}}$ be a multiset of size s with $\textsc{sum}\mathrm{PD}(S_{\mathrm{OPT}}) = \textsc{Opt-sum}(\mathbf{F}, s)$. Observation 43 implies that the step when $|S| = i$, there exists $z \in S_{\mathrm{OPT}}$ with $\textsc{sum}\text{-}d_H(S, z) \geqslant \frac{i}{s(s-1)} \cdot \textsc{Opt-sum}(\mathbf{F}, s)$. Hence, the point z^* added to S at step i by $\mathcal{O}$ satisfies $\textsc{sum}\text{-}d_H(S, z^*) \geqslant \frac{i(1-\delta)}{s(s-1)} \cdot$

$\textsc{Opt-sum}(\mathbf{F}, s)$. We now show by induction that once the i-th point z^* is added by algorithm, $\textsc{sum}\mathrm{PD}(S) \geqslant \frac{i(i-1)(1-\delta)}{2s(s-1)} \cdot \textsc{Opt-sum}(\mathbf{F}, s)$. This is trivially true when $|S| = 1$. Assume that when $|S| = i-1$, $\textsc{sum}\mathrm{PD}(S) \geqslant \frac{(i-1)(i-2)(1-\delta)}{2s(s-1)} \cdot \textsc{Opt-sum}(\mathbf{F}, s)$. Because the point z^* added to S next satisfies $\textsc{sum-}d_H(S, z^*) \geqslant \frac{(i-1)(1-\delta)}{s(s-1)} \cdot \textsc{Opt-sum}(\mathbf{F}, s)$, the value of $\textsc{sum}\mathrm{PD}(S)$ at the end of round i is at least $\left(\frac{(i-1)(i-2)(1-\delta)}{2s(s-1)} + \frac{(i-1)(1-\delta)}{s(s-1)}\right) \cdot \textsc{Opt-sum}(\mathbf{F}, s) = \frac{i(i-1)(1-\delta)}{2s(s-1)} \cdot \textsc{Opt-sum}(\mathbf{F}, s)$. Since $i = s$, at the end of the farthest point insertion procedure, $\textsc{sum}\mathrm{PD}(S) \geqslant \frac{(1-\delta)}{2} \cdot \textsc{Opt-sum}(\mathbf{F}, s)$.

We now show that at the end of the local search procedure, $\textsc{sum}\mathrm{PD}(S) \geqslant \frac{(1-\delta)(s-1)}{(1+\delta)s+(1-\delta)} \cdot \textsc{Opt-sum}(\mathbf{F}, s)$. At each step of the procedure, either $\textsc{sum}\mathrm{PD}(S)$ increases by at least 1, or $\textsc{sum}\mathrm{PD}(S)$ remains unchanged (such an S is called a 'local optimum'). Observe that at any iteration, if the value of $\textsc{sum}\mathrm{PD}(S)$ is unchanged at the end of it, it also does not change during any of the later iterations. Because $\textsc{sum}\mathrm{PD}(A) \leqslant s^2 n$ for any multiset $A \subseteq \{0,1\}^n$ of size s, the algorithm reaches a local optimum within $s^2 n$ iterations.

Consider any set S which is a local optimum, and a set S_{OPT}, such that $\textsc{sum}\mathrm{PD}(S_{\mathrm{OPT}}) = \textsc{Opt-sum}(\mathbf{F}, s)$. Because the local search employed on S does not improve $\textsc{sum}\mathrm{PD}(S)$, the property of $\mathcal{O}$ implies that

$$\textsc{sum-}d_H(S\backslash\{x\}, x) \geqslant (1-\delta) \cdot \textsc{sum-}d_H(S\backslash\{x\}, y) \text{ for all } x \in S, y \in \Omega_{\mathbf{F}} .$$

Specifically, this holds for all $y \in S_{\mathrm{OPT}}$. Hence, we can sum over all $x \in S, y \in S_{\mathrm{OPT}}$ to obtain that

$$s \cdot \textsc{sum}\mathrm{PD}(S) \geqslant \frac{(1-\delta)(s-1)}{2} \cdot d_H(S, S_{\mathrm{OPT}}) , \tag{3}$$

where $d(S, S_{\mathrm{OPT}}) = \sum_{x \in S, y \in S_{\mathrm{OPT}}} d_H(x, y)$. We now use the inequality that

$$d_H(S, S_{\mathrm{OPT}}) \geqslant \textsc{sum}\mathrm{PD}(S) + \textsc{sum}\mathrm{PD}(S_{\mathrm{OPT}}) . \tag{4}$$

This follows from the fact that the Hamming metric is of negative type [CEZ19, Lemma 1] We now use this in Equation (3) to obtain that

$$s \cdot \textsc{sum}\mathrm{PD}(S) \geqslant \frac{(1-\delta)(s-1)}{2} \left(\textsc{sum}\mathrm{PD}(S) + \textsc{sum}\mathrm{PD}(S_{\mathrm{OPT}})\right)$$

Rearranging, this implies that $\textsc{sum}\mathrm{PD}(S) \geqslant \frac{(1-\delta)(s-1)}{(1+\delta)s+(1-\delta)} \cdot \textsc{Opt-sum}(\mathbf{F}, s)$.

B.2 Min Dispersion: the proof of Lemma 24

Lemma 24. *Suppose there exists a $1-\delta$-approximate farthest point oracle, $\mathcal{O}$ that takes a k-CNF formula $\mathbf{F}$ and a set $S \subseteq \{0,1\}^n$ as input and with probability $1 - 2^{-2n}$, outputs $z^* \in \Omega_{\mathbf{F}}$ such that $\textsc{min-}d_H(S, z^*) \geqslant (1-\delta) \cdot \max_{z' \in \Omega_{\mathbf{F}}} \textsc{min-}d_H(S, z')$. Then, there exists an algorithm taking $\mathbf{F}$ and s as input that uses s calls to $\mathcal{O}$ (and an additional $sn^{O(1)}$ overhead) that outputs a set $S^* \subseteq \Omega_{\mathbf{F}}$ with $\textsc{min}PD(S^*) \geqslant \frac{1}{2}(1-\delta) \cdot \textsc{Opt-min}(\mathbf{F}, s)$ with probability $1 - o(1)$.*

To prove this lemma, consider the following farthest point insertion algorithm, originally studied by Gonzales [Gon85].

Algorithm 7: Min Dispersion

```
Input: A $k$-CNF formula $\mathbf{F}$, a number $s$
Output: $S \in \Omega^s_{\mathbf{F}}$ with $\textsc{minPD}(S) \geqslant \frac{1}{2}(1-\delta) \cdot \textsc{Opt-min}(\mathbf{F}, s)$ if $\mathbf{F}$ is satisfiable, $\perp$
        otherwise.
1 Use the PPZ algorithm (or Schöning's algorithm) to find a satisfying assignment $z^*_1$ to $\mathbf{F}$. ;
2 Set $S \leftarrow \{z^*_1\}$ ;
3 for $i \in \{2, 3, \ldots, s\}$ do
4     $z^* := \mathcal{O}(\mathbf{F}, S)$
5     $S \leftarrow S \bigcup \{z^*\}$
```

Because $\mathcal{O}$ is invoked at most s times during the whole duration of the algorithm, the union bound implies that with probability at most $1 - o(1)$, behaves as approximate farthest point oracle each time it is invoked. Next, we show that at the end of the algorithm, $\textsc{minPD}(S) \geqslant \frac{1}{2}(1-\delta) \cdot \textsc{Opt-min}(\mathbf{F}, s)$ using induction. First, observe that $\textsc{minPD}(\{z^*_1, z^*_2\}) \geqslant \frac{1}{2}(1-\delta)\textsc{Opt-min}(\mathbf{F}, 2)$ using the triangle inequality. Suppose that before the i-th iteration of the algorithm, $|S| = i-1$ and $\textsc{minPD}(S) \geqslant \frac{1}{2}(1-\delta) \cdot \textsc{Opt-min}(\mathbf{F}, i-1)$. Let $S_{\mathrm{OPT}} \subseteq \Omega_{\mathbf{F}}$ be a set of size i with $\textsc{minPD}(S_{\mathrm{OPT}}) = \textsc{Opt-min}(\mathbf{F}, i)$. Observation 44 (stated and proved below) implies that there exists $x \in S_{\mathrm{OPT}}$ such that $\textsc{min-}d_H(x, S) \geqslant 1/2 \cdot \textsc{Opt-min}(\mathbf{F}, i)$. Hence, the assignment added to S at step i, z^* satisfies $\textsc{min-}d_H(S, z^*) \geqslant \frac{1}{2}(1-\delta) \cdot \textsc{Opt-min}(\mathbf{F}, i)$, which implies that $\textsc{minPD}(S) \geqslant \frac{1}{2}(1-\delta) \cdot \textsc{Opt-min}(\mathbf{F}, i)$ at the end of the i-th iteration.

Observation 44. [Farthest Point insertion] Let $A, B \subseteq \{0,1\}^n$ be two subsets with $|A| < |B|$. Then there exists $b \in B$ such that $\textsc{min-}d_H(b, A) \geqslant 1/2 \cdot \textsc{minPD}(B)$.

Proof. TOPROVE 18 ☐

C On returning sets instead of multisets

In this section, we extend our results for $\textsc{Opt-sum}$ to $\textsc{Opt-sum}_{\neq}$. Recall that the algorithm for $\textsc{Opt-sum}$ returned a multiset of size s that is an approximation of $\textsc{Opt-sum}(\mathbf{F}, s)$.

We showed that given any multi-set $T \subseteq \{0,1\}^n$, the sequence obtained from repeatedly sampling from $\{0,1\}^n \times S_n$ and running PPZ-Modify contains z^* such that $\textsc{sum-}d_H(z^*, T) \geqslant \frac{k-1}{k+1} \cdot r_{sum}$, where $r_{sum} = \max_{z \in \Omega_{\mathbf{F}}} \textsc{sum-}d_H(z, T)$. In this section, we extend that result to the $\textsc{Opt-sum}_{\neq}$ problem.

Lemma 45. *Let $\mathbf{F}$ be a satisfiable k-SAT formula, $T \subseteq \{0,1\}^n$ be a set of size $t = o\left(\frac{n}{\log(n)}\right)$, and $r_{sum} = \max_{z \in \Omega_{\mathbf{F}} \setminus T} \textsc{sum-}d_H(z, T)$. Let y and π be chosen uniformly at random from $\{0,1\}^n$ and S_n respectively. The probability that PPZ-Modify$(\mathbf{F}, y, \pi)$ outputs $z^* \in \Omega_{\mathbf{F}} \setminus T$ with $\textsc{sum-}d_H(z^*, T) \geqslant \frac{k-1}{k+1} \cdot r_{sum}$ is at least $2^{-n+n/k-o(1)}$*

Proof. TOPROVE 19 ☐

Now, we have shown that there exists an approximate farthest point oracle for computing $\max_{z \in \Omega_{\mathbf{F}} \setminus T} \textsc{sum-}d_H(z, T)$, as long as $|T| = o\left(\frac{n}{\log(n)}\right)$. Hence, we can now use this approximate farthest point oracle in the algorithms by Gonzales and Cevallos, Eisenbrand, and Zenklusen, proving the following theorem.

Theorem 46. *[PPZ approximating* $\textsc{Opt-sum}_{\neq}(\mathbf{F}, s)$*] Let* $\mathbf{F}$ *be a* k*-CNF formula on* n *variables. There exists a randomized algorithm that takes* $\mathbf{F}$ *and an integer* $s \geqslant 1$ *as input and if* $\mathbf{F}$ *is satisfiable and has at least* s *satisfying assignments, with probability at least* $1 - o(1)$*, outputs a set* $S^* \subseteq \Omega_{\mathbf{F}}$ *of size* s *such that:*

1. $\textsc{sum}PD(S^*) \geqslant \frac{1}{2} \cdot \left(1 - \frac{2}{k+1}\right) \cdot \textsc{Opt-sum}(\mathbf{F}, s)$ *if* $s \leqslant \left\lfloor \frac{3k+1}{k-1} \right\rfloor$.

2. $\textsc{sum}PD(S^*) \geqslant \frac{k-1}{k+3} \left(\frac{1-\frac{1}{s}}{1+\frac{k-1}{(k+3)} \cdot \frac{1}{s}} \right) \cdot \textsc{Opt-sum}(\mathbf{F}, s)$ *if* $s \geqslant \left\lceil \frac{3k+1}{k-1} \right\rceil$.

The algorithm runs in time $O^*\left(2^{n-n/k+o(n)}\right)$*, as long as* $s = o\left(\frac{n}{\log(n)}\right)$

D Relationship between Min-Ones and Farthest-Point

In this section, we point out that a farthest point oracle can be derived from an algorithm that outputs a satisfying assignment to $\mathbf{F}$ with minimum weight. This problem, formally called $\textsc{Min-Ones}-k-\text{SAT}$ has an exact algorithm that runs in time $O^*\left(\left(2 - \frac{1}{k}\right)^n\right)$. For simplicity, we define the decision versions of these problems.

Problem 6 (Min-Ones)**. Input:** A k-CNF formula $\mathbf{F}$, $r \in [n]$.
Output: Yes, if there exists $z^* \in \Omega_{\mathbf{F}}$ such that $|z^*| \leqslant r$, No otherwise.

Problem 7 (Farthest-Point)**. Input:** A k-CNF formula $\mathbf{F}$, $z \in \{0,1\}^n$, $r \in [n]$.
Output: Yes, if there exists $z^* \in \Omega_{\mathbf{F}}$ such that $d_H(z^*, z) \geqslant r$, No otherwise.

We now show that the problems Min-Ones and Farthest-Point are equivalent to each other.

Lemma 47. *There exists a reduction, running in* $n^{O(1)}$ *time, from* Min-Ones *to* Farthest-Point *and vice versa*

Proof. TOPROVE 20 □

E Using Uniform Sampling to generate diverse satisfying assignments

Let $\mathcal{A}$ be an algorithm that takes in $\mathbf{F}$ as input, and in $O^*(a^n)$ running time, outputs a satisfying assignment to $\Omega_{\mathbf{F}}$ such that each $z \in \Omega_{\mathbf{F}}$ is output with probability $1/|\Omega_{\mathbf{F}}|$ (in other words, it uniformly samples over the space of satisfying assignments). Note that because k-SAT is a self reducible problem[13], an algorithm for $\#k$-SAT, that counts the number of satisfying assignments can be used to also uniformly sample from the space of satisfying assignments. We define the following algorithm that approximates the diameter of $\Omega_{\mathbf{F}}$, using the uniform sampler $\mathcal{A}$ as a black box. It runs in time $O^*(b^n)$, where b^n is some time budget that we choose.

[13]For the class of problems that are 'self reducible', counting and sampling are equivalent, and approximate counting and approximate sampling are equivalent as well [SJ89]

Theorem 48. *Let $\mathbf{F}$ be a k-SAT formula with at least 2 satisfying assignments. Let $\mathcal{A}$ be an algorithm that uniformly samples satisfying assignments to k-SAT instances that runs in time $O^*(a^n)$. Consider any $b > a$. There exists an algorithm that runs in time $O^*(b^n)$ and with probability $1-o(1)$, and outputs two satisfying assignments $z_1, z_2 \in \Omega_{\mathbf{F}}$, with $d_H(z_1, z_2) \geqslant \min\{\frac{1}{2}, H^{-1}(\log(b/a))\} \cdot \textsc{Diam}(\mathbf{F})$.*

Proof. TOPROVE 21 □

Comparison to our results: We now perform some calculations assuming an approximation guarantee of $1/H^{-1}(\log(b/a))$ for the above algorithm. We use the state of the art existing algorithms for $\#k$ -SAT to come up with bounds for the run-time and approximation factors and compare them with our more 'geometry-based' sampling algorithms we propose.

For 3-SAT, the best known approximate counting algorithms are by Schmitt and Wanka [SW13], running in time $O^*(1.51426^n)$.

Hence, we can calculate the approximation factor this algorithm achieves for $k = 3$, where the budget $b = 2^{2/3}$. To do that, we plug in $b = 2^{2/3}$, and $a = 1.51426$ in $\frac{1}{H^{-1}(b/a)}$, which is $1/123$. This means that the sampling algorithm gives a $1/123$-approximation factor for the diameter of 3-SAT. On the other hand, our Theorem 4 gives a $1/3$-approximation ratio in the same running time. We remark that this gap widens as k increases.

F Schöning run time calculation

Lemma 49. *For every $t \in [n]$, $\frac{2^n}{\binom{n}{t}c^{-t}} \geqslant \frac{1}{n^{O(1)}} \cdot \frac{2^n}{\binom{n}{\lfloor \frac{n}{c+1} \rfloor} c^{-\lfloor \frac{n}{c+1} \rfloor}} = \frac{1}{n^{O(1)}} \cdot \left(\frac{2}{1+\frac{1}{c}}\right)^n$.*

Proof. TOPROVE 22 □