THEORY OF COMPUTING, Volume 12 (2), 2016, pp. 1–34
www.theoryofcomputing.org

Lattice Sparsification and the Approximate Closest Vector Problem

Daniel Dadush Gábor Kun*

Received July 11, 2013; Revised January 11, 2016; Published June 16, 2016

Abstract: We give a deterministic algorithm for solving the $(1+\varepsilon)$-approximate Closest Vector Problem (CVP) on any n-dimensional lattice and in any near-symmetric norm in $2^{O(n)}(1+1/\varepsilon)^n$ time and $2^n\,\mathrm{poly}(n)$ space. Our algorithm builds on the lattice point enumeration techniques of Micciancio and Voulgaris (STOC 2010, SICOMP 2013) and Dadush, Peikert and Vempala (FOCS 2011), and gives an elegant, deterministic alternative to the "AKS Sieve"-based algorithms for $(1+\varepsilon)$-CVP (Ajtai, Kumar, and Sivakumar; STOC 2001 and CCC 2002). Furthermore, assuming the existence of a $\mathrm{poly}(n)$-space and $2^{O(n)}$-time algorithm for exact CVP in the ℓ_2 norm, the space complexity of our algorithm can be reduced to polynomial.

Our main technical contribution is a method for "sparsifying" any input lattice while approximately maintaining its metric structure. To this end, we employ the idea of random sublattice restrictions, which was first employed by Khot (FOCS 2003, J. Comp. Syst. Sci. 2006) for the purpose of proving hardness for the Shortest Vector Problem (SVP) under ℓ_p norms.

ACM Classification: F.2.2, F.1.2, G.1.6

AMS Classification: 68W25, 68Q25, 68W20

Key words and phrases: approximation algorithms, high dimensional geometry, lattice algorithms, closest vector problem

A preliminary version of this paper appeared in the Proc. of the 24th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA'13).

*Research was supported by Subhash Khot's NSF Waterman Award CCF-1061938, the MTA Rényi "Lendület" Groups and Graphs Research Group and by the Marie Curie IIF Fellowship Grant No. 627476.

DOI: 10.4086/toc.2016.v012a002

1 Introduction

An n-dimensional lattice $\mathcal{L}$ is defined as a set

$$\left\{\sum_{i=1}^{n} z_i \mathbf{b}_i : z_i \in \mathbb{Z}, i \in [n]\right\}$$

for some basis $\mathbf{b}_1, \ldots, \mathbf{b}_n$ of $\mathbb{R}^n$. Given a lattice $\mathcal{L}$ and norm $\|\cdot\|$ in $\mathbb{R}^n$, the Shortest Vector Problem (SVP) is to find a shortest *non-zero* $\mathbf{v} \in \mathcal{L}$ under $\|\cdot\|$. Given an additional target $\mathbf{t} \in \mathbb{R}^n$, the Closest Vector Problem (CVP)—the inhomogeneous analog of SVP—is to find a closest $\mathbf{v} \in \mathcal{L}$ to $\mathbf{t}$. Here, one often works with the ℓ_2 norm and other ℓ_p norms, or, most generally, with (possibly asymmetric) norms induced by a convex body K containing 0 in its interior, defined by

$$\|\mathbf{x}\|_K = \inf\{s \geq 0 : \mathbf{x} \in sK\}\,.$$

The SVP and CVP on lattices are central algorithmic problems in the geometry of numbers, with applications to Integer Programming [32], factoring polynomials over the rationals [31], cryptanalysis (e. g., [42, 25, 41]), and much more. For different applications, one must often consider lattice problems expressed under a variety of norms. As examples, decoding signals over a Gaussian channel is expressed as a CVP under ℓ_2 [48], computing simultaneous Diophantine approximations is generally expressed as an SVP under ℓ_∞ [18], Schnorr reduced (under some unproven number theoretic assumptions) factoring to an SVP under the ℓ_1 norm [46], the Frobenius problem can be expressed as a lattice problem under an asymmetric simplicial norm [28], the Integer Programming problem reduces to lattice problems under near-symmetric norms [27, 12, 11], etc.

Much is known about the computational complexity of SVP and CVP, in both their exact and approximation versions. On the negative side, SVP in ℓ_2 is NP-hard (under randomized reductions) to solve exactly, or even to approximate to within any constant factor [1, 9, 34, 29]. Many more hardness results are known for other ℓ_p norms and under stronger complexity assumptions than $\mathsf{P} \neq \mathsf{NP}$ [17, 14, 43, 22, 35]. CVP is NP-hard to approximate to within $n^{c/\log\log n}$ factors for some constant $c > 0$ [4, 15], where n is the dimension of the lattice. Therefore, we do not expect to solve (or even closely approximate) these problems efficiently in high dimensions. Still, algorithms providing weak approximations or having super-polynomial running times are the foundations for the many applications mentioned above.

Though the applications are often expressed using a variety of norms, the majority of the algorithmic work on SVP and CVP over the last quarter century has focused on the important case of the ℓ_2 norm. While there has been both tremendous practical and theoretical progress for ℓ_2-based solvers, progress for more general norms has been much slower (we overview this history below). Illustrative of this, for most of the problems mentioned above, the solution strategy has almost invariably been to approximate the problem via a reduction to ℓ_2. In many cases, the desired computational problem requires only a "coarse" approximate solution to the underlying lattice problem (e. g., where a $\mathrm{poly}(n)$ or even $2^{O(n)}$ factor approximation suffices), in which case approximation by ℓ_2 is often sufficient. In some cases, however, the errors induced by the ℓ_2 approximation can result in a substantial increase in worst case running time or yield unusable results. As an example, with respect to the Integer Programming Problem (IP), in a sequence of works Dadush, Peikert and Vempala [12, 10] worked directly with norms induced by the continuous relaxation—avoiding direct ellipsoidal approximations—to reduce the time complexity of solving an n-variable IP from $2^{O(n)}n^{2n}$ (the previous best using ℓ_2 techniques [24]) to $2^{O(n)}n^n$. From these considerations we see that the problem of developing effective algorithms for solving the SVP and CVP under general norms is well motivated.

The algorithmic history of the SVP and CVP is long and rich. We relate the broad outlines here, highlighting the pertinent developments for general norms, and refer the reader to [36, 20] for a more complete accounting. There are three main classes of methods for solving lattice problems: basis reduction, randomized sieving, and Voronoi cell-based search.

Basis reduction combines both local search on lattice bases and lattice point enumeration. The celebrated LLL basis reduction algorithm [31] and further extensions [6, 45] give $2^{n/\mathrm{polylog}(n)}$ approximations to SVP and CVP under ℓ_2 in $\mathrm{poly}(n)$ time. Variants of basis reduction for symmetric norms are explored in [33, 26] and give similar approximation guarantees for SVP (though not for CVP) as the ℓ_2 versions. However, bounds on the time complexity were only proved for fixed dimension (where the running time is polynomial). For exact SVP and CVP in the ℓ_2 norm, Kannan's algorithm and its subsequent improvements [27, 23, 21, 38] use basis reduction techniques to deterministically compute solutions in $2^{O(n\log n)}$ time and $\mathrm{poly}(n)$ space.

This performance remained essentially unchallenged until the breakthrough **randomized "sieving"** algorithm of Ajtai, Kumar and Sivakumar [2], which gave a $2^{O(n)}$-time and -space randomized algorithm for exact SVP under ℓ_2. The randomized sieving approach consists of sampling an exponential number of "perturbed" lattice points, and then iteratively clustering and combining them to give shorter and shorter lattice points. Subsequently, the randomized sieve was greatly extended to yield solutions for more general norms and for $(1+\varepsilon)$-CVP. For exact SVP, the randomized sieve was extended (in the same time complexity) to ℓ_p norms [7], symmetric norms [5] and to near-symmetric norms [11]. Here, an asymmetric norm with unit ball $K \subseteq \mathbb{R}^n$ is near-symmetric if $\mathrm{vol}_n(K) \leq 2^{O(n)}\,\mathrm{vol}_n(K \cap -K)$. For CVP, the randomized sieve was further used to give a $(2+1/\varepsilon)^{O(n)}$-time and -space algorithm for $(1+\varepsilon)$-CVP under the ℓ_2 norm [3, 7], ℓ_p norms [7] and near-symmetric norms [11]. We remark that near-symmetric norms appear naturally in the context of Integer Programming: the problem of finding a lattice point near the "center" of the continuous relaxation (which need not be symmetric) can be directly expressed as a CVP under a near-symmetric norm [11]. Crucially, for any convex body $K \subseteq \mathbb{R}^n$, one can find a point $\mathbf{x} \in K$ (e. g., the center of gravity) such that $\mathrm{vol}_n((K-\mathbf{x}) \cap (\mathbf{x}-K)) \geq 2^{-n}\,\mathrm{vol}_n(K)$ [39], i. e., such that $K-\mathbf{x}$ is near-symmetric. Lastly, for the specific case of ℓ_∞, Eisenbrand, Hähnle and Niemeier [16] show that $(1+\varepsilon)$-CVP can be solved using $O(\ln(1/\varepsilon))^n$ calls to any 2-approximate solver, via an elegant cube-covering technique. It is worth noting that AKS sieve-based algorithms are *Monte Carlo*: while they output correct solutions (i. e., a shortest or closest vector) with high probability, the correctness is not guaranteed.

In a major breakthrough, Micciancio and Voulgaris [37] gave a *deterministic* $2^{O(n)}$-time and -space algorithm for exact SVP and CVP under the ℓ_2 norm using the **Voronoi cell** of a lattice. The Voronoi cell, which is the symmetric polytope consisting of all points in space closer to the origin (under ℓ_2) than to any other lattice point, is represented algorithmically here by $O(2^n)$ lattice points corresponding to the facets of the Voronoi cell (these lattice points are known as Voronoi-relevant vectors). The relevant vectors form an "extended basis" for the lattice, which Micciancio and Voulgaris (MV) use to efficiently guide closest lattice point search. Though it is tempting to try and directly extend the MV techniques to other norms, this appears to be quite challenging. A major difficulty is that for more general norms the Voronoi cell need not be convex, and furthermore, no good bounds are known for the number of relevant vectors. In a subsequent work, however, Dadush, Peikert and Vempala [12] showed that MV lattice point search techniques can, in a qualified sense, be extended to general norms (in fact, to general convex bodies) by reducing lattice problems to enumeration within ellipsoids. Combining a technique for constructing "efficient" ellipsoid coverings—using the M-Ellipsoid concept from convex geometry—together with Voronoi cell-based search, they showed that the lattice points inside a convex body can be computed in time proportional to the maximum number of lattice points that any translation of the body can contain. With some further improvements [13, 10], the DPV lattice point enumeration technique was used to give the first deterministic $2^{O(n)}$-time and -space algorithms for SVP and α-Bounded Distance Decoding (α-BDD), for any fixed constant $\alpha > 0$, under near-symmetric norms.[1]

Given the recent progress for general norm lattice problems, a major open problem is whether there exists a $2^{O(n)}$-time algorithm for solving exact CVP under near-symmetric norms (which would also imply a $2^{O(n)}$-time algorithm for Integer Programming [11]). While this problem seems currently out of reach, a natural next question is whether one can improve the extant $(1+\varepsilon)$-approximation algorithms for near-symmetric norm CVP. In particular, does there exist a *deterministic* algorithm for $(1+\varepsilon)$-CVP achieving comparable asymptotic complexity to the randomized sieving algorithms?

1.1 Results and techniques

Building on the DPV lattice point enumeration techniques, we give the first *deterministic* algorithm for $(1+\varepsilon)$-CVP with *comparable asymptotic complexity* to the AKS randomized sieve. Our main theorem is stated as follows:

Theorem 1.1 (Approximate CVP in any norm, informal)**.** *There is a deterministic algorithm that, given any near-symmetric norm $\|\cdot\|_K$, n-dimensional lattice $\mathcal{L}$, target $\mathbf{x} \in \mathbb{R}^n$ and $0 < \varepsilon \leq 1$, computes $\mathbf{y} \in \mathcal{L}$, a $(1+\varepsilon)$-approximate minimizer of $\|\mathbf{y}-\mathbf{x}\|_K$ in $(1+1/\varepsilon)^n \cdot 2^{O(n)}$ time and $\tilde{O}(2^n)$ space.*

Compared to AKS, our approach also achieves a better dependence on ε, namely, $2^{O(n)}(1+1/\varepsilon)^n$ instead of $2^{O(n)}(1+1/\varepsilon)^{2n}$, and uses significantly less space, namely, $\tilde{O}(2^n)$ compared to $2^{O(n)}(1+1/\varepsilon)^n$. Additionally, as we will discuss below, continued progress for exact CVP under ℓ_2 could further reduce the space usage of the algorithm. We note, however, that the $2^{O(n)}$ factor in the running time is currently much larger than in AKS, though little effort has been spent in trying to calculate or optimize it. To explain our approach, we first present the main DPV enumeration algorithm in its most recent formulation [10].

Theorem 1.2 (Enumeration in Convex Bodies, informal)**.** *There is a deterministic algorithm that, given an n-dimensional bounded convex body K and an n-dimensional lattice $\mathcal{L}$, enumerates the elements of $K \cap \mathcal{L}$ in time $2^{O(n)}G(K,\mathcal{L})$ using $\tilde{O}(2^n)$ space, where $G(K,\mathcal{L}) = \max_{\mathbf{x}\in\mathbb{R}^n} |(K+\mathbf{x}) \cap \mathcal{L}|$. Furthermore, given an algorithm that solves exact CVP on n-dimensional lattices under ℓ_2 in $T(n)$ time and $S(n)$ space, $K \cap \mathcal{L}$ can be enumerated in $2^{O(n)}T(n)G(K,\mathcal{L})$ time using $S(n)+\mathrm{poly}(n)$ space.*

[1] α-BDD is the promise version of CVP where the distance to the target is guaranteed to be at most an α-factor times the length of the shortest non-zero vector of the lattice.

The above lattice point enumerator will form the core of our $(1+\varepsilon)$-CVP algorithm. As we will show, the space used by our approximate CVP algorithm will only be larger than that used for the enumeration algorithm by an additive polynomial term. Therefore, if one could develop an algorithm for exact CVP under ℓ_2 which runs in $2^{O(n)}$ time and $\mathrm{poly}(n)$ space, then the space usage of our $(1+\varepsilon)$-CVP algorithm would be reduced to $\mathrm{poly}(n)$ in the same time complexity. The possibility of such a solver is discussed in [37], and devising one remains an important open problem. We remark that by plugging in Kannan's algorithm for CVP under ℓ_2, we do indeed get a $\mathrm{poly}(n)$-space $(1+\varepsilon)$-CVP solver, though at the cost of an $n^{n/2}$ factor increase in running time.

Using the above enumerator as a black box, we now present the approach taken in [12] to solve CVP and explain the main problem that arises. Given the target $\mathbf{t} \in \mathbb{R}^n$, the algorithm first computes an initial coarse underestimate d_0 of the distance of $\mathbf{t}$ to $\mathcal{L}$ under $\|\cdot\|_K$ (using LLL, for example). For the next step, they use the lattice point enumerator to successively compute the sets $(\mathbf{t}+2^i d_0 K) \cap \mathcal{L}$ (i. e., all lattice points at distance at most $2^i d_0$ from $\mathbf{t}$), $i = 0, 1, 2, \ldots$, until a lattice point is found. Finally, a closest vector to $\mathbf{t}$ in the final enumerated set is returned.

From the description, it is relatively straightforward to show that the complexity of the algorithm is essentially $G(dK, \mathcal{L})$, where d is the distance of $\mathbf{t}$ to $\mathcal{L}$. The main problem with this approach is that, in general, one cannot a priori bound $G(dK, \mathcal{L})$; even in 2 dimensions and for the Euclidean norm this quantity can be arbitrarily large. The only generic setting where such a bound is indeed available is when the distance d of the target is bounded by $\alpha\lambda$, where λ is the length of a shortest non-zero lattice vector under $\|\cdot\|_K$. In this situation, we can bound $G(dK, \mathcal{L})$ by $2^{O(n)}(1+\alpha)^n$. We note that this bound depends crucially on K being near-symmetric, that is, one can construct examples where $G(dK, \mathcal{L})$ is arbitrarily large when K has no near-symmetry requirement. We remark further that solving CVP with this type of guarantee corresponds to α-BDD, and by a standard reduction it can be used to solve SVP (using $\alpha = 1$ above) in near-symmetric norms as well [19].

To circumvent the above problem, we propose the following simple solution. Instead of solving the CVP on the original lattice $\mathcal{L}$, we attempt to solve it on a sparser sublattice $\mathcal{L}' \subseteq \mathcal{L}$, where the distance of $\mathbf{t}$ to $\mathcal{L}'$ is not much larger than its distance to $\mathcal{L}$ (we settle for an approximate solution here) and where the maximum number of lattice points at the new target distance is appropriately bounded. Our main technical contribution is to show the existence of such "lattice sparsifiers," and to give a deterministic algorithm to compute them:

Theorem 1.3 (Lattice Sparsifier, informal)**.** *There is a deterministic algorithm that, given any n-dimensional lattice $\mathcal{L}$, any near-symmetric norm $\|\cdot\|_K$, and distance $t \geq 0$, computes a sublattice $\mathcal{L}' \subseteq \mathcal{L}$ in $2^{O(n)}$ time and $\tilde{O}(2^n)$ space satisfying: (1) the distance from any point in $\mathbb{R}^n$ to $\mathcal{L}'$ is at most t plus its distance to $\mathcal{L}$, (2) the number of points in $\mathcal{L}'$ at distance t from any point in $\mathbb{R}^n$ is at most $2^{O(n)}$.*

We shall call a sublattice $\mathcal{L}' \subseteq \mathcal{L}$ satisfying the conditions of Theorem 1.3 a (K,t)-sparsifier for $\mathcal{L}$, or simply a t-sparsifier for $\mathcal{L}$ when the context is clear.

Given such sparsifiers, efficiently solving $(1+\varepsilon)$-CVP is straightforward. We simply compute a sparsifier $\mathcal{L}'$ for $\mathcal{L}$ under $\|\cdot\|_K$ with $t = \varepsilon d_K(\mathbf{t}, \mathcal{L})$ (the distance from $\mathbf{t}$ to $\mathcal{L}$), and then solve the exact CVP on $\mathcal{L}'$ using the DPV algorithm. By the guarantees on the sparsifier, $\mathcal{L}'$ contains a point at distance at most $d_K(\mathbf{t}, \mathcal{L}) + \varepsilon d_K(\mathbf{t}, \mathcal{L}) = (1+\varepsilon) d_K(\mathbf{t}, \mathcal{L})$, and using a simple packing argument (see Lemma 2.3) we can show that

$$G((1+\varepsilon)dK,\mathcal{L}') = 2^{O(n)}\left(1+\frac{1}{\varepsilon}\right)^n G(\varepsilon dK,\mathcal{L}') = 2^{O(n)}\left(1+\frac{1}{\varepsilon}\right)^n.$$

Here we note that the correctness of the output follows from the distance-preserving properties of $\mathcal{L}'$, and the desired running time follows from the above bound on $G((1+\varepsilon)dK,\mathcal{L}')$.

Apart from enabling a deterministic algorithm, we believe that the use of lattice sparsifiers makes the mechanics of our algorithm far more transparent than the randomized sieving approaches. Indeed, it was in trying to understand why the randomized sieving methods work that we were led to the sparsifier concept.

Another benefit of our construction is that it helps elucidate an important qualitative difference between approximate and exact CVP under near-symmetric norms. In particular, our construction shows that one can often retain only a small fraction of the lattice without changing the distance to the target by much, while almost surely losing the exact closest vector. We note that it was shown in [11] that there is a randomized polynomial-time reduction from integer programming feasibility (deciding whether a convex body contains an integer point) to exact CVP under near-symmetric norms, and the current fastest integer programming algorithms have complexity $n^{O(n)}$ [27, 10].

Existence of sparsifiers. We note that it is far from obvious that lattice sparsifiers, as described in Theorem 1.3, even exist. Somewhat surprisingly, these good sparsifying sublattices are ubiquitous: one can show that a random sublattice of $\mathcal{L}$ chosen from a certain uniparametric family satisfies the conditions of Theorem 1.3 with lower bounded constant probability. Interestingly, this same family of random sublattices was used by Khot [30, 29] as a tool for creating hard SVP instances (in fact, he uses them to perform a "very weak" version of sparsification).

Let us assume that $K \subseteq \mathbb{R}^n$ is a symmetric convex body (the near symmetric case reduces to this by taking $K \cap -K$). To show the existence of sparsifiers for $\mathcal{L}$ and K, we shall essentially reduce to the following task: for any $t > 0$, find a sublattice $\mathcal{L}' \subseteq \mathcal{L}$ such that

1. the shortest non-zero vector of $\mathcal{L}'$ with respect to $\|\cdot\|_K$ has length $\geq t$,
2. every element of $\mathcal{L}$ is at distance at most αt from $\mathcal{L}'$, for some absolute constant $\alpha \geq 1$.

Let us now show that both these conditions imply that $\mathcal{L}'$ is a distance αt-sparsifier for $\mathcal{L}$.

Firstly, by the packing bound (see Lemma 2.3 part 1) and condition (1), the number of vectors in $\mathcal{L}'$ at distance αt from any point is $G(\alpha t K,\mathcal{L}) \leq (1+2\alpha)^n = 2^{O(n)}$.

Secondly, condition (2) ensures that for any vector $\mathbf{t}$ in $\mathbb{R}^n$, the distance from $\mathcal{L}'$ is at most an additive αt more than its distance to $\mathcal{L}$. To see this, let $\mathbf{x}$ be a closest vector to $\mathbf{t}$ in $\mathcal{L}$, and let $\mathbf{y}$ be a closest vector to $\mathbf{x}$ in $\mathcal{L}'$. By the triangle inequality,

$$\|\mathbf{y}-\mathbf{t}\|_K \leq \|\mathbf{y}-\mathbf{x}\|_K + \|\mathbf{x}-\mathbf{t}\|_K \leq \alpha t + \|\mathbf{x}-\mathbf{t}\|_K\,,$$

as claimed. Hence $\mathcal{L}'$ is a valid distance αt-sparsifier for $\mathcal{L}$.

We now outline two methods to construct $\mathcal{L}'$, beginning with a natural approach based on scaling a good basis, which presents difficulties, and a second approach based on random sublattices, which forms the heart of our algorithm.

Basis scaling. A first natural approach for building $\mathcal{L}'$ is to start with a "short" basis $\mathbf{b}_1, \dots, \mathbf{b}_n$ of $\mathcal{L}$, and let $\mathcal{L}'$ be generated by a suitable integer scaling of the basis vectors. Here we would like to integrally scale up the basis vectors (with a potentially different scaling for each basis element) as little as possible so as to ensure that $\mathcal{L}'$ contains no non-zero vectors of length at most t.

Unfortunately, it is not easy to accurately lower bound the length of the shortest vector of $\mathcal{L}'$ just by looking at one of its bases. The most useful lower bound on the length here, under the Euclidean norm, is the minimum norm of the Gram-Schmidt orthogonalization of the basis (which has a natural generalization to other norms). This value, however, can be much smaller than the minimum norm of the non-orthogonalized basis. In particular, if one integrally scales the starting basis so that all the Gram-Schmidt vectors have length at least t, then even starting from the "best" basis (say Hermite-Korkine-Zolotareff or Seysen reduced [47]), standard techniques only allow us to show a bound of $n^{O(\log n)}t$ on the distance of $\mathcal{L}$ to $\mathcal{L}'$, which is far too large for our application. We remark that even when $\mathcal{L}$ has an orthogonal basis and the norm is Euclidean, scaling the orthogonal basis can lead to $\mathcal{L}$ having points at distance $\Omega(\sqrt{n}t)$ from $\mathcal{L}'$.

Random sublattices. We now present the approach taken throughout the rest of the paper. For simplicity of presentation (and without loss of generality), we shall assume that $\mathcal{L} = \mathbb{Z}^n$. We recall that every full-rank sublattice $\mathcal{L}'$ of $\mathbb{Z}^n$ can be expressed as

$$\mathcal{L}' = \{\mathbf{x} \in \mathbb{Z}^n : A\mathbf{x} \equiv \mathbf{0} \pmod{m\mathbb{Z}^k}\}$$

where A is a $k \times n$ matrix, $k \le n$, with entries in $\mathbb{Z}_m$ (integers mod m), for some $k, m \in \mathbb{N}$. That is, every sublattice of $\mathbb{Z}^n$ is the solution set to a homogeneous system of modular equations. In a precise sense, finding a good sparsifying matrix A is in fact "dual" to the basis scaling approach.

To show that we can find an αt-sparsifier $\mathcal{L}'$, it will be helpful to rephrase conditions (1) and (2) in terms of the matrix A. Let $G_A = \{A\mathbf{x} \pmod{m\mathbb{Z}^k} : \mathbf{x} \in \mathbb{Z}^n\}$. Note that G_A is an abelian group of size at most m^k and that $\mathcal{L} \pmod{\mathcal{L}'}$ is isomorphic to G_A. In all cases of interest, the image of A will be $\mathbb{Z}_m^k$, in which case $G_A \simeq \mathbb{Z}_m^k$. For $\mathcal{L}'$ to be our desired sparsifier, we want $A \in \mathbb{Z}_m^{k\times n}$ such that:

1'. For all $\mathbf{x} \in \mathbb{Z}^n \setminus \{\mathbf{0}\}$, $\|\mathbf{x}\|_K \le t$, $A\mathbf{x} \not\equiv \mathbf{0} \pmod{m\mathbb{Z}^k}$.

2'. For all $\mathbf{c} \in G_A$, there exists $\mathbf{x} \in \mathbb{Z}^n$, $\|\mathbf{x}\|_K \le \alpha t$, such that $\mathbf{c} \equiv A\mathbf{x} \pmod{m\mathbb{Z}^k}$.

As mentioned previously, our choice for A will be random. At a high level, we will want a distribution over A for which the values $A\mathbf{x} \pmod{m\mathbb{Z}^k}$, $\mathbf{x} \in tK \cap \mathbb{Z}^n$, are "well spread." The most natural such distribution is to let the coefficients of A be chosen uniformly from $\mathbb{Z}_m$. With this choice, for any vector $\mathbf{x} \in \mathbb{Z}^n$ containing at least one coefficient relatively prime to m, the random vector $A\mathbf{x} \pmod{m\mathbb{Z}^k}$ is uniform in $\mathbb{Z}_m^k$. Hence, we should expect the map $h_A(\mathbf{x}) = A\mathbf{x} \pmod{m\mathbb{Z}^k}$ to act like a good hash function. With this in mind, we now outline why we can choose k and m so that conditions $(1')$ and $(2')$ are satisfied with constant probability.

Condition 1'. From the perspective of condition $(1')$, if the non-zero points in $tK \cap \mathbb{Z}^n$ are "generic enough" (i. e., having some coefficients relatively prime to m), then each of them gets mapped to $\mathbf{0}$ under h_A with probability m^{-k}. Therefore, as long as m^k (the size of the group $\mathbb{Z}_m^k$) is a constant factor larger than $|tK \cap \mathbb{Z}^n|$, by the union bound a random choice of A should ensure that $\mathcal{L}'$ contains no non-zero vectors of length $\leq t$ with constant probability.

To satisfy the genericity assumption, it suffices to pick m to be a prime $p \geq |tK \cap \mathbb{Z}^n|$. To see this, note that if a non-zero $\mathbf{x} \in tK \cap \mathbb{Z}^n$ does not have a relatively prime coefficients to p, then $\mathbf{x}/p \in \mathbb{Z}^n$ and by convexity $k\mathbf{x}/p \in tK \cap \mathbb{Z}^n$ for $0 \leq k \leq p$ (since $\mathbf{0} \in K$). But then $|tK \cap \mathbb{Z}^n| \geq p+1$, a contradiction. From here, since the probability of hitting $\mathbf{0}$ only depends on the size m^k, we may as well set $k = 1$ (we will see another reason for this later), letting A contain a single row.

Our choice of m, k is now reduced to choosing a suitable prime $p \geq |tK \cap \mathbb{Z}^n|$. We replace the matrix A with a vector $\mathbf{a}$ chosen uniformly from $\mathbb{Z}_p^n$, and we let $h_{\mathbf{a}}(\mathbf{x}) = \langle \mathbf{a}, \mathbf{x} \rangle \pmod{p}$.

Condition 2'. We now explain why condition $(2')$ can be made to hold simultaneously with $(1')$.

We recall our current choice of $k = 1$ and $m = p$ with p prime. With this choice, A is now a uniformly chosen vector $\mathbf{a} \in \mathbb{Z}_p^n$. For each $z \in \mathbb{Z}_p$, we associate the coset $\mathcal{L}'_z = \{\mathbf{x} \in \mathbb{Z}^n : h_{\mathbf{a}}(\mathbf{x}) \equiv z \pmod{p}\}$ of $\mathcal{L}'$, and define the length of $\mathcal{L}'_z$ to be $\min\{\|\mathbf{x}\|_K : \mathbf{x} \in \mathcal{L}'_z\}$. Note that $\mathcal{L}/\mathcal{L}' = \{\mathcal{L}'_z : z \in \mathbb{Z}_p\}$ as long as $\mathbf{a} \neq \mathbf{0}$. With this notation, our task is now to show that with constant probability, for a random $\mathbf{a} \in \mathbb{Z}_p^n$, every coset $\mathcal{L}'_z$, $z \in \mathbb{Z}_p$, has length at most αt, for $\alpha = O(1)$.

Note that if the size of the group $\mathbb{Z}_p$ is much larger than $|tK \cap \mathcal{L}|$, then there may not be enough elements in $\alpha tK \cap \mathbb{Z}_n$ for this to be possible. On the other hand, if $|\mathbb{Z}_p| < |tK \cap \mathcal{L}|$, then we guarantee the existence of short vectors in $\mathcal{L}'$ (of length at most $2t$) by the pigeonhole principle. Hence, for our construction to work it makes sense to choose $p = r|tK \cap \mathbb{Z}^n|$ for some $1 < r = O(1)$.

We note that by construction, the cosets of $\mathcal{L}'$ induced by $R = \{h_{\mathbf{a}}(\mathbf{x}) : \mathbf{x} \in tK \cap \mathbb{Z}^n\}$ have length at most t. The crucial observation is that if the cosets induced by R are short, then so are the cosets induced by $R + R = \{a + b : a, b \in R\}$, since by the triangle inequality they have length at most $2t$ (this is the last place the convexity of K is needed). Hence we can hope to show that if $|R|$ is large with respect to p, then we can cover all of $\mathbb{Z}_p$ with a constant number of sums of R, which would imply that all cosets $\mathcal{L}'_z$, $z \in \mathbb{Z}_p$, have length $O(t)$. Of course, if R was already a subgroup of $\mathbb{Z}_p$, then this process could not succeed. Crucially, $\mathbb{Z}_p$ has no non-trivial proper subgroups—this is the main reason to chose $k = 1$ and m prime—and hence this obstruction is avoided.

At this point, we rely on the well-known Cauchy-Davenport sumset inequality in additive combinatorics, which states for any subsets $C, D \subseteq \mathbb{Z}_p$ that $|C + D| \geq \min\{p, |C| + |D| - 1\}$. Since sumsets in $\mathbb{Z}_p$ grow at least linearly fast, given the previous observations, it suffices to show that $|R| = \Omega(p)$ with constant probability. Since $R = h_{\mathbf{a}}(tK \cap \mathbb{Z}^n)$ and p is chosen to be $\Theta(|tK \cap \mathbb{Z}^n|)$, we simply need to show that a random $h_{\mathbf{a}}$ is close to being injective on $tK \cap \mathbb{Z}^n$. This will follow from the fact that $h_{\mathbf{a}}$ is a good hash function. In particular, the probability that distinct elements $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_p^n$ collide under $h_{\mathbf{a}}$ (that is $h_{\mathbf{a}}(\mathbf{x}) = h_{\mathbf{a}}(\mathbf{y})$) is exactly $1/p$. Using a slight strengthening of the genericity argument for condition $(1')$, one can show that points in $tK \cap \mathbb{Z}^n$ are in fact all distinct when taken mod $p\mathbb{Z}^n$. Assuming this over all pairs of points in $tK \cap \mathbb{Z}^n$ the expected number of collisions is roughly $O(|tK \cap \mathbb{Z}^n|^2/p) = O(p)$. Therefore the average element in $tK \cap \mathbb{Z}^n$ collides with $O(p/|tK \cap \mathbb{Z}^n|) = O(1)$ other elements. From this, one can easily deduce that the map $h_{\mathbf{a}}$ can only shrink $tK \cap \mathbb{Z}^n$ by a constant factor, as needed. This completes the description of our randomized construction.

Derandomizing the construction. We claim in Theorem 1.3 to give a deterministic $2^{O(n)}$-time construction for lattice sparsifiers, whereas the above construction is clearly randomized.

To make the construction completely deterministic, we rely on the following main ideas. Firstly, instead of computing the sparsifier in one step, we compute a sequence of sparsifiers

$$\mathcal{L} = \mathcal{L}_0 \supseteq \mathcal{L}_1 \supseteq \cdots \supseteq \mathcal{L}_k\,,$$

with a geometric sparsification rate. This allows us to maintain that the sparsification distance at each step is (roughly speaking) at most a constant factor larger than the minimum distance (length of shortest non-zero vector) of the working lattice, and hence the number of lattice points the next sparsifier needs to remove at each step is bounded by $2^{O(n)}$.

Second, given a sparsification set $tK \cap \mathbb{Z}^n$ (we take any basis of $\mathcal{L}_i$ to rephrase the problem in terms of $\mathbb{Z}^n$) of size $2^{O(n)}$ as above, we need to find the sparsifying vector $\mathbf{a} \in \mathbb{Z}_p^n$, for $p = \Theta(|tK \cap \mathbb{Z}^n|)$, quickly and deterministically. Here, naive exhaustive search requires at least $p^n = 2^{O(n^2)}$ time, since $\mathbf{a}$ has n coordinates. Instead, we show that one can, roughly speaking, choose the coordinates of $\mathbf{a}$ one at a time (not in a fixed basis however) in $p^{O(1)}$ time, requiring $np^{O(1)}$ time for the whole vector.

We defer a detailed outline of our deterministic construction to Section 5.

Organization. In Section 3, we provide the exact reduction from $(1+\varepsilon)$-CVP to lattice sparsification, formalizing Theorem 1.1. In Section 4, we prove the existence of lattice sparsifiers using the probabilistic method. In Section 5, we give the derandomized lattice sparsifier construction, formalizing Theorem 1.3. Lastly, in Section 6, we discuss further applications and future directions.

2 Preliminaries

Convexity and norms. For sets $A, B \subseteq \mathbb{R}^n$, let $A + B = \{a + b : a \in A, b \in B\}$ denote their Minkowski sum. B_2^n denotes the n-dimensional Euclidean unit ball in $\mathbb{R}^n$. A convex body $K \subseteq \mathbb{R}^n$ is a full-dimensional compact convex set. A convex body K is $(\mathbf{a}_0, r, R)$-centered if $\mathbf{a}_0 + rB_2^n \subseteq K \subseteq \mathbf{a}_0 + RB_2^n$. For a convex body $K \subseteq \mathbb{R}^n$ containing $\mathbf{0}$ in its interior, we define the (possibly asymmetric) norm $\|\cdot\|_K$ induced by K as $\|\mathbf{x}\|_K = \inf\{s \geq 0 : \mathbf{x} \in sK\}$. For a $(\mathbf{0}, r, R)$-centered convex body K, we note that

$$\frac{1}{R}\|\mathbf{x}\|_2 \leq \|\mathbf{x}\|_K \leq \frac{1}{r}\|\mathbf{x}\|_2\,.$$

If K is symmetric (i. e., $K = -K$), then $\|\cdot\|_K$ is also symmetric ($\|\mathbf{x}\|_K = \|-\mathbf{x}\|_K$). A convex body K is γ-symmetric for $\gamma \in (0,1]$ if $\mathrm{vol}_n(K \cap -K) \geq \gamma^n \,\mathrm{vol}_n(K)$. We say that K is near-symmetric if it is $\Omega(1)$-symmetric.

Computational model. The convex bodies and norms will be presented to our algorithms via weak membership and distance oracles. For $\varepsilon \geq 0$ and a convex body $K \subseteq \mathbb{R}^n$, we define $K^{\varepsilon} = K + \varepsilon B_2^n$ and $K^{-\varepsilon} = \{\mathbf{x} \in K : \mathbf{x} + \varepsilon B_2^n \subseteq K\}$. A *weak membership oracle* O_K for K is a function which takes as input a point $\mathbf{x} \in \mathbb{Q}^n$ and rational $\varepsilon > 0$, and returns $O_K(\mathbf{x}, \varepsilon) = 1$ if $\mathbf{x} \in K^{-\varepsilon}$, 0 if $\mathbf{x} \notin K^{\varepsilon}$, and either 0 or 1 if $\mathbf{x} \in K^{\varepsilon} \setminus K^{-\varepsilon}$. A *weak distance oracle* $D_{K,\cdot}$ for K is a function that takes as input a point $\mathbf{x} \in \mathbb{Q}^n$ and rational $\varepsilon > 0$, and returns a rational number $D_{K,\varepsilon}(\mathbf{x})$ satisfying $|D_{K,\varepsilon}(\mathbf{x}) - \|\mathbf{x}\|_K| \leq \varepsilon \min\{1, \|\mathbf{x}\|_K\}$. All our oracles will come with a guarantee that their associated convex body K is $(\mathbf{a}_0, r, R)$-centered, for some

$\mathbf{a}_0 \in \mathbb{Q}^n$ and rational $0 < r \leq R$. The running time of our algorithms will be measured by the number of oracle calls and arithmetic operations, where we shall consider the bit description length of the oracle guarantees as part of the input. It is not hard to check that given centering guarantees a weak membership and distance oracle for K are polynomial time equivalent, hence we shall assume that we can convert between the two at no essential cost. For simplicity, we use the notation $\mathrm{poly}(\cdot)$ to denote a polynomial factor in all the relevant input parameters (dimension, encoding length of basis / oracle guarantees, etc.). In this paper, all arithmetic operations will occur on numbers of size polynomial in the input length.

Lattices. A full rank n-dimensional lattice $\mathcal{L} \subset \mathbb{R}^n$ is a full-dimensional discrete subgroup of $\mathbb{R}^n$. We will consider lattices of full rank in this paper. Equivalently, $\mathcal{L}$ can be expressed as $B\mathbb{Z}^n$, where $B \in \mathbb{R}^{n\times n}$ is a non-singular matrix, which we refer to as a basis for $\mathcal{L}$. The dual lattice of $\mathcal{L}$ is

$$\mathcal{L}^* = \{\mathbf{y} \in \mathbb{R}^n : \forall \mathbf{x} \in \mathcal{L}, \langle \mathbf{x}, \mathbf{y}\rangle \in \mathbb{Z}\},$$

which is generated by the basis B^{-T} (the inverse transpose of B). For a sublattice $M \subseteq \mathcal{L}$, we define the quotient group $\mathcal{L}/M = \{M + a : a \in \mathcal{L}\}$, corresponding to the cosets of $\mathcal{L}$ modulo M. We shall use the shorthand $\mathbb{Z}_p^n = \mathbb{Z}^n/p\mathbb{Z}^n$, for a natural number $p \in \mathbb{N}$.

We define the length of the shortest non-zero vector (minimum distance) of $\mathcal{L}$ under $\|\cdot\|_K$ by

$$\lambda_1(K, \mathcal{L}) = \min_{\mathbf{y}\in\mathcal{L}\setminus\{\mathbf{0}\}} \|\mathbf{y}\|_K .$$

We let

$$\mathrm{SVP}(K, \mathcal{L}) = \arg\min\nolimits_{\mathbf{z}\in\mathcal{L}\setminus\{\mathbf{0}\}} \|\mathbf{z}\|_K$$

denote the set of shortest non-zero vectors of $\mathcal{L}$ under $\|\cdot\|$. For $\mathbf{x} \in \mathbb{R}^n$, define the distance of $\mathbf{x}$ to $\mathcal{L}$ under $\|\cdot\|_K$ by

$$d_K(\mathcal{L}, \mathbf{x}) = \min_{\mathbf{y}\in\mathcal{L}} \|\mathbf{y} - \mathbf{x}\|_K .$$

We let

$$\mathrm{CVP}(K, \mathcal{L}, \mathbf{x}) = \arg\min\nolimits_{\mathbf{y}\in\mathcal{L}} \|\mathbf{y} - \mathbf{x}\|_K$$

denote the set of closest vectors to $\mathbf{x}$ in $\mathcal{L}$ under $\|\cdot\|_K$.

Covering numbers and lattice point bounds. For a set $A \subseteq \mathbb{R}^n$, let $\mathrm{int}(A)$ denote the interior of A. For convex bodies $A, B \subseteq \mathbb{R}^n$, we define the covering number

$$N(A, B) = \inf\{|\Lambda| : \Lambda \subseteq \mathbb{R}^n, A \subseteq \Lambda + B\},$$

i. e., the minimum number of translates of B needed to cover A. We will require the following standard inequalities on the covering numbers. We include proofs for completeness.

Lemma 2.1. *Let $A, B \subseteq \mathbb{R}^n$ be convex bodies, where B is symmetric. Then*

$$N(A, B) \leq \frac{\mathrm{vol}_n(A + B/2)}{\mathrm{vol}_n(B/2)} .$$

Proof. Let $T \subseteq A$ be any maximal set of points such that for all distinct $\mathbf{x}, \mathbf{y} \in T$, $(\mathbf{x}+B/2) \cap (\mathbf{y}+B/2) = \emptyset$. We claim that $A \subseteq T + B$. Note that by the maximality of T for any $\mathbf{z} \in A$ there exists $\mathbf{x} \in T$ such that $(\mathbf{z}+B/2) \cap (\mathbf{x}+B/2) \neq \emptyset$. Therefore $\mathbf{z} \in \mathbf{x}+B/2-B/2 = \mathbf{x}+B$, as needed.

Since $T+B/2$ corresponds to $|T|$ disjoint translates of $B/2$, we have that

$$|T|\,\mathrm{vol}_n(B/2) = \mathrm{vol}_n(T+B/2) \leq \mathrm{vol}_n(A+B/2)\,.$$

Rearranging the above inequality yields the lemma. □

Lemma 2.2. *For any convex body $K \subseteq \mathbb{R}^n$ and $d > 0$, we have $N(dK, K) \leq (4d+2)^n$.*

Proof. From the well-known relation

$$\begin{aligned}\int_K \mathrm{vol}_n((K-\mathbf{x}) \cap (\mathbf{x}-K))/\,\mathrm{vol}_n(K)\,\mathrm{dx} &= \int_K \int_K \mathbf{1}[2\mathbf{x}-\mathbf{y} \in K]/\,\mathrm{vol}_n(K)\,\mathrm{dy}\,\mathrm{dx} \\ &= \int_K \mathrm{vol}_n((K+\mathbf{y})/2)/\,\mathrm{vol}_n(K)\,\mathrm{dy} = 2^{-n}\,,\end{aligned}$$

there exists $\mathbf{x} \in K$ such that $K[\mathbf{x}] = (K-\mathbf{x}) \cap (\mathbf{x}-K)$ satisfies $\mathrm{vol}_n(K[\mathbf{x}])/\,\mathrm{vol}_n(K) \geq 2^{-n}$.

By Lemma 2.1, since $K[\mathbf{x}]$ is symmetric, we have

$$\begin{aligned}N(dK,K) \leq N(dK,K[\mathbf{x}]) &\leq \frac{\mathrm{vol}_n(dK+K[\mathbf{x}]/2)}{\mathrm{vol}_n(K[\mathbf{x}]/2)} \\ &\leq \frac{\mathrm{vol}_n((d+1/2)K)}{\mathrm{vol}_n(K[\mathbf{x}]/2)} = (2d+1)^n \frac{\mathrm{vol}_n(K)}{\mathrm{vol}_n(K[\mathbf{x}])} \leq (4d+2)^n\,,\end{aligned}$$

as needed. □

For a lattice $\mathcal{L}$ and a convex body K in $\mathbb{R}^n$, let $G(K,\mathcal{L})$ be the largest number of lattice points contained in any translate of K, that is

$$G(K,\mathcal{L}) = \max_{\mathbf{x} \in \mathbb{R}^n} |(K+\mathbf{x}) \cap \mathcal{L}|\,.$$

We will need the following bounds on $G(K,\mathcal{L})$ from [10]. We include a proof for completeness.

Lemma 2.3. *Let $K \subseteq \mathbb{R}^n$ denote a γ-symmetric convex body and let $\mathcal{L}$ denote an n-dimensional lattice. Then for $d > 0$ we have that*

1. $G(dK,\mathcal{L}) \leq \gamma^{-n}\left(1+\dfrac{2d}{\lambda_1(K \cap -K,\mathcal{L})}\right)^n$,
2. $G(dK,\mathcal{L}) \leq \gamma^{-n}(2d+1)^n \cdot |(K \cap -K) \cap \mathcal{L}|$,
3. $G(dK,\mathcal{L}) \leq N(dK,K)G(K,\mathcal{L}) \leq (4d+2)^n G(K,\mathcal{L})$.

Proof of 1. Let $s = (1/2)\lambda_1(K \cap -K, \mathcal{L})$. For $\mathbf{x} \in \mathcal{L}$, we examine

$$\mathbf{x} + \mathrm{int}(s(K \cap -K)) = \{\mathbf{z} \in \mathbb{R}^n : \|\mathbf{z} - \mathbf{x}\|_{K \cap -K} < s\}.$$

Now for $\mathbf{x}, \mathbf{y} \in \mathcal{L}$, $\mathbf{x} \neq \mathbf{y}$, we claim that

$$(\mathbf{x} + \mathrm{int}(s(K \cap -K))) \cap (\mathbf{y} + \mathrm{int}(s(K \cap -K))) = \emptyset. \tag{2.1}$$

Assume not, then there exists $\mathbf{z} \in \mathbb{R}^n$ such that $\|\mathbf{z} - \mathbf{x}\|_{K \cap -K}, \|\mathbf{z} - \mathbf{y}\|_{K \cap -K} < s$. Since $K \cap -K$ is symmetric, we note that $\|\mathbf{y} - \mathbf{z}\|_{K \cap -K} = \|\mathbf{z} - \mathbf{y}\|_{K \cap -K} < s$. But then we have that

$$\begin{aligned}\|\mathbf{y} - \mathbf{x}\|_{K \cap -K} &= \|\mathbf{y} - \mathbf{z} + \mathbf{z} - \mathbf{x}\|_{K \cap -K} \leq \|\mathbf{y} - \mathbf{z}\|_{K \cap -K} + \|\mathbf{z} - \mathbf{x}\|_{K \cap -K} \\ &< s + s = 2s = \lambda_1(K \cap -K, \mathcal{L}),\end{aligned}$$

a clear contradiction since $\mathbf{y} - \mathbf{x} \neq 0$.

Take $\mathbf{c} \in \mathbb{R}^n$. To bound $G(dK, \mathcal{L})$ we must bound $|(\mathbf{c} + dK) \cap \mathcal{L}|$. For $\mathbf{x} \in \mathbf{c} + dK$, we note that $\mathbf{x} + s(K \cap -K) \subseteq \mathbf{c} + (d + s)K$. Therefore,

$$\begin{aligned}\mathrm{vol}_n((d+s)K) = \mathrm{vol}_n(\mathbf{c} + (d+s)K) &\geq \mathrm{vol}_n\left(((\mathbf{c} + dK) \cap \mathcal{L}) + s(K \cap -K)\right) \\ &= |(\mathbf{c} + dK) \cap \mathcal{L}| \, \mathrm{vol}_n(s(K \cap -K))\end{aligned}$$

where the last equality follows from (2.1). Therefore, we have that

$$|(\mathbf{c} + dK) \cap \mathcal{L}| \leq \frac{\mathrm{vol}_n((d+s)K)}{\mathrm{vol}_n(s(K \cap -K))} = \left(\frac{d+s}{\gamma s}\right)^n = \gamma^{-n}\left(1 + \frac{2d}{\lambda_1(K \cap -K, \mathcal{L})}\right)^n,$$

as needed. □

Proof of 2. Examine $dK + \mathbf{c}$. Let $\mathbf{y}_1, \ldots, \mathbf{y}_N \in (dK + \mathbf{c}) \cap \mathcal{L}$ denote a maximal collection of points such that the translates

$$\mathbf{y}_i + \frac{1}{2}(K \cap -K), \qquad i \in [N],$$

are interior disjoint. We claim that

$$(dK + \mathbf{c}) \cap \mathcal{L} \subseteq \bigcup_{i=1}^{N} \mathbf{y}_i + (K \cap -K).$$

Take $\mathbf{z} \in (dK + \mathbf{c}) \cap \mathcal{L}$. Then by the construction of $\mathbf{y}_1, \ldots, \mathbf{y}_N$, there exists $i \in [N]$ such that

$$\left(\mathbf{z} + \frac{1}{2}(K \cap -K)\right) \cap \left(\mathbf{y}_i + \frac{1}{2}(K \cap -K)\right) \neq \emptyset,$$

which implies that $\mathbf{z} \in \mathbf{y}_i + (K \cap -K)$, as needed. Therefore

$$|(dK + \mathbf{c}) \cap \mathcal{L}| \leq \sum_{i=1}^{n} |(\mathbf{y}_i + (K \cap -K)) \cap \mathcal{L}| = N|(K \cap -K) \cap \mathcal{L}|.$$

Since K is γ-symmetric, we get that

$$N = \frac{\mathrm{vol}_n(\bigcup_{i=1}^n \mathbf{y}_i + \frac{1}{2}(K \cap -K))}{\mathrm{vol}_n(\frac{1}{2}(K \cap -K))} \leq 2^n \gamma^{-n} \frac{\mathrm{vol}_n(dK + \frac{1}{2}(K \cap -K))}{\mathrm{vol}_n(K)} \leq \gamma^{-n}(2d+1)^n \,,$$

as needed. Since the above bound holds for all $\mathbf{c} \in \mathbb{R}^n$, we get that

$$G(dK, \mathcal{L}) \leq \gamma^{-n}(2d+1)^n \cdot |(K \cap -K) \cap \mathcal{L}| \,,$$

as needed. □

Proof of 3. Choose $T \subseteq \mathbb{R}^n$ such that $dK \subseteq T + K$ and $|T| = N(dK, K)$. Then for any $\mathbf{c} \in \mathbb{R}^n$,

$$\begin{aligned}|(dK + \mathbf{c}) \cap \mathcal{L}| &\leq |(K + T + \mathbf{c}) \cap \mathcal{L}| \leq \sum_{\mathbf{y} \in T} |(K + \mathbf{y} + \mathbf{c}) \cap \mathcal{L}| \leq |T| \max_{\mathbf{y} \in T} |(K + \mathbf{y} + \mathbf{c}) \cap \mathcal{L}| \\ &\leq N(dK, K) G(K, \mathcal{L}) \leq (4d+2)^n G(K, \mathcal{L}) \,,\end{aligned}$$

as needed. □

Algorithms. We will need the following lattice point enumeration algorithm from [12, 10].

Theorem 2.4 (Algorithm Lattice-Enum($K, \mathcal{L}, \varepsilon$))**.** *Let $K \subseteq \mathbb{R}^n$ be a $(\mathbf{a}_0, r, R)$-centered convex body given by weak membership oracle O_K, let $\mathcal{L} \subseteq \mathbb{R}^n$ be an n-dimensional lattice with basis $B \in \mathbb{Q}^{n \times n}$ and let $\varepsilon > 0$. Then there is a deterministic algorithm that on inputs $K, \mathcal{L}, \varepsilon$ (lazily) outputs a set S satisfying*

$$K \cap \mathcal{L} \subseteq S \subseteq (K + \varepsilon B_2^n) \cap \mathcal{L}$$

in $G(K, \mathcal{L}) \cdot 2^{O(n)} \cdot \mathrm{poly}(\cdot)$ time using $2^n \,\mathrm{poly}(\cdot)$ space.

We will require the following SVP solver from [12, 10].

Theorem 2.5 (Algorithm Shortest-Vectors($K, \mathcal{L}, \varepsilon$))**.** *Let $K \subseteq \mathbb{R}^n$ be a $(\mathbf{a}_0, r, R)$-centered symmetric convex body given by a weak membership oracle O_K, and let $\mathcal{L} \subseteq \mathbb{R}^n$ be an n-dimensional lattice with basis $B \in \mathbb{Q}^{n \times n}$, and let $\varepsilon > 0$. Let $\lambda_1 = \lambda_1(K, \mathcal{L})$. Then there is an algorithm that on inputs $K, \mathcal{L}, \varepsilon$ outputs a set $S \subseteq \mathcal{L}$ satisfying*

$$\mathrm{SVP}(K, \mathcal{L}) \subseteq S \subseteq \{\mathbf{y} \in \mathcal{L} \setminus \{\mathbf{0}\} : \|\mathbf{y}\|_K \leq \lambda_1 + \varepsilon \min\{1, \lambda_1\}\} \tag{2.2}$$

in deterministic $2^{O(n)}\mathrm{poly}(\cdot)$ time and $2^n \,\mathrm{poly}(\cdot)$ space.

3 CVP via lattice sparsification

To start, we give a precise definition of a lattice sparsifier.

Definition 3.1 (Lattice Sparsifier)**.** Let $K \subseteq \mathbb{R}^n$ be a γ-symmetric convex body, $\mathcal{L}$ be an n-dimensional lattice and $t \geq 0$. A (K, t)-sparsifier for $\mathcal{L}$ is a sublattice $\mathcal{L}' \subseteq \mathcal{L}$ satisfying

1. $G(tK, \mathcal{L}') = \gamma^{-n} 2^{O(n)}$,
2. $\forall \mathbf{x} \in \mathbb{R}^n$, $d_K(\mathcal{L}', \mathbf{x}) \leq d_K(\mathcal{L}, \mathbf{x}) + t$.

For the above, we trivially have $d_K(\mathcal{L}', \mathbf{x}) \geq d_K(\mathcal{L}, \mathbf{x})$, and hence $\mathcal{L}'$ preserves distances up to *additive error* t. For $t < \lambda_1(K-K, \mathcal{L})$, it is relatively straightforward to check that $G(tK, \mathcal{L}) = 1$, and hence $\mathcal{L}$ is a (K,t)-sparsifier for itself. The above definition therefore only becomes interesting above $t > \lambda_1(K-K, \mathcal{L})$. Indeed, the sparsification distances used in our algorithms may be arbitrarily larger than this in general.

We now prove a simple equivalence between building a sparsifier for symmetric and asymmetric norms. This will be useful for our sparsifier construction.

Lemma 3.2. *Let K be a γ-symmetric convex body, $\mathcal{L}$ be an n-dimensional lattice, $\mathcal{L}' \subseteq \mathcal{L}$ be an n-dimensional sublattice and $t > 0$. If $\mathcal{L}'$ is a $(K \cap -K, t)$-sparsifier for $\mathcal{L}$ then it is also a (K,t)-sparsifier for $\mathcal{L}$.*

Proof. Let $\mathcal{L}' \subseteq \mathcal{L}$ be a $(K \cap -K, t)$-sparsifier. Since $K \cap -K$ is 1-symmetric, we have by definition that $G(t(K \cap -K), \mathcal{L}') = 2^{O(n)}$. By Lemma 2.1 and γ-symmetry of K, we have that

$$N(tK, t(K \cap -K)) = N(K, K \cap -K) \leq \frac{\mathrm{vol}_n(K + \frac{1}{2}(K \cap -K))}{\mathrm{vol}_n(\frac{1}{2}(K \cap -K)} \leq \frac{\mathrm{vol}_n(\frac{3}{2}K)}{\mathrm{vol}_n(\frac{1}{2}(K \cap -K))} \leq 3^n \gamma^{-n}\,.$$

Therefore

$$G(tK, \mathcal{L}') \leq N(tK, t(K \cap -K)) G(t(K \cap -K), \mathcal{L}') = (3^n \gamma^{-n}) 2^{O(n)} = \gamma^{-n} 2^{O(n)}\,,$$

as needed. Since $K \cap -K \subseteq K$, we note that $\|\mathbf{a}\|_K \leq \|\mathbf{a}\|_{K \cap -K}$ for all $\mathbf{a} \in \mathbb{R}^n$. Take $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{z} \in \mathrm{CVP}(K, \mathcal{L}, \mathbf{x})$. By the guarantee on $\mathcal{L}'$, there exists $\mathbf{y} \in \mathcal{L}'$ such that

$$\|\mathbf{y} - \mathbf{z}\|_{K \cap -K} \leq d_{K \cap -K}(\mathcal{L}, \mathbf{z}) + t = t\,,$$

where the last equality follows since $\mathbf{z} \in \mathcal{L}$. Using the triangle inequality, we have that

$$\|\mathbf{y} - \mathbf{x}\|_K \leq \|\mathbf{y} - \mathbf{z}\|_K + \|\mathbf{z} - \mathbf{x}\|_K \leq \|\mathbf{y} - \mathbf{z}\|_{K \cap -K} + d_K(\mathcal{L}, \mathbf{x}) \leq d_K(\mathcal{L}, \mathbf{x}) + t\,,$$

as needed. Therefore, $\mathcal{L}'$ is a (K,t)-sparsifier for $\mathcal{L}$ as claimed. □

The following theorem represents the formalization of our lattice sparsifier construction.

Theorem 3.3 (Algorithm Lattice-Sparsifier)**.** *Let $K \subseteq \mathbb{R}^n$ be a $(\mathbf{0}, r, R)$-centered and γ-symmetric convex body specified by a weak membership oracle O_K, and let $\mathcal{L}$ denote an n-dimensional lattice given by basis $B \in \mathbb{Q}^{n \times n}$. For $t \geq 0$, a basis $B' \in \mathbb{Q}^{n \times n}$ for a (K,t)-sparsifier $\mathcal{L}'$ for $\mathcal{L}$ can be computed in deterministic $2^{O(n)} \mathrm{poly}(\cdot)$ time and $2^n \mathrm{poly}(\cdot)$ space.*

The proof of the above theorem is the subject of Sections 4 and 5 (which give randomized and deterministic constructions, respectively). Using the above lattice sparsifier construction, we present the following simple algorithm (Algorithm 1) for $(1+\varepsilon)$-CVP, which formalizes the algorithm presented in the introduction.

We now sketch Algorithm 1 and its analysis, avoiding the technical details associated with distance oracles. To begin, the algorithm computes a lower bound d on $d_K(\mathcal{L}, \mathbf{x})$. It then successively doubles d until $\mathbf{x} + (1+\varepsilon/3)dK$ intersects an $(\varepsilon/3)d$-sparsifier $\mathcal{L}'$ for $\mathcal{L}$, which we check via enumeration. At this point, we return all closest vectors to $\mathbf{x}$ in $\mathcal{L}'$ using enumeration.

For the running time, every enumeration takes at most $2^{O(n)}(1/\varepsilon)^n\gamma^{-n}$ time since we only enumerate on appropriately sparsified lattices. Furthermore, it is clear that the enumeration at the last phase controls the complexity of algorithm. For the approximation guarantee, we first note that if $d \geq d_K(\mathcal{L}, \mathbf{x})$ the $(\varepsilon/3)d$-sparsifier $\mathcal{L}'$ contains a lattice vector at distance $(1+\varepsilon/3)d$ from $\mathbf{x}$, and hence the algorithm returns all closest vectors to $\mathbf{x}$ in $\mathcal{L}'$. In particular, we see that at the final iteration $d/2 \leq d_K(\mathcal{L}, \mathbf{x})$, since otherwise we would have terminated in the previous iteration. Thus at the final iteration, $\mathcal{L}'$ in fact contains a vector at distance $d_K(\mathcal{L}, \mathbf{x}) + \varepsilon d/3 < (1+\varepsilon)d_K(\mathcal{L}, \mathbf{x})$ from $\mathbf{x}$, as needed.

Algorithm 1 Approx-Closest-Vectors$(K, \mathcal{L}, \mathbf{x}, \varepsilon)$

Require: $(\mathbf{0}, r, R)$-centered convex body $K \subseteq \mathbb{R}^n$ with weak distance oracle D_K for $\|\cdot\|_K$, a basis $B \in \mathbb{Q}^{n\times n}$ for $\mathcal{L}$, target $\mathbf{x} \in \mathbb{Q}^n$, $0 < \varepsilon \leq 1$.

Ensure: a non-empty set $S \subseteq \{\mathbf{y} \in \mathcal{L} : \|\mathbf{y} - \mathbf{x}\|_K \leq (1+\varepsilon)d_K(\mathcal{L}, \mathbf{x})\}$.

```
if x ∈ L then return {x}
Compute z ∈ CVP(B_2^n, L, x) using the MV algorithm.
l ← ||z − x||_2/R; ε_0 ← (ε/9) min{1, l}.
d ← l/2; d̃_x ← ∞.
repeat
    d ← 2d.
    L' ← Lattice-Sparsifier(K, L, (ε/3)d).
    for all y ∈ Lattice-Enum((1 + ε/3)dK + x, L', rε_0) do
        d̃_x ← min{d̃_x, D_{K,ε_0}(y − x), (1 + ε/3)d + ε_0}.
until d̃_x < ∞
return Lattice-Enum((d̃_x + ε_0)K + x, L', rε_0)
```

Theorem 3.4. *Algorithm 1 (Approx-Closest-Vectors) is correct, and on inputs $K, \mathcal{L}, \mathbf{x}, \varepsilon$ (as above), where K is γ-symmetric, it runs in deterministic $2^{O(n)}\gamma^{-n}(1+1/\varepsilon)^n \operatorname{poly}(\cdot)$ time and $2^n \operatorname{poly}(\cdot)$ space.*

Proof. We separately establish correctness and running time.

Correctness: If $\mathbf{x} \in \mathcal{L}$ then we are clearly done. Next, since K is $(0, r, R)$-centered, we have that

$$\frac{\|\mathbf{y}\|_2}{R} \leq \|\mathbf{y}\|_K \leq \frac{\|\mathbf{y}\|_2}{r}$$

for all $\mathbf{y} \in \mathbb{R}^n$. Now take any $\tilde{\mathbf{z}} \in \mathrm{CVP}(K, \mathcal{L}, \mathbf{x})$ and $\mathbf{z} \in \mathrm{CVP}(B_2^n, \mathcal{L}, \mathbf{x})$ (as in the algorithm). Define $d_x = \|\tilde{\mathbf{z}} - \mathbf{x}\|_K$, where we note that $d_x = d_K(\mathcal{L}, \mathbf{x})$. As in the algorithm, let $l = \|\mathbf{z} - \mathbf{x}\|_2/R$. Now we see that

$$l = \frac{\|\mathbf{z} - \mathbf{x}\|_2}{R} \leq \frac{\|\tilde{\mathbf{z}} - \mathbf{x}\|_2}{R} \leq \|\tilde{\mathbf{z}} - \mathbf{x}\|_K = d_x \leq \|\mathbf{z} - \mathbf{x}\|_K \leq \frac{\|\mathbf{z} - \mathbf{x}\|_2}{r} = l\frac{R}{r}\,.$$

Therefore, $l \leq d_x \leq lR/r$.

Let d_f denote the value of d after the repeat loop terminates. We claim that

$$\frac{1}{2}d_f \leq d_x \leq \left(1+\frac{\varepsilon}{3}\right) d_f + \varepsilon_0 \,.$$

When the repeat loop terminates, we are guaranteed that the call to

$$\text{Lattice-Enum}\left(\left(1+\frac{\varepsilon}{3}\right) d_f K + \mathbf{x}, \mathcal{L}', r\varepsilon_0\right)$$

outputs a lattice vector in $\mathcal{L}'$ at distance at most $(1+\varepsilon/3)d_f + \varepsilon_0$ from $\mathbf{x}$ under $\|\cdot\|_K$. Since $\mathcal{L}' \subseteq \mathcal{L}$, we clearly have that $d_x \leq (1+\varepsilon/3)d_f + \varepsilon_0$ as needed.

If the repeat loop terminates after the first iteration, then $d_f = l \leq d_x$ and hence $(1/2)d_f < d_x$, as needed. If the loop iterates more than once, then for the sake of contradiction, assume that $(1/2)d_f > d_x$. Then in the before last iteration, the value of d is greater than d_x. Now we are guaranteed that Lattice-Sparsifier($K, \mathcal{L}, (\varepsilon/3)d$) returns a lattice $\mathcal{L}'$ satisfying

$$d_K(\mathcal{L}', \mathbf{x}) \leq d_K(\mathcal{L}, \mathbf{x}) + \frac{\varepsilon}{3} d \leq \left(1+\frac{\varepsilon}{3}\right) d \,.$$

But then the call to

$$\text{Lattice-Enum}\left(\left(1+\frac{\varepsilon}{3}\right) dK + \mathbf{x}, \mathcal{L}', r\varepsilon_0\right)$$

is guaranteed to return a lattice point, and hence the repeat loop terminates at this iteration, a clear contradiction. Hence $(1/2)d_f \leq d_x$, as needed.

Let $d'_x = d_K(\mathcal{L}', \mathbf{x})$, for $\mathcal{L}'$ at the end of the repeat loop. We now claim that $\tilde{d}_x$ (as in the algorithm) satisfies $d'_x - \varepsilon_0 \leq \tilde{d}_x \leq d'_x + \varepsilon_0$. We first note that $\tilde{d}_x = \min\{(1+\varepsilon/3)d_f + \varepsilon_0, D_{K,\varepsilon_0}(\mathbf{z}-\mathbf{x})\}$ for some $\mathbf{z} \in \mathcal{L}'$. By the guarantees on $D_{K,\varepsilon_0}(\cdot)$ and Lattice-Enum, we get that

$$\tilde{d}_x = \min\{(1+\varepsilon/3)d_f + \varepsilon_0, D_{K,\varepsilon_0}(\mathbf{z}-\mathbf{x})\} \geq \min\{d'_x, \|\mathbf{z}-\mathbf{x}\|_K - \varepsilon_0\} \geq d'_x - \varepsilon_0 \,,$$

as needed. For the second inequality, we examine two cases. First assume that

$$\text{Lattice-Enum}\left(\left(1+\frac{\varepsilon}{3}\right) d_f K + \mathbf{x}, \mathcal{L}', r\varepsilon_0\right)$$

outputs $\mathbf{z} \in \mathrm{CVP}(K, \mathcal{L}', \mathbf{x})$. Then $\tilde{d}_x \leq D_{K,\varepsilon_0}(\mathbf{z}-\mathbf{x}) \leq d'_x + \varepsilon_0$, as needed. If Lattice-Enum does not output any element of $\mathrm{CVP}(K, \mathcal{L}', \mathbf{x})$, we must have that $(1+\varepsilon/3)d_f < d'_x$ and hence

$$\tilde{d}_x \leq (1+\varepsilon/3)d_f + \varepsilon_0 < d'_x + \varepsilon_0 \,,$$

as needed.

By the construction of $\mathcal{L}'$, we also have that $d'_x \leq d_x + \varepsilon/3 d_f \leq (1+2\varepsilon/3)d_x$. Since $d'_x \leq \tilde{d}_x + \varepsilon_0$, we know that $((\tilde{d}_x + \varepsilon_0)K + \mathbf{x}) \cap \mathcal{L} \neq \emptyset$. Therefore we are guaranteed that the final call to Lattice-Enum($(\tilde{d}_x + \varepsilon_0)K + \mathbf{x}, \mathcal{L}', r\varepsilon_0$) outputs all the closest vectors of $\mathcal{L}'$ to $\mathbf{x}$. Finally, any vector $\mathbf{y}$ outputted during this call satisfies

$$\|\mathbf{y}-\mathbf{x}\|_K \leq \tilde{d}_x + 2\varepsilon_0 \leq d'_x + 3\varepsilon_0 \leq (1+2\varepsilon/3)d_x + (\varepsilon/3)l \leq (1+\varepsilon)d_x \,,$$

as needed.

Running time: We first bound the running time of each call to Lattice-Enum. Within the repeat loop, the calls to Lattice-Enum($(1+\varepsilon/3)dK+\mathbf{x},\mathcal{L}',r\varepsilon_0$) run in $2^{O(n)}G((1+\varepsilon/3)dK,\mathcal{L}')\mathrm{poly}(\cdot)$ time and $2^n\mathrm{poly}(\cdot)$ space. By Lemma 2.3 part 3, since $(1+\varepsilon/3)=t(\varepsilon/3)$ for $t=(3/\varepsilon+1)$, we have that

$$G((1+\varepsilon/3)dK,\mathcal{L}')\leq(4t+2)^nG((\varepsilon/3)dK,\mathcal{L}')=6^n(1+2/\varepsilon)^nG((\varepsilon/3)dK,\mathcal{L}')=2^{O(n)}\gamma^{-n}(1+1/\varepsilon)^n,$$

where the last inequality follows from $G((\varepsilon/3)dK,\mathcal{L}')=\gamma^{-n}2^{O(n)}$ by the guarantees on Lattice-Sparsifier. Next the final call to Lattice-Enum($(\tilde{d}_x+\varepsilon_0)K+\mathbf{x},\mathcal{L}',r\varepsilon_0$) runs in $2^{O(n)}G((\tilde{d}_x+\varepsilon_0)K,\mathcal{L}')\mathrm{poly}(\cdot)$ time and $2^n\mathrm{poly}(\cdot)$ space. Now note that $\varepsilon_0\leq(1/9)\varepsilon d_x$, and hence

$$(1+\varepsilon/3)d_f\geq d_x-\varepsilon_0\geq(1-\varepsilon/9)d_x\,.$$

From here we get that

$$d_f\geq\frac{1-\varepsilon/9}{1+\varepsilon/3}d_x\geq\frac{1-1/9}{1+1/3}d_x=2/3d_x\,.$$

Finally,

$$\tilde{d}_x+\varepsilon_0\leq(1+\varepsilon/3)d_f+2\varepsilon_0\leq(1+\varepsilon/3)d_f+2/9\varepsilon d_x\leq(1+2\varepsilon/3)d_f\,.$$

Therefore, since $(1+2\varepsilon/3)=t(\varepsilon/3)$ for $t=(2+3/\varepsilon)$, by Lemma 2.3 part 3 and the guarantees on Lattice-Sparsifier as above

$$\begin{aligned}G((\tilde{d}_x+\varepsilon_0)K,\mathcal{L}')&\leq G((1+2\varepsilon/3)d_fK,\mathcal{L}')\leq(4t+2)^nG((\varepsilon/3)d_fK,\mathcal{L}')\\&=(10+12/\varepsilon)^nG((\varepsilon/3)d_fK,\mathcal{L}')=2^{O(n)}\gamma^{-n}(1+1/\varepsilon)^n\,.\end{aligned}$$

Lastly, note that each call to Lattice-Sparsifier takes at most $2^{O(n)}\mathrm{poly}(\cdot)$ time and $2^n\mathrm{poly}(\cdot)$ space. Since the repeat loop iterates polynomially many times (i. e., at most $\log_2(2R/r)$), the total running time is $2^{O(n)}\gamma^{-n}(1+1/\varepsilon)^n\,\mathrm{poly}(\cdot)$ and the total space usage is $2^n\,\mathrm{poly}(\cdot)$, as needed. □

4 A simple randomized lattice sparsifier construction

We begin with an existence proof for lattice sparsifiers using the probabilistic method. We will use the Cauchy-Davenport sumset inequality and another lemma in number theory about prime gaps, a consequence of a theorem of Rosser and Schoenfeld [44, 40].[2]

Theorem 4.1. *Let $p\geq1$ be a prime. Then for any $C_1,\ldots,C_k\subseteq\mathbb{Z}_p$, we have that*

$$|C_1+\cdots+C_k|\geq\min\Big\{p,\sum_{i=1}^k|C_i|-k+1\Big\}$$

Lemma 4.2 (Prime Gap)**.** *For every $x>1000$ there exists a prime $p\in\mathbb{Z}$ satisfying $x<p<4x/3$.*

[2]The authors are indebted to János Pintz for finding these references.

Proof. We will use the bounds

$$\pi(x) > x/\ln(x) \quad \text{if} \quad x > 17\,, \quad \text{and} \qquad \pi(x) < 1.25506x/\ln(x) \quad \text{if} \quad x > 1\,,$$

where $\pi(x)$ denotes the number of primes less than x [44, 40]. If $x > 1000$ then

$$\pi(4x/3) > (4x/3)/\ln(4x/3) > 1.25506x/\ln(x) > \pi(x)\,,$$

and the lemma follows. □

As explained in the introduction, our sparsifiers will take the form of random sublattices. To express these sublattices, we introduce the following notation. For a lattice $\mathcal{L}$, prime number $p \in \mathbb{N}$ and $\mathbf{w} \in \mathcal{L}^*/p\mathcal{L}^*$, we define the sublattice

$$\mathcal{L}(\mathbf{w}, p) = \{\mathbf{y} \in \mathcal{L} : \langle \mathbf{y}, \mathbf{w} \rangle \equiv 0 \pmod{p}\}\,.$$

To see that this is well-defined, first note that the inner products between vectors in $\mathcal{L}$ and $\mathcal{L}^*$ are integer valued. Second, since $\langle \mathbf{y}, \mathbf{x}+\mathbf{z} \rangle = \langle \mathbf{y}, \mathbf{x} \rangle \pmod{p}$ for any $\mathbf{x} \in \mathcal{L}^*, \mathbf{z} \in p\mathcal{L}^*$, one may indeed view $\mathbf{w}$ above as an element of $\mathcal{L}^*/p\mathcal{L}^*$ (any representative yields the same sublattice). Furthermore, letting B^* denote a basis of $\mathcal{L}^*$ it is easy to check that $\mathcal{L}^*/p\mathcal{L}^* = B^*\mathbb{Z}_p^n$.

The following theorem shows how the algebraic properties of the sparsifying vector $\mathbf{w}$ influence the geometry of $\mathcal{L}(\mathbf{w}, p)$. It formalises the arguments given in the introduction.

Theorem 4.3. *Let $K \subseteq \mathbb{R}^n$ be a symmetric convex body, $\mathcal{L} \subseteq \mathbb{R}^n$ an n-dimensional lattice and $t \geq 0$. Let B^* denote a basis of $\mathcal{L}^*$ and $p \in \mathbb{N}$ be an odd prime. Pick $\mathbf{a} \in \mathbb{Z}_p^n$ and let $\mathbf{w} = B^*\mathbf{a}$.*

Define
$$\begin{aligned} S &= \{B^{*T}\mathbf{y} \pmod{p\mathbb{Z}^n} : \mathbf{y} \in \mathcal{L} \cap tK\}\,,\\ S_0 &= \{\mathbf{z} \in S : \langle \mathbf{a}, \mathbf{z} \rangle \equiv 0 \pmod{p}\}\,,\\ R &= \{\langle \mathbf{a}, \mathbf{z} \rangle : \mathbf{z} \in S\}\,. \end{aligned}$$

Then, the following holds:

1. *For $p > |\mathcal{L} \cap (1-1/p)tK|$,*
   - *(a) $|S| = |\mathcal{L} \cap tK|$,*
   - *(b) For $d > 0$, $G(dtK, \mathcal{L}(\mathbf{w}, p)) \leq (2d+1)^n |S_0|$;*
2. *$\forall \mathbf{x} \in \mathbb{R}^n$, $d_K(\mathcal{L}(\mathbf{w}, p), \mathbf{x}) \leq d_K(\mathcal{L}, \mathbf{x}) + \lceil (p-1)/(|R|-1) \rceil t$.*

Proof of 1. We first prove part (a). By the construction $|S| \leq |\mathcal{L} \cap tK|$. We only need to show that $|S| \geq |\mathcal{L} \cap tK|$. Assume this is not the case. Then there exists distinct $\mathbf{x}, \mathbf{y} \in \mathcal{L} \cap tK$ such that $B^{*T}\mathbf{x} \equiv B^{*T}\mathbf{y} \pmod{p\mathbb{Z}^n}$, hence $\mathbf{x} - \mathbf{y} \in p\mathcal{L}$. Since $\mathbf{x}, \mathbf{y} \in K$, by the symmetry of K we have that $\mathbf{x} - \mathbf{y} \in 2K \cap p\mathcal{L}$. Therefore, for every $k \in \mathbb{Z}$, where $|k|$ is at most $(p-1)/2$, we have that

$$(k/p)(\mathbf{x} - \mathbf{y}) \in \mathcal{L} \cap (p-1)/(2p)(2K) = \mathcal{L} \cap (1-1/p)K\,.$$

Since $\mathbf{x} \neq \mathbf{y}$, these elements are all distinct. Therefore $|\mathcal{L} \cap (1-1/p)K| \geq 1 + 2((p-1)/2) = p$, contradicting our assumption on p.

We prove part (b) now. To begin, we see that

$$\begin{aligned}|\mathcal{L}(\mathbf{w},p) \cap tK| &= |\{\mathbf{y} \in \mathcal{L} \cap tK : \langle \mathbf{w},\mathbf{y} \rangle \equiv 0 \pmod{p}\}| = |\{\mathbf{y} \in \mathcal{L} \cap tK : \langle \mathbf{a}, B^{*T}\mathbf{y} \rangle \equiv 0 \pmod{p}\}| \\ &= |\{\mathbf{z} \in S : \langle \mathbf{a},\mathbf{z} \rangle \equiv 0 \pmod{p}\}| = |S_0|\,.\end{aligned}$$

By Lemma 2.3 part 2 we have that

$$G(dtK, \mathcal{L}(\mathbf{w},p)) \leq (2d+1)^n |\mathcal{L}(\mathbf{w},p) \cap tK| = (2d+1)^n |S_0|\,,$$

as needed. □

Proof of 2. Take $\mathbf{x} \in \mathbb{R}^n$ and let $\mathbf{y} \in \mathcal{L}$ be a closest vector to $\mathbf{x}$, i. e., satisfying $d_K(\mathcal{L},\mathbf{x}) = \|\mathbf{y}-\mathbf{x}\|_K$. Let $l = \lceil (p-1)/(|R|-1) \rceil$. Since $R \subseteq \mathbb{Z}_p$, by Theorem 4.1

$$|\underbrace{R + \cdots + R}_{l \text{ times}}| \geq \min\{p, l(|R|-1)+1\} = p \quad \Rightarrow \quad \underbrace{R + \cdots + R}_{l \text{ times}} = \mathbb{Z}_p\,.$$

Therefore, by the construction of R there exist

$$\mathbf{y}_1, \ldots, \mathbf{y}_l \in tK \cap \mathcal{L} \text{ such that } \langle \mathbf{y},\mathbf{a} \rangle \equiv \sum_{i=1}^{l} \langle \mathbf{y}_i, \mathbf{a} \rangle \pmod{p} \Leftrightarrow \mathbf{y} - \sum_{i=1}^{l} \mathbf{y}_i \in \mathcal{L}(\mathbf{w},p)\,.$$

Let $\mathbf{z} = \mathbf{y} - \sum_{i=1}^{l} \mathbf{y}_i$. Now, by the triangle inequality and the symmetry of K we get that

$$\|\mathbf{z}-\mathbf{x}\|_K \leq \|\mathbf{y}-\mathbf{x}\|_K + \|\mathbf{z}-\mathbf{y}\|_K \leq d_K(\mathcal{L},\mathbf{x}) + \sum_{i=1}^{l} \|\mathbf{y}_i\|_K \leq d_K(\mathcal{L},\mathbf{x}) + lt\,,$$

as needed. □

The following lemma will show the existence of a vector $\mathbf{a}$ that can be used to build a good sparsifier.

Lemma 4.4. *Let p be a prime and $S \subseteq \mathbb{Z}_p^n$ satisfying $1000 < |S| < p < 4|S|/3$ and $\mathbf{0} \in S$. Then there exists $\mathbf{a} \in \mathbb{Z}_p^n$ satisfying*

1. $|\{\mathbf{y} \in S : \langle \mathbf{y},\mathbf{a} \rangle \equiv 0 \pmod{p}\}| \leq 6$,
2. $|\{\langle \mathbf{y},\mathbf{a} \rangle \pmod{p} : \mathbf{y} \in S\}| \geq \dfrac{p+2}{3}$.

Proof. Let $\mathbf{a}$ denote a uniform random vector in $\mathbb{Z}_p^n$. We will show that $\mathbf{a}$ satisfies both conditions (1) and (2) with non-zero probability. Let $E_i^{\mathbf{y}}$ denote the indicator of the event $\langle \mathbf{a},\mathbf{y} \rangle \equiv i$ for $\mathbf{y} \in S$ and $i \in \mathbb{Z}_p$.

Claim 4.5. $\mathrm{E}\left[\sum_{\mathbf{y} \in S \setminus \{\mathbf{0}\}} E_0^{\mathbf{y}}\right] = \dfrac{|S|-1}{p}$.

Proof of Claim 4.5. By the linearity of expectation it suffices to prove

$$\mathrm{E}[E_0^{\mathbf{y}}] = \Pr[\langle \mathbf{a}, \mathbf{y} \rangle \equiv 0] = \frac{1}{p}$$

for $\mathbf{y} \in S \setminus \{\mathbf{0}\}$. Since $\mathbf{y} \neq \mathbf{0}$, p is a prime and $\mathbf{a}$ is uniform in $\mathbb{Z}_p^n$, we have that $\langle \mathbf{a}, \mathbf{y} \rangle$ is distributed uniformly in $\mathbb{Z}_p$. Therefore $\Pr[\langle \mathbf{a}, \mathbf{y} \rangle \equiv 0] = 1/p$. $\square$

Claim 4.6. $\mathrm{E}\left[\sum_{\mathbf{x},\mathbf{y} \in S, \mathbf{x} \neq \mathbf{y}} E_0^{\mathbf{x}-\mathbf{y}}\right] = \dfrac{|S|^2 - |S|}{p}$.

Proof of Claim 4.6. If $\mathbf{x} \neq \mathbf{y}$ then $\mathrm{E}[E_0^{\mathbf{x}-\mathbf{y}}] = 1/p$. The Claim follows by the linearity of expectation. $\square$

Returning to the proof of Lemma 4.4, we will now choose the vector $\mathbf{a} \in \mathbb{Z}_p^n$. By Markov's inequality,

$$\Pr\left[\left|\{\mathbf{y} \in S \setminus \{\mathbf{0}\} : \langle \mathbf{a}, \mathbf{y} \rangle \equiv 0\}\right| < 6\right] \geq 1 - \frac{|S| - 1}{6p} > \frac{5}{6},$$

and

$$\Pr\left[\left|\{(\mathbf{x},\mathbf{y}) : \mathbf{x},\mathbf{y} \in S, \mathbf{x} \neq \mathbf{y}, \langle \mathbf{a}, \mathbf{x} \rangle \equiv \langle \mathbf{a}, \mathbf{y} \rangle\}\right| \leq \frac{6|S|}{5}\right] \geq 1 - \frac{5|S|^2 - 5|S|}{6|S|p} > \frac{1}{6}.$$

Hence there exists an $\mathbf{a}$ such that both events hold. The first condition of the lemma is easy to check:

$$|\{\mathbf{y} \in S : \langle \mathbf{y}, \mathbf{a} \rangle \equiv 0\}| = |\{\mathbf{y} \in S \setminus \{\mathbf{0}\} : \langle \mathbf{y}, \mathbf{a} \rangle \equiv 0\}| + 1 \leq 5 + 1 = 6.$$

Now we will prove the second condition using our assumption and the Cauchy-Schwarz inequality:

$$\begin{aligned}\frac{11|S|}{5} &\geq |\{(\mathbf{x},\mathbf{y}) : \mathbf{x},\mathbf{y} \in S, \mathbf{x} \neq \mathbf{y}, \langle \mathbf{a}, \mathbf{x} \rangle \equiv \langle \mathbf{a}, \mathbf{y} \rangle\}| + |S| = |\{(\mathbf{x},\mathbf{y}) : \mathbf{x},\mathbf{y} \in S, \langle \mathbf{a}, \mathbf{x} \rangle \equiv \langle \mathbf{a}, \mathbf{y} \rangle\}| \\ &= \sum_{z \in \mathbb{Z}_p} |\{\mathbf{y} \in S : \langle \mathbf{a}, \mathbf{y} \rangle \equiv z\}|^2 \geq |S|^2 / |\{\langle \mathbf{y}, \mathbf{a} \rangle \pmod{p} : \mathbf{y} \in S\}|.\end{aligned}$$

These yield

$$|\{\langle \mathbf{y}, \mathbf{a} \rangle \pmod{p} : \mathbf{y} \in S\}| > \frac{5|S|}{11} > \frac{15p}{44} > \frac{p+2}{3},$$

as needed. $\square$

We now give our first lattice sparsifier construction, which combines Theorems 4.3 and 4.4. While we state it for symmetric norms only, it directly extends to asymmetric norms (see Lemma 3.2).

Theorem 4.7. *Let $K \subseteq \mathbb{R}^n$ be a symmetric convex body, $\mathcal{L} \subseteq \mathbb{R}^n$ an n-dimensional lattice, and $t \geq 0$ a non-negative number. Then there exists a prime $p \in \mathbb{N}$ and $\mathbf{w} \in \mathcal{L}^*/p\mathcal{L}^*$ such that*

1. $G(tK, \mathcal{L}(\mathbf{w}, p)) \leq 1000 \cdot 7^n$,
2. $\forall \mathbf{x} \in \mathbb{R}^n$, $d_K(\mathcal{L}(\mathbf{w}, p), \mathbf{x}) \leq d_K(\mathcal{L}, \mathbf{x}) + t$.

Proof. Let $N = |\mathcal{L} \cap (t/3)K|$. If $N \leq 1000$, let $p = 3$ and $\mathbf{w} = 0$. For condition (1), by Lemma 2.3 part 2 we have that

$$G(tK, \mathcal{L}) \leq 7^n |\mathcal{L} \cap (t/3)K| \leq 1000 \cdot 7^n\,,$$

as needed. Since $\mathcal{L}(\mathbf{w}, p) = \mathcal{L}$, condition (2) trivially holds.

Now assume that $N > 1000$. Then by Lemma 4.2, there exists a prime $p \in \mathbb{N}$, such that $N < p < (4/3)N$. Let B^* be a basis of $\mathcal{L}^*$ and let

$$S = \{B^{*T}\mathbf{y} \pmod{p\mathbb{Z}^n} : \mathbf{y} \in \mathcal{L} \cap (t/3)K\}\,.$$

Then by Theorem 4.3 part 1a, we have that $|S| = N$. Therefore by Lemma 4.4, there exists $\mathbf{a} \in \mathbb{Z}_p^n$ such that

$$|\{\mathbf{z} \in S : \langle \mathbf{a}, \mathbf{z} \rangle \equiv 0 \pmod{p}\}| \leq 6 \quad \text{and} \quad |\{\langle \mathbf{a}, \mathbf{z} \rangle : \mathbf{z} \in S\}| \geq (p+2)/3\,.$$

Hence by Lemma 4.3 parts 1b and 2, setting $\mathbf{w} = B^*\mathbf{a}$, $\mathcal{L}(\mathbf{w}, p)$ satisfies condition (1) with bound $6 \cdot 7^n < 1000 \cdot 7^n$ and condition (2) with additive error $(t/3)\lceil (p-1)/((p+2)/3-1) \rceil = t$, as needed. ☐

5 Derandomizing the lattice sparsifier construction

We begin with a high level outline of the deterministic sparsifier construction. To recap, in the previous section, we build a (K, t)-sparsifier for $\mathcal{L}$ as follows:

1. Compute $N \leftarrow |tK \cap \mathcal{L}|$. If $N \leq 1000$ then return $\mathcal{L}' = \mathcal{L}$. Else find a prime p satisfying $N < p < 4N/3$.
2. Build a basis $B^* \in \mathbb{Q}^{n \times n}$ for $\mathcal{L}^*$ and compute $S \leftarrow \{B^{*T}\mathbf{y} \pmod{p} : \mathbf{y} \in tK \cap \mathcal{L}\}$.
3. Find a vector $\mathbf{a} \in \mathbb{Z}_p^n$ satisfying[3]

$$(a) \quad |\{\mathbf{y} \in S : \langle \mathbf{a}, \mathbf{y} \rangle \equiv 0 \pmod{p}\}| \leq 6\,, \qquad (b) \quad |\{\langle \mathbf{a}, \mathbf{y} \rangle : \mathbf{y} \in S\}| \geq \frac{p+2}{3}\,.$$

4. Return the sublattice $\mathcal{L}' = \{\mathbf{y} \in \mathcal{L} : \langle \mathbf{y}, B^*\mathbf{a} \rangle \equiv 0 \pmod{p}\}$.

To implement the above construction efficiently and deterministically, we must overcome two main obstacles. First, the construction of the vector $\mathbf{a}$ is probabilistic (see Lemma 4.4), and so we must replace this with an explicit deterministic construction. For our derandomization approach, we will use a dimension reduction strategy which will allow the vector $\mathbf{a}$ to be computed efficiently via exhaustive search. We describe this method in Section 5.1. Second, the number of lattice points N in $tK \cap \mathcal{L}$ could be very large (since we have no control on t), and hence we cannot hope to directly compute the set S efficiently via lattice point enumeration. To deal with this, we show that the sparsifier can be computed iteratively via a "chain" of sparsifiers. This will allow us to keep t bounded by a constant times the minimum distance of the lattice at every step, keeping the size of N bounded by $2^{O(n)}$ at every step. The details of this approach are presented in Section 5.2.

[3]For slightly worse parameters, it is easy to check that a random $\mathbf{a} \in \mathbb{Z}_p^n$ satisfies these with constant probability.

5.1 Deterministic construction of a good vector

We now outline our approach for deterministically implementing Lemma 4.4. As stated above, the main idea is to use a dimension reduction procedure which allows $\mathbf{a}$ to be computed efficiently via exhaustive enumeration (i. e., trying all possible $\mathbf{a}$). Let N and S be as in the description. Since $N < p < 4N/3$, we note that an exhaustive search over $\mathbb{Z}_p^n$ requires a search over $p^n \leq (4N/3)^n$ possibilities, and the validity check (i. e., conditions (a) and (b)) for any particular $\mathbf{a}$ can be implemented in $\mathrm{poly}(N)$ time by simple counting. Since the existence of the desired $\mathbf{a}$ depends only on $|S|$ and p (and not on n), if we can compute a linear projection $\pi : \mathbb{Z}_p^n \to \mathbb{Z}_p^{n-1}$ such that $|\pi(S)| = |S|$, then we can reduce the problem to finding a good $\mathbf{a} \in \mathbb{Z}_p^{n-1}$ for $\pi(S)$. Indeed, such a map π can be computed efficiently and deterministically as long as $n \geq 3$. To see this, we first identify full rank $n-1$ dimensional projections with their kernels, i. e., lines through the origin in $\mathbb{Z}_p^n$. From here, we note that distinct elements $\mathbf{x}, \mathbf{y} \in S$ collide under the projection induced by a line l iff $\mathbf{x} - \mathbf{y} \in l$. Since the total number of lines spanned by differences of elements in S is at most $\binom{|S|}{2} < \binom{p}{2}$, as long as there are at least $\binom{p}{2}$ lines in $\mathbb{Z}_p^n$ (for $n \geq 3$) we can compute the desired projection. Therefore, repeating the process $n-2$ times, we are left with finding a good $\mathbf{a} \in \mathbb{Z}_p^2$, which we can do by trying all $p + 1 < (4N/3) + 1$ lines in $\mathbb{Z}_p^2$.

In the next section, we will show how to guarantee that $N = 2^{O(n)}$, and hence the entire construction described above will be implementable in $2^{O(n)}$ time and space.

For a linear subspace $S \subseteq \mathbb{Z}_p^n$, we define its orthogonal complement

$$S^{\perp} = \{\mathbf{y} \in \mathbb{Z}_p^n : \langle \mathbf{x}, \mathbf{y} \rangle \equiv 0 \pmod{p}, \forall\, \mathbf{x} \in S\}\,.$$

Since p is prime, we have the equality $(S^{\perp})^{\perp} = S$. Let $\mathrm{Lines}(\mathbb{Z}_p^n)$ denote the set of lines in $\mathbb{Z}_p^n$, i. e., 1-dimensional linear subspaces. For a vector $\mathbf{q} \in \mathbb{Z}_p^n$ we denote the orthogonal complement to its linear span $\mathbf{q}\mathbb{Z}_p$ by $\mathbf{q}^{\perp} = \{\mathbf{y} \in \mathbb{Z}_p^n : \langle \mathbf{q}, \mathbf{y} \rangle \equiv 0 \pmod{p}\}$.

The following pseudocode implements the algorithm described above.

For the desired application of the algorithm given below, the set S above will in fact be represented implicitly. Here the main access methodology we will require from S is a way to iterate over its elements. In the context of $(1+\varepsilon)$-CVP, the enumeration method over S will correspond to the Lattice-Enum algorithm. Here we state the guarantees of the algorithm abstractly in terms of the number of iterations required over S.

Theorem 5.1. *Algorithm 2 is correct, and performs* $\mathrm{poly}(n, \log p) p^4$ *arithmetic operations and* $O(np^3)$ *iterations over the elements of S. Furthermore, the space usage (not counting the space needed to iterate over S) is* $\mathrm{poly}(n, \log p)$.

Analysis of Good-Vector.

Correctness: We must show that the output vector $\mathbf{a}$ satisfies the guarantees of Lemma 4.4:

1. $|\{\mathbf{y} \in S : \langle \mathbf{a}, \mathbf{y} \rangle \equiv 0 \pmod{p}\}| \leq 6$.

2. $|\{\langle \mathbf{a}, \mathbf{y} \rangle \pmod{p} : \mathbf{y} \in S\}| \geq \dfrac{p+2}{3}$.

Algorithm 2 Algorithm Good-Vector(S, p)

Require: $S \subseteq \mathbb{Z}_p^n$, $\mathbf{0} \in S$, integer $n \geq 1$, a prime p satisfying $1000 < |S| < p < 4|S|/3$.
Ensure: $\mathbf{a} \in \mathbb{Z}_p^n$ satisfying the conditions of Lemma 4.4 .

1: **if** $n = 1$, **return** 1
2: $P \leftarrow I_n$ ($n \times n$ identity).
3: **for** n_0 **in** n **to** 3 **do**
4:     **for all** $\mathbf{q} \in \mathrm{Lines}(\mathbb{Z}_p^{n_0})$ **do**
5:         Compute a basis $B \in \mathbb{Z}_p^{n_0 \times (n_0-1)}$ for $\mathbf{q}^{\perp}$, i.e. satisfying $\mathbf{q}^{\perp} = B\mathbb{Z}_p^{(n_0-1)}$.
6:         For all distinct $\mathbf{x}, \mathbf{y} \in PS$ check that $B^T\mathbf{x} \not\equiv B^T\mathbf{y} \pmod{p\mathbb{Z}^{n_0-1}}$.
            If no collisions, set $P \leftarrow B^T P$ and exit loop; otherwise, continue.
7: **for all** $\mathbf{q} \in \mathrm{Lines}(\mathbb{Z}_p^2)$ **do**
8:     Pick $\mathbf{a} \in \mathbf{q} \setminus \{\mathbf{0}\}$.
9:     Compute zeros $\leftarrow |\{\mathbf{y} \in PS : \langle \mathbf{a}, \mathbf{y} \rangle \equiv 0 \pmod{p}\}|$.
10:     Compute distinct $\leftarrow |\{\langle \mathbf{a}, \mathbf{y} \rangle \pmod{p} : \mathbf{y} \in PS\}|$.
11:     **if** zeros ≤ 6 and distinct $\geq (p+2)/3$ **then**
12:         **return** $P^t\mathbf{a}$

If $n = 1$ then setting $\mathbf{a} \in \mathbb{Z}_p$ to 1 (i. e., line 1) trivially satisfies both conditions. We assume $n \geq 2$. We prove the following invariant for the first loop (line 2): at the beginning of each iteration, $P \in \mathbb{Z}_p^{n_0 \times n}$ and $|PS| = |S|$.

First let us assume that during the loop iteration, we find $B \in \mathbb{Z}_p^{n_0 \times (n_0-1)}$ satisfying $B^T\mathbf{x} \neq B^T\mathbf{y}$ for all distinct $\mathbf{x}, \mathbf{y} \in PS$ (verified in line 5). This yields that the map $\mathbf{x} \to B^T\mathbf{x}$ is injective when restricted to PS, and hence $|B^T PS| = |S|$. Next, since $B \in \mathbb{Z}_p^{n_0 \times (n_0-1)}$ and $P \in \mathbb{Z}_p^{n_0 \times n}$, we have that P is set to $B^T P \in \mathbb{Z}_p^{(n_0-1) \times n}$ for the next iteration, as needed.

Now we show that a valid projection matrix B^T is guaranteed to exist as long as $n_0 \geq 3$. First, we claim that there exists $\mathbf{q} \in \mathrm{Lines}(\mathbb{Z}_p^{n_0})$, such that for all distinct $\mathbf{x}, \mathbf{y} \in PS$, $(\mathbf{q}+\mathbf{x}) \cap (\mathbf{q}+\mathbf{y}) = \emptyset$, i.e. all the lines passing through PS in the direction $\mathbf{q}$ are disjoint. A line $\mathbf{q}$ fails to satisfy this condition if and only if $\mathbf{x} - \mathbf{y} \in \mathbf{q}$ for distinct $\mathbf{x}, \mathbf{y} \in PS$. The number of lines that can be generated in this way from PS is at most

$$\binom{|PS|}{2} = \binom{|S|}{2} < p(p-1)/2\,.$$

Since

$$|\mathrm{Lines}(\mathbb{Z}_p^{n_0})| = \frac{p^{n_0}-1}{p-1} > \frac{p(p-1)}{2}$$

for $n_0 \geq 3$ we may pick $\mathbf{q} \in \mathrm{Lines}(\mathbb{Z}_p^n)$ that satisfies the condition. Now let $B \in \mathbb{Z}_p^{n_0 \times (n_0-1)}$ denote a basis satisfying $\mathbf{q}^{\perp} = B\mathbb{Z}_p^{n_0-1}$. We claim that $|B^T PS| = |PS|$. Assume not then there exists distinct $\mathbf{x}, \mathbf{y} \in PS$ such that

$$B^T\mathbf{x} \equiv B^T\mathbf{y} \quad \Leftrightarrow \quad B^T(\mathbf{x}-\mathbf{y}) \equiv \mathbf{0} \quad \Leftrightarrow \quad (\mathbf{x}-\mathbf{y}) \in (B\mathbb{Z}_p^{n_0-1})^{\perp} = \mathbf{q}\,,$$

which contradicts our assumption on $\mathbf{q}$. Therefore, the algorithm is indeed guaranteed to find a valid projection, as needed.

After the first for loop, we have constructed $P \in \mathbb{Z}_p^{2\times n}$ satisfying $|PS| = |S|$, where $|S| < p < 4|S|/3$. By Lemma 4.4, there exists $\mathbf{a} \in \mathbb{Z}_p^2$ satisfying (1) and (2) for the set PS. Since (1) and (2) hold for any non-zero multiple of $\mathbf{a}$, i. e., any vector defining the same line as $\mathbf{a}$, we may restrict the search to elements of $\mathrm{Lines}(\mathbb{Z}_p^2)$. Therefore, by trying all $p+1$ elements of $\mathrm{Lines}(\mathbb{Z}_p^2)$ the algorithm is guaranteed to find a valid $\mathbf{a}$ for the PS. Noting that $\langle \mathbf{a}, P\mathbf{y}\rangle \equiv \left\langle P^T\mathbf{a}, \mathbf{y}\right\rangle$, we get that $P^T\mathbf{a}$ satisfies (1) and (2) for the set S, as needed.

Running time: For $n = 1$ the running time is constant. We assume $n \geq 2$. Here the first for loop is executed $n-2$ times. For each loop iteration we run though $\mathbf{q} \in \mathrm{Lines}(\mathbb{Z}_p^{n_0})$ until we find one inducing a good projection matrix B. From the above analysis, we iterate through at most $\binom{|S|}{2} < p(p-1)/2$ elements $\mathbf{q} \in \mathrm{Lines}(\mathbb{Z}_p^{n_0})$ before finding a good projection matrix. For each $\mathbf{q}$, we build a basis matrix B for $\mathbf{q}^\perp$ which can be done using $\mathrm{poly}(n, \log p)$ arithmetic operations. Next, we check for collisions against each pair $\mathbf{x}, \mathbf{y} \in PS$, which can be done using $O(|S|) = O(p)$ iterations over S. Therefore, at each loop iteration we enumerate over S at most p^3 times while performing only polynomial-time computations. Hence, the total number of operations (excluding the time needed to output the elements of S) is at most $\mathrm{poly}(n, \log p)p^4$.

For the last phase, we run through the elements in $\mathrm{Lines}(\mathbb{Z}_p^2)$, where $|\mathrm{Lines}(\mathbb{Z}_p^2)| = p+1$. The validity check for $\mathbf{a} \in \mathrm{Lines}(\mathbb{Z}_p^2)$ requires computing both the quantities (1) and (2). To compute $|\{\mathbf{y} \in S : \langle \mathbf{y}, \mathbf{a}\rangle \equiv 0 \pmod p\}|$ we iterate once over the set S and count how many zero dot products there are. To compute $|\{\langle \mathbf{a}, \mathbf{y}\rangle : \mathbf{y} \in S\}|$, we first iterate over all residues in $\mathbb{Z}_p$. Next, for each residue $i \in \mathbb{Z}_p$, if we find $\mathbf{y} \in S$ satisfying $\langle \mathbf{a}, \mathbf{y}\rangle \equiv i \pmod p$, we increment our counter by one, and otherwise continue. Hence for any specific $\mathbf{a} \in \mathbb{Z}_p^2$, we iterate over the set S exactly $p+1$ times, performing $\mathrm{poly}(n, \log p)p^2$ operations. Hence, over the whole loop we perform $O(p^2)$ iterations over the set S, and perform $\mathrm{poly}(n, \log p)p^3$ operations.

Therefore, over the whole algorithm we iterate over the set S at most np^3 times, and perform at most $\mathrm{poly}(n, \log p)p^4$ operations. Furthermore, not counting the space needed to iterate over the set S, the space used by the algorithm is $\mathrm{poly}(n, \log p)$. □

5.2 Deterministic sparsifier construction

We now detail how to iteratively use the Good-Vector algorithm to get a completely deterministic lattice sparsifier construction.

To assure that we never need to apply the Good-Vector algorithm to a set S of size larger than $2^{O(n)}$, we will build the (K,t)-sparsifier iteratively. In particular, we will compute a sequence of sparsifiers $\mathcal{L}_1, \ldots, \mathcal{L}_k$, satisfying that $\mathcal{L}_i$ is a $(K, c^i\lambda)$-sparsifier for $\mathcal{L}_{i-1}$ for $i \geq 1$, where $\mathcal{L}_0 = \mathcal{L}$, $\lambda = \lambda_1(K, \mathcal{L})$ and $c > 1$ is a constant. Here we start the sparsification process at the minimum distance of $\mathcal{L}$ and increase the sparsification distance by a constant factor at each step. To build the sparsifier $\mathcal{L}_i$, $i \geq 1$, Good-Vector will (roughly speaking) only need to enumerate from the set $\mathcal{L}_{i-1} \cap c^i\lambda K$ which contains at most

$$G(c^i\lambda K, \mathcal{L}_{i-1}) \leq (4c+2)^n G(c^{i-1}\lambda K, \mathcal{L}_{i-1}) = 2^{O(n)}$$

points (by Lemma 2.3 part 3), since $\mathcal{L}_{i-1}$ is a $(K, c^{i-1}\lambda)$-sparsifier. Hence we will be able to guarantee that the number of lattice points we process at each step is $2^{O(n)}$. Furthermore, the geometric growth rate in the sparsification distance will allow us to conclude that $\mathcal{L}_i$ is in fact a $(K, (c^{i+1}/(c-1))\lambda)$-sparsifier for $\mathcal{L}$. Hence, iterating the process $O(\ln(t/\lambda_1))$ times will yield the final desired sparsifier.

Algorithm 3 Algorithm Lattice-Sparsifier(K, $\mathcal{L}$, t)

Require: $(\mathbf{0}, r, R)$-centered convex body $K \subseteq \mathbb{R}^n$ with distance oracle $D_{K,\cdot}$ for $\|\cdot\|_K$, basis $B \in \mathbb{Q}^{n\times n}$ for $\mathcal{L}$, and $t \geq 0$.

Ensure: (K,t)-sparsifier for $\mathcal{L}$.

1: $K \leftarrow K \cap -K$.
2: Compute $\mathbf{y} \in$ Shortest-Vectors$(K, \mathcal{L}, 1/3)$.
3: $\lambda \leftarrow D_{K,1/2}(\mathbf{y})$; $\varepsilon \leftarrow 7^{-(n+5)}$.
4: $k \leftarrow \lfloor \ln\left(\frac{2}{3}\frac{t}{\lambda}+1\right)/\ln 3 \rfloor$.
5: $\mathcal{L}_0 \leftarrow \mathcal{L}; B_0 \leftarrow B$.
6: **for** i **in** 0 **to** $k-1$ **do**
7:     $\bar{S} \leftarrow$ Lattice-Enum$(3^i(1-\varepsilon)\lambda K, \mathcal{L}_i, \varepsilon\lambda r)$.
8:     Compute $N \leftarrow |\bar{S}|$.
9:     **if** $N > 1000$ **then**
10:         Compute $B_i^* \leftarrow B_i^{-T}$, a basis for $\mathcal{L}_i^*$.
11:         Compute prime p satisfying $N < p < 4N/3$.
12:         $\mathbf{a} \leftarrow$ Good-Vector$(B_i^{*T}\bar{S} \pmod{p\mathbb{Z}^n}, p)$.
13:         Compute $\mathcal{L}_{i+1} \leftarrow \{\mathbf{y} \in \mathcal{L}_i : \langle B_i^*\mathbf{a}, \mathbf{y}\rangle \equiv 0 \pmod{p}\}$ and basis B_{i+1} for $\mathcal{L}_{i+1}$.
14:     **else**
15:         $\mathcal{L}_{i+1} \leftarrow \mathcal{L}_i; B_{i+1} \leftarrow B_i$.
16: **return** $\mathcal{L}_k$

The pseudocode in Algorithm 3 implements the above algorithm. The correctness and running time bound of this algorithm will yield the proof of Theorem 3.3. We begin its analysis with the following lemma. We remark that the lemma works on $K \cap -K$, the symmetrized version of K.

Lemma 5.2. *Let i denote the iteration of the for loop at line* 6*, and let the variables be defined as in each iteration. For $i \in \{0, \ldots, k\}$, the following holds:*

1. $G(3^i\lambda K, \mathcal{L}_i) \leq 7^{n+4}$;

2. $\forall \mathbf{x} \in \mathbb{R}^n$, $d_K(\mathcal{L}_i, \mathbf{x}) \leq d_K(\mathcal{L}, \mathbf{x}) + \dfrac{3}{2}(3^i - 1)\lambda$.

Proof. We first claim that λ satisfies $\lambda_1(K,\mathcal{L})/2 \leq \lambda \leq 2\lambda_1(K,\mathcal{L})$. Here, we have that

$$\|\mathbf{y}\|_K \leq \frac{4}{3}\lambda_1(K,\mathcal{L})$$

by the guarantee on Shortest-Vector$(K,\mathcal{L},1/3)$. By the guarantees on $D_{K,1/2}$, we have that $\lambda = D_{K,1/2}(\mathbf{y})$ satisfies

$$\lambda_1(K,\mathcal{L})/2 \leq \|\mathbf{y}\|_K/2 \leq \lambda \leq \frac{3}{2}\|\mathbf{y}\|_K \leq 2\lambda_1(K,\mathcal{L}),$$

as needed.

We now establish the lemma by induction on i. For $i = 0$, we have that $\mathcal{L}_0 = \mathcal{L}$. Since $\lambda \leq 2\lambda_1(K,\mathcal{L})$, by Lemma 2.3 part 1 we have that $G(\lambda K, \mathcal{L}_0) \leq (2 \cdot 2 + 1)^n = 5^n < 7^{n+4}$. Hence $\mathcal{L}_0$ satisfies (2). Furthermore, since $\mathcal{L}_0 = \mathcal{L}$, we have that $\mathcal{L}_0$ trivially satisfies property (2).

We now prove the claim for $i \geq 1$. Let $\bar{S}$ denote the set outputted by

$$\text{Lattice-Enum}(3^{i-1}(1-\varepsilon)\lambda K, \mathcal{L}_{i-1}, \varepsilon\lambda r)\,.$$

By the guarantees on Lattice-Enum, the set $\bar{S}$ satisfies

$$3^{i-1}(1-\varepsilon)\lambda K \cap \mathcal{L}_{i-1} \subseteq S \subseteq (3^{i-1}(1-\varepsilon)\lambda K + \varepsilon\lambda r B_2^n) \cap \mathcal{L}_{i-1}\ .$$

Since $rB_2^n \subseteq K$ and $i \geq 1$ we have $3^{i-1}(1-\varepsilon)\lambda K + \varepsilon\lambda r B_2^n \subseteq 3^{i-1}\lambda K$. Therefore,

$$3^{i-1}(1-\varepsilon)\lambda K \cap \mathcal{L}_{i-1} \subseteq \bar{S} \subseteq 3^{i-1}\lambda K \cap \mathcal{L}_{i-1}\,. \tag{5.1}$$

Set $N = |\bar{S}|$ (line 8). By (5.1) and the induction hypothesis we have

$$|3^{i-1}(1-\varepsilon)\lambda K \cap \mathcal{L}_{i-1}| \leq N \leq |3^{i-1}\lambda K \cap \mathcal{L}_{i-1}| \leq G(3^{i-1}\lambda K, \mathcal{L}_{i-1}) \leq 7^{n+4}\,.$$

Assume $N \leq 1000$. Then the algorithm sets $\mathcal{L}_i = \mathcal{L}_{i-1}$ and $B_i = B_{i-1}$. By (5.1), we have that $|3^{i-1}(1-\varepsilon)\lambda K \cap \mathcal{L}_i| \leq N \leq 1000$. Therefore, Lemma 2.3 part 2 yields

$$\begin{aligned} G(3^i\lambda K, \mathcal{L}_i) &\leq (2 \cdot 3(1/(1-\varepsilon)) + 1)^n |3^{i-1}(1-\varepsilon)\lambda K \cap \mathcal{L}_{i-1}| \\ &\leq 7^n(1+2\varepsilon)^n \cdot 1000 \leq 7^{n+4}, \end{aligned}$$

where the last two inequalities follow since $\varepsilon \leq 7^{-(n+5)}$. Therefore $\mathcal{L}_i$ satisfies condition (1), as needed. Furthermore, since $\mathcal{L}_i = \mathcal{L}_{i-1}$, the induction hypothesis trivially implies that $\mathcal{L}_i$ satisfies condition (2).

Assume $N > 1000$. Here, we first compute a prime $p \in \mathbb{N}$ satisfying $N < p < 4N/3 < 1/\varepsilon$, and a dual basis B_{i-1}^* for $\mathcal{L}_{i-1}^*$. Let $S = B_{i-1}^{*T}\bar{S} \pmod{p\mathbb{Z}^n}$. By Theorem 4.3 and Equation (5.1), since

$$p > |\bar{S}| \geq |\mathcal{L}_{i-1} \cap 3^{i-1}(1-\varepsilon)\lambda K| \geq |\mathcal{L}_{i-1} \cap 3^{i-1}(1-1/p)\lambda K|$$

we have that $|S| = N$. Therefore Good-Vector applied to S returns a vector $\mathbf{a} \in \mathbb{Z}_p^n$ satisfying

a. $|\{\mathbf{y} \in 3^{i-1}(1-\varepsilon)\lambda K \cap \mathcal{L}_{i-1} : \langle B^*\mathbf{a}, \mathbf{y}\rangle \equiv 0 \pmod{p}\}| \leq 6$,

b. $|\{\langle B^*\mathbf{a}, \mathbf{y}\rangle \pmod{p} : \mathbf{y} \in 3^{i-1}\lambda K \cap \mathcal{L}_{i-1}\}| \geq \dfrac{p+2}{3}$.

We recall that $\mathcal{L}_i = \{\mathbf{y} \in \mathcal{L}_{i-1} : \langle B^*\mathbf{a}, \mathbf{y}\rangle = 0 \pmod{p}\}$. Using (a) above and Theorem 4.3 part 1b we have that

$$\begin{aligned} G(3^i\lambda K, \mathcal{L}_i) &\leq (2 \cdot 3 \cdot (1/(1-\varepsilon)) + 1)^n |3^{i-1}(1-\varepsilon)\lambda K \cap \mathcal{L}_i| \\ &\leq 7^n(1+2\varepsilon)^n \cdot 6 < 7^{n+4}\,. \end{aligned}$$

Therefore $\mathcal{L}_i$ satisfies condition (1). By (b) above and part 2 of Theorem 4.3 we have that

$$d_K(\mathcal{L}_i, \mathbf{x}) \leq d_K(\mathcal{L}_{i-1}, \mathbf{x}) + 3^i\lambda\,.$$

The induction hypothesis on $\mathcal{L}_{i-1}$ implies that

$$d_K(\mathcal{L}_i,\mathbf{x}) \leq d_K(\mathcal{L}_{i-1},\mathbf{x}) + 3^i\lambda \leq d_K(\mathcal{L},\mathbf{x}) + \frac{3}{2}(3^{i-1}-1)\lambda + 3^i\lambda = d_K(\mathcal{L},\mathbf{x}) + \frac{3}{2}(3^i-1)\lambda\,.$$

Therefore $\mathcal{L}_i$ satisfies condition (2), as needed. The lemma thus follows. □

Proof of Theorem 3.3 (Lattice Sparsifier Construction).

Correctness: We show that the outputted lattice is a (K,t)-sparsifier for $\mathcal{L}$. By Lemma 3.2 it suffices to show that the algorithm outputs a $(K\cap -K,t)$-sparsifier, which justifies the switch in line 2 from K to $K\cap -K$. In what follows, we therefore assume that K is symmetric.

Given Lemma 2, we will show that $\mathcal{L}_k$ is a (K,t)-sparsifier for $\mathcal{L}$. By our choice of k, note that

$$\frac{3}{2}(3^k-1)\lambda \leq t \leq \frac{3}{2}(3^{k+1}-1)\lambda\,.$$

By Lemma 2, for $\mathbf{x}\in\mathbb{R}^n$,

$$d_K(\mathcal{L}_k,\mathbf{x}) \leq d_K(\mathcal{L},\mathbf{x}) + \frac{3}{2}(3^k-1)\lambda \leq d_K(\mathcal{L},\mathbf{x}) + t\,.$$

It therefore only remains to bound $G(tK,\mathcal{L}_k)$. By the previous bounds

$$\frac{t}{3^k\lambda} \leq \frac{3}{2}\frac{(3^{k+1}-1)\lambda}{3^k\lambda} < \frac{9}{2}.$$

Therefore, Lemma 2.3 part 3 implies that

$$G(tK,\mathcal{L}_k) \leq \left(4\cdot\frac{9}{2}+2\right)^n G(3^k\lambda K,\mathcal{L}_k) \leq (20)^n\cdot 7^{n+4} = 2^{O(n)}\,,$$

as needed. The algorithm returns a valid (K,t)-sparsifier for $\mathcal{L}$.

Running time: The algorithm first runs the Shortest-Vectors on K and $\mathcal{L}$, which takes $2^{O(n)}\mathrm{poly}(\cdot)$ time and $2^n\mathrm{poly}(\cdot)$ space. Next, the for loop on line 6 iterates

$$k = \left\lfloor \ln\left(\frac{2}{3}\frac{t}{\lambda}+1\right)/\ln 3\right\rfloor = \mathrm{poly}(\cdot)$$

times.

Each for loop iteration, indexed by i satisfying $0\leq i\leq k-1$, consists of computations over the set $\bar{S}\leftarrow$ Lattice-Enum$(3^i(1-\varepsilon)\lambda K,\mathcal{L}_i,\varepsilon\lambda r)$. For the intended implementation, we do not store the set $\bar{S}$ explicitly. Every time the algorithm needs to iterate over $\bar{S}$, we implement this by performing a call to Lattice-Enum$(3^i(1-\varepsilon)\lambda K,\mathcal{L}_i,\varepsilon\lambda r)$. Furthermore, the algorithm only interacts with $\bar{S}$ by iterating over its elements, and hence the implemented interface suffices. Now at the loop iteration indexed by i, we do as follows:

1. Compute $N = |\bar{S}|$. This is implemented by iterating over the elements of $\bar{S}$ and counting, and so by the guarantees of Lattice-Enum requires at most $2^{O(n)} G(3^i \lambda K, \mathcal{L}_i)\,\mathrm{poly}(\cdot) = 2^{O(n)}\,\mathrm{poly}(\cdot)$ time—-by Lemma 2 part 1—and $2^n\,\mathrm{poly}(\cdot)$ space.

2. If $N \le 1000$, we keep the same lattice and skip to the next loop iteration. If $N > 1000$, continue.

3. Compute $B_i^* = B_i^{-T}$. This can be done in $\mathrm{poly}(\cdot)$ time and space.

4. Compute a prime p satisfying $N < p < 4N/3$. Such a prime can be computed by trying all integers in the previous range and using trial division. This takes at most $O(N^2\,\mathrm{poly}(\log N)) = 2^{O(n)}$ time and $\mathrm{poly}(n)$ space.

5. Call Good-Vector($B^{T*}\bar{S} \pmod{p\mathbb{Z}^n}, p$). By the guarantees on Good-Vector, the algorithm performs $\mathrm{poly}(n, \log p) p^4 = 2^{O(n)}$ operations and iterates at most $np^3 = 2^{O(n)}$ times over the set $B^{T*}\bar{S} \pmod{p\mathbb{Z}^n}$. These iterations can be performed in $2^{O(n)}\,\mathrm{poly}(\cdot)$ time and $2^n\,\mathrm{poly}(\cdot)$ space by the guarantees on Lattice-Enum.

6. Compute a basis B_{i+1} for the new lattice $\mathcal{L}_{i+1} = \{\mathbf{y} \in \mathcal{L}_i : \langle B^{*T}\mathbf{a}, \mathbf{y}\rangle \equiv 0 \pmod{p}\}$. This can be done in $\mathrm{poly}(\cdot)$ time.

From the above analysis, we see that the entire algorithm runs in $2^{O(n)}\,\mathrm{poly}(\cdot)$ time and $2^n\,\mathrm{poly}(\cdot)$ space as needed. $\square$

6 Further applications and future directions

Integer programming. We explain how the techniques in this paper apply to Integer Programming (IP), i. e., the problem of deciding whether a polytope contains an integer point, and discuss some potential associated avenues for improving the complexity of IP. For a brief history, the first breakthrough works on IP are by Lenstra [32] and Kannan [27], where it was shown that any n-variable IP can be solved in $2^{O(n)} n^{2.5n}$ time (with polynomial dependencies on the remaining parameters). Since then, progress on IP has been slow, though recent complexity improvements have been made: the dependence on n was reduced to n^{2n} [24], $\tilde{O}(n)^{\frac{4}{3}n}$ [12], and finally n^n [10].

Let $K \subseteq \mathbb{R}^n$ denote a polytope. To find an integer point inside K, the general outline of the above algorithms is as follows. Pick a center point $\mathbf{c} \in K$, and attempt to "round" $\mathbf{c}$ to a point in $\mathbb{Z}^n$ inside K. If this fails, decompose the integer program on K into subproblems. Here, the decomposition is generally achieved by partitioning $\mathbb{Z}^n$ along shifts of some rational linear subspace H (often a hyperplane) and recursing on the integral shifts of H intersecting K.

In [11], an algorithm is given to perform the above rounding step in a "near-optimal" manner. More precisely, the center $\mathbf{c}$ of K is chosen to be the center of gravity $\mathbf{b}$ of K (which can be estimated via random sampling), and rounding $\mathbf{b}$ to $\mathbb{Z}^n$ is done via an approximate CVP computation with target $\mathbf{b}$, lattice $\mathbb{Z}^n$, and norm $\|\cdot\|_{K-\mathbf{b}}$ (corresponding to scaling K about $\mathbf{b}(K)$). Here the AKS randomized sieve is used to perform the approximate CVP computation, which is efficient due to the fact that $K - \mathbf{b}$ is near-symmetric (see [39]). Let $\mathbf{y} \in \mathbb{Z}^n$ be the returned $(1+\varepsilon)$-CVP solution, and assume that $\mathbf{y}$ is correctly computed (which occurs with high probability). We can now examine the following cases.

If $\mathbf{y} \in K$, we have solved the IP. If $\|\mathbf{y}-\mathbf{b}\|_{K-\mathbf{b}} > (1+\varepsilon)$, then by the guarantee on $\mathbf{y}$, for any $\mathbf{z} \in \mathbb{Z}^n$ we have that $\|\mathbf{z}-\mathbf{b}\|_{K-\mathbf{b}} > 1 \Leftrightarrow \mathbf{z} \notin K$. Hence, we can immediately decide that $K \cap \mathbb{Z}^n = \emptyset$. Lastly, if $1 < \|\mathbf{y}-\mathbf{b}\|_{K-\mathbf{b}} \leq (1+\varepsilon)$, we know that

$$\frac{1}{1+\varepsilon}K + \frac{\varepsilon}{1+\varepsilon}\mathbf{b}$$

is integer-free while $(1+\varepsilon)K - \varepsilon\mathbf{b}$ contains $\mathbf{y}$. In this final case, we are in essentially a near-optimal situation for computing a "good" decomposition (using the so-called "flatness" theorems in the geometry of numbers). We note that with previous methods (i. e., using only symmetric norm or ℓ_2 techniques), the ratio of scalings between the integer-free and not integer-free case was $O(n)$ in the worst case as opposed to $(1+\varepsilon)^2$ (here ε can be any constant ≤ 1).

With the techniques in this paper, we note that the above rounding procedure can be made Las Vegas (i. e., no probability of error, randomized running time) by replacing the AKS Sieve with our new DPV-based solver (randomness is still needed to estimate the center of gravity). This removes any probability of error in the above inferences, making the above rounding algorithm easier to apply in the IP setting. We note that the geometry induced by the above rounding procedure is currently poorly understood, and very little of it is being exploited by IP algorithms. One hope for improving the complexity of IP with the above methods, is that with a strong rounding procedure as above one may be able to avoid the worst case bounds on the number of subproblems created at every recursion node. Currently, the main way to show that K admits a small decomposition into subproblems is to show that the covering radius of K (i. e., the minimum scaling such that every shift of K intersects $\mathbb{Z}^n$) is large. Using the above techniques, we easily get that in the final case the covering radius is at least $1/(1+\varepsilon)$ (since $(1/(1+\varepsilon))K + (\varepsilon/(1+\varepsilon))\mathbf{b}$ is integer-free), however in reality the covering radius could be much larger (yielding smaller decompositions). An interesting direction would be to try and show that on the aggregate (over all subproblems), the covering radii of the nodes must grow as we go down the recursion tree. This would allow us to show that as we descend the recursion tree, the branching factor shrinks quickly, allowing us to get better bounds on the size of the recursion tree (which yields the dominant complexity term for current IP algorithms).

CVP under ℓ_∞. While the ideas presented here do not seem to be practically implementable in general (at least currently), there are special cases where the overhead incurred by our approach may be acceptable. One potential target is solving $(1+\varepsilon)$-CVP under ℓ_∞. This is one of the most useful norms that is often approximated by ℓ_2 for lack of a better alternative.

As an example, in [8], they reduce the problem of computing machine-efficient polynomial approximations (i. e., having small coefficient sizes) of 1-dimensional functions to CVP under ℓ_∞. The goal in this setting is to generate a high quality approximation that is suitable for hardware implementation or for use in a software library, and hence spending considerable computational resources to generate it is justified.

We now explain why the ℓ_∞ norm version of our algorithms may be suitable for practical implementation (or at least efficient "heuristic" implementation). Most importantly, for ℓ_∞ the DPV lattice point enumerator is trivial to implement. In particular, to enumerate the lattice points in a cube, one simply enumerates the points in the outer containing ball and retains those in the cube. Second, if one is comfortable with randomization, the sparsifier can be constructed by adding a simple random modular form to the base lattice. For provable guarantees, the main issue is that the modulus must be carefully chosen (see Section 4), however it seems plausible that in practice an appropriate modulus may be guessed heuristically.

Acknowledgments. We are very grateful to the anonymous referees for their comments and suggestions which greatly helped us improve the quality of the presentation.

References

[1] MIKLÓS AJTAI: The shortest vector problem in L_2 is NP-hard for randomized reductions (extended abstract). In *Proc. 30th STOC*, pp. 10–19. ACM Press, 1998. [doi:10.1145/276698.276705] 2

[2] MIKLÓS AJTAI, RAVI KUMAR, AND D. SIVAKUMAR: A sieve algorithm for the shortest lattice vector problem. In *Proc. 33rd STOC*, pp. 601–610. ACM Press, 2001. [doi:10.1145/380752.380857] 3

[3] MIKLÓS AJTAI, RAVI KUMAR, AND D. SIVAKUMAR: Sampling short lattice vectors and the closest lattice vector problem. In *Proc. 17th IEEE Conf. on Computational Complexity (CCC'02)*, pp. 53–57. IEEE Comp. Soc. Press, 2002. [doi:10.1109/CCC.2002.1004339] 3

[4] SANJEEV ARORA, LÁSZLÓ BABAI, JACQUES STERN, AND Z. SWEEDYK: The hardness of approximate optima in lattices, codes, and systems of linear equations. *J. Comput. System Sci.*, 54(2):317–331, 1997. Preliminary version in FOCS'93. [doi:10.1006/jcss.1997.1472] 2

[5] VIKRAMAN ARVIND AND PUSHKAR S. JOGLEKAR: Some sieving algorithms for lattice problems. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'08)*, volume 2 of *LIPIcs*, pp. 25–36. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2008. [doi:10.4230/LIPIcs.FSTTCS.2008.1738] 3

[6] LÁSZLÓ BABAI: On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986. Preliminary version in STACS'85. [doi:10.1007/BF02579403] 3

[7] JOHANNES BLÖMER AND STEFANIE NAEWE: Sampling methods for shortest vectors, closest vectors and successive minima. *Theoret. Comput. Sci.*, 410(18):1648–1665, 2009. Preliminary version in ICALP'07. [doi:10.1016/j.tcs.2008.12.045] 3

[8] NICOLAS BRISEBARRE AND SYLVAIN CHEVILLARD: Efficient polynomial L^∞-approximations. In *18th IEEE Symposium on Computer Arithmetic (ARITH'07)*, pp. 169–176. IEEE Comp. Soc. Press, 2007. [doi:10.1109/ARITH.2007.17] 29

[9] JIN-YI CAI AND AJAY NERURKAR: Approximating the SVP to within a factor $(1+1/\dim^\varepsilon)$ is NP-hard under randomized reductions. *J. Comput. System Sci.*, 59(2):221–239, 1999. Preliminary version in CCC'98. [doi:10.1006/jcss.1999.1649] 2

[10] DANIEL DADUSH: *Integer Programming, Lattice Algorithms, and Deterministic Volume Estimation*. Ph. D. thesis, Georgia Institute of Technology, 2012. Available at author's webpage. 2, 4, 6, 11, 13, 28

[11] DANIEL DADUSH: A randomized sieving algorithm for approximate integer programming. *Algorithmica*, 70(2):208–244, 2014. Preliminary version in LATIN'12. [doi:10.1007/s00453-013-9834-8] 2, 3, 4, 6, 28

[12] DANIEL DADUSH, CHRIS PEIKERT, AND SANTOSH VEMPALA: Enumerative lattice algorithms in any norm via M-ellipsoid coverings. In *Proc. 52rd FOCS*, pp. 580–589. IEEE Comp. Soc. Press, 2011. [doi:10.1109/FOCS.2011.31, arXiv:abs/1011.5666] 2, 4, 5, 13, 28

[13] DANIEL DADUSH AND SANTOSH S. VEMPALA: Near-optimal deterministic algorithms for volume computation via M-ellipsoids. *Proceedings of the National Academy of Sciences*, 2013. [doi:10.1073/pnas.1203863110] 4

[14] IRIT DINUR: Approximating SVP_∞ to within almost-polynomial factors is NP-hard. *Theoret. Comput. Sci.*, 285(1):55–71, 2002. Preliminary versions in CIAC'00 and ECCC. [doi:10.1016/S0304-3975(01)00290-0] 2

[15] IRIT DINUR, GUY KINDLER, RAN RAZ, AND SHMUEL SAFRA: Approximating CVP to within almost-polynomial factors is NP-hard. *Combinatorica*, 23(2):205–243, 2003. Preliminary version in FOCS'98. [doi:10.1007/s00493-003-0019-y] 2

[16] FRIEDRICH EISENBRAND, NICOLAI HÄHNLE, AND MARTIN NIEMEIER: Covering cubes and the closest vector problem. In *Proc. 27th Symp. on Computational Geometry (SoCG'11)*, pp. 417–423. ACM Press, 2011. [doi:10.1145/1998196.1998264, arXiv:1012.2289] 3

[17] PETER VAN EMDE BOAS: Another NP-complete partition problem and the complexity of computing short vectors in a lattice. Technical Report 81-04, University of Amsterdam, 1981. Available at author's webpage. 2

[18] ANDRÁS FRANK AND ÉVA TARDOS: An application of simultaneous diophantine approximation in combinatorial optimization. *Combinatorica*, 7(1):49–65, 1987. [doi:10.1007/BF02579200] 2

[19] ODED GOLDREICH, DANIELE MICCIANCIO, SHMUEL SAFRA, AND JEAN-PIERRE SEIFERT: Approximating shortest lattice vectors is not harder than approximating closest lattice vectors. *Inform. Process. Lett.*, 71(2):55–61, 1999. [doi:10.1016/S0020-0190(99)00083-6] 5

[20] GUILLAUME HANROT, XAVIER PUJOL, AND DAMIEN STEHLÉ: Algorithms for the shortest and closest lattice vector problems. In *IWCC 2011*, volume 6639 of *LNCS*, pp. 159–190. Springer, 2011. [doi:10.1007/978-3-642-20901-7_10] 3

[21] GUILLAUME HANROT AND DAMIEN STEHLÉ: Improved analysis of Kannan's Shortest Lattice Vector algorithm. In *Advances in Cryptology (CRYPTO) 2007, 27th Internat. Cryptology Conf.*, volume 4622 of *LNCS*, pp. 170–186. Springer, 2007. [doi:10.1007/978-3-540-74143-5_10] 3

[22] ISHAY HAVIV AND ODED REGEV: Tensor-based hardness of the shortest vector problem to within almost polynomial factors. *Theory of Computing*, 8(1):513–531, 2012. Preliminary version in STOC'07. [doi:10.4086/toc.2012.v008a023] 2

[23] BETTINA HELFRICH: Algorithms to construct Minkowski reduced and Hermite reduced lattice bases. *Theoret. Comput. Sci.*, 41:125–139, 1985. [doi:10.1016/0304-3975(85)90067-2] 3

[24] ROBERT HILDEBRAND AND MATTHIAS KÖPPE: A new Lenstra-type algorithm for quasiconvex polynomial integer minimization with complexity $2^{O(n \log n)}$. *Discrete Optimization*, 10(1):69–84, 2013. [doi:10.1016/j.disopt.2012.11.003, arXiv:1006.4661] 3, 28

[25] ANTOINE JOUX AND JACQUES STERN: Lattice reduction: A toolbox for the cryptanalyst. *J. Cryptology*, 11(3):161–185, 1998. [doi:10.1007/s001459900042] 2

[26] MICHAEL KAIB AND HARALD RITTER: Block reduction for arbitrary norms. Manuscript, 1995. Available at the Goethe-Universität webpage. 3

[27] RAVI KANNAN: Minkowski's convex body theorem and integer programming. *Math. Oper. Res.*, 12(3):415–440, 1987. [doi:10.1287/moor.12.3.415] 2, 3, 6, 28

[28] RAVI KANNAN: Lattice translates of a polytope and the Frobenius problem. *Combinatorica*, 12(2):161–177, 1992. Preliminary version in FSTTCS'89. [doi:10.1007/BF01204720] 2

[29] SUBHASH KHOT: Hardness of approximating the shortest vector problem in lattices. *J. ACM*, 52(5):789–808, 2005. Preliminary version in FOCS'04. [doi:10.1145/1089023.1089027] 2, 6

[30] SUBHASH KHOT: Hardness of approximating the shortest vector problem in high L_p norms. *J. Comput. System Sci.*, 72(2):206–219, 2006. Preliminary version in FOCS'03. [doi:10.1016/j.jcss.2005.07.002] 6

[31] ARJEN K. LENSTRA, HENDRIK W. LENSTRA, JR., AND LÁSZLÓ LOVÁSZ: Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982. [doi:10.1007/BF01457454] 2, 3

[32] HENDRIK W. LENSTRA: Integer programming with a fixed number of variables. *Math. Oper. Res.*, 8(4):538–548, 1983. [doi:10.1287/moor.8.4.538] 2, 28

[33] LÁSZLÓ LOVÁSZ AND HERBERT E. SCARF: The generalized basis reduction algorithm. *Math. Oper. Res.*, 17(3):751–764, 1992. [doi:10.1287/moor.17.3.751] 3

[34] DANIELE MICCIANCIO: The shortest vector in a lattice is hard to approximate to within some constant. *SIAM J. Comput.*, 30(6):2008–2035, 2000. Preliminary version in FOCS'98. [doi:10.1137/S0097539700373039] 2

[35] DANIELE MICCIANCIO: Inapproximability of the shortest vector problem: Toward a deterministic reduction. *Theory of Computing*, 8(22):487–512, 2012. Preliminary version in ECCC. [doi:10.4086/toc.2012.v008a022] 2

[36] DANIELE MICCIANCIO AND SHAFI GOLDWASSER: *Complexity of Lattice Problems: a cryptographic perspective*. Volume 671 of *The Kluwer Internat. Series in Engineering and Comput. Science*. Kluwer, 2002. [doi:10.1007/978-1-4615-0897-7] 3

[37] DANIELE MICCIANCIO AND PANAGIOTIS VOULGARIS: A deterministic single exponential time algorithm for most lattice problems based on Voronoi cell computations. *SIAM J. Comput.*, 42(3):1364–1391, 2013. Preliminary version in STOC'10. [doi:10.1137/100811970] 3, 5

[38] DANIELE MICCIANCIO AND MICHAEL WALTER: Fast lattice point enumeration with minimal overhead. In *Proc. 26th Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA'15)*, pp. 276–294. ACM Press, 2015. [doi:10.1137/1.9781611973730.21] 3

[39] VITALI D. MILMAN AND ALAIN PAJOR: Entropy and asymptotic geometry of non-symmetric convex bodies. *Advances in Mathematics*, 152(2):314–335, 2000. [doi:10.1006/aima.1999.1903] 3, 28

[40] WŁADYSŁAW NARKIEWICZ: *The Development of Prime Number Theory, From Euclid to Hardy and Littlewood*. Springer, 2000. [doi:10.1007/978-3-662-13157-2] 17, 18

[41] PHONG Q. NGUYEN AND JACQUES STERN: The two faces of lattices in cryptology. In *Cryptography and Lattices (CaLC'01)*, volume 2146 of *LNCS*, pp. 146–180. Springer, 2001. [doi:10.1007/3-540-44670-2_12] 2

[42] ANDREW MICHAEL ODLYZKO: The rise and fall of knapsack cryptosystems. In *Cryptology and Computational Number Theory*, volume 42 of *Proceedings of Symposia in Applied Mathematics*, pp. 75–88, 1990. Available at author's webpage. 2

[43] ODED REGEV AND RICKY ROSEN: Lattice problems and norm embeddings. In *Proc. 38th STOC*, pp. 447–456. ACM Press, 2006. [doi:10.1145/1132516.1132581] 2

[44] JOHN BARKLEY ROSSER AND LOWELL SCHOENFELD: Approximate formulas for some functions of prime numbers. *Illinois J. Math.*, 6(1):64–94, 1962. Available at Project Euclid. 17, 18

[45] CLAUS-PETER SCHNORR: A hierarchy of polynomial time lattice basis reduction algorithms. *Theoret. Comput. Sci.*, 53(2-3):201–224, 1987. [doi:10.1016/0304-3975(87)90064-8] 3

[46] CLAUS-PETER SCHNORR: Factoring integers and computing discrete logarithms via diophantine approximation. In *Advances in Cryptology (EUROCRYPT'91)*, volume 547 of *LNCS*, pp. 281–293. Springer, 1991. [doi:10.1007/3-540-46416-6_24] 2

[47] MARTIN SEYSEN: Simultaneous reduction of a lattice basis and its reciprocal basis. *Combinatorica*, 13(3):363–376, 1993. [doi:10.1007/BF01202355] 7

[48] EMANUELE VITERBO AND JOSEPH BOUTROS: A universal lattice code decoder for fading channels. *IEEE Trans. Inform. Theory*, 45(5):1639–1642, 1999. [doi:10.1109/18.771234] 2

AUTHORS

Daniel Dadush
Tenure Track Researcher
Centrum Wiskunde & Informatica (CWI)
Amsterdam, The Netherlands
dadush@cwi.nl
http://homepages.cwi.nl/~dadush/

Gábor Kun
Alfréd Rényi Institute of Mathematics
Hungarian Academy of Sciences
13-15 Reáltanoda u., Budapest, Hungary 1053
kungabor@renyi.hu
http://www.renyi.hu/~kungabor/

ABOUT THE AUTHORS

DANIEL DADUSH is a tenure track researcher at the Centrum Wiskunde & Informatica (CWI) in Amsterdam. He earned his Ph. D. in algorithms, combinatorics and optimization (ACO) from Georgia Tech in 2012, where his advisor was Santosh Vempala. Before joining CWI, he spent two years as a Simons postdoctoral fellow in the Computer Science Department at New York University. His research has focused on algorithms for lattice problems, integer programming, and high-dimensional convex geometry. He lives and works in Amsterdam, and is glad that, thus far, the city remains above water level. He enjoys reading the New York Times, listening to NPR, and taking long bike rides through the canals when it is not raining.

GÁBOR KUN works at the Rényi Institute in Budapest. He earned his Ph. D. at Eötvös University, Budapest, where his advisor was Csaba Szabó. This makes him László Babai's mathematical great-grandchild. In his thesis he studied constraint satisfaction problems in terms of logic; he derandomized the reduction that led Feder and Vardi to the CSP dichotomy conjecture. He was a postdoc at DIMACS, the Institute for Advanced Study, and Simon Fraser University and held visiting positions at Charles University (Prague), the Courant Institute, the University of Cambridge, and the University of Memphis. In his spare time he likes to do sports: bridge, soccer, hiking, swimming, and especially chess.