Approximating Multidimensional Subset Sum and the Minkowski Decomposition of Polygons [*]

Ioannis Z. Emiris [†] Anna Karasoulou [†] Charilaos Tzovas[‡]

Abstract

We consider the approximation of two NP-hard problems: Minkowski Decomposition (MinkDecomp) of integral lattice polygons, and the related Multidimensional Subset Sum (kD-SS). We prove, through a gap-preserving reduction, that, for general dimension k, kD-SS does not have an FPTAS. For $2D$-SS, we present an $O(n^7/\epsilon^4)$ approximation algorithm, where n is the set cardinality and ϵ bounds the approximation, and use it to approximate MinkDecomp.

1 Introduction

A polygon Q is called *lattice polygon* when all its vertices are integer points.

Problem 1 ***Minkowski Decomposition (MinkDecomp).*** *Given a lattice convex polygon Q, decide if it is decomposable, that is, if there are nontrivial lattice polygons A and B such that $A + B = Q$, where $+$ denotes Minkowski sum. The polygons A and B are called the* summands.

Problem 2 ***MinkDecomp-μ-approx***
Input: A lattice polygon Q and a parameter $0 < \epsilon < 1$ and μ is a measure of polygons.
Output: Lattice polygons A, B such that $\mu(Q) - \epsilon X < \mu(A+B) < \mu(Q) + \epsilon X$. We call such an output an ϵX-solution.

Problem 1 is proven NP-complete in [5] and can be reduced to $2D$-SS. For the reduction see Section 4.

Problem 3 ***kD-Subset Sum (kD-SS)***
Input: A vector set $S = \{v_i \mid v_i \in \mathbb{Z}^k, 1 \leq i \leq n\}$ and a target vector $t \in \mathbb{Z}^k$.
Output: Decide whether there exist a vector subset $S_\tau = \{v_1, v_2, \ldots, v_\tau\} \subseteq S$ such that $\sum_{i=1}^{\tau} v_i = t$.

This is a generalization of the classic $1D$-SS problem, and as such, is also NP-complete.

Let P_i be the set of all possible vector sums that can be produced by adding at most i elements from the first i vectors in S. Then, P_n is the set of all possible vector sums. Here is the approximation version:

Problem 4 ***kD-SS-approx***
Input: A set $S = \{v_i \mid v_i \in \mathbb{Z}^k, 1 \leq i \leq n, k \geq 1\}$, a nonzero target $t \in P_n$ and $0 < \epsilon < 1$.
Output: Find a subset $S_\tau = \{v_1, v_2, \ldots, v_\tau\} \subseteq S$ whose vector sum t' satisfies $dist(t, t') \leq \epsilon|t|$, where $|t|$ is the length of t.

We consider Euclidean distance l_2 but the discussion is easily generalized to any l_p, $1 \leq p < \infty$.

Definition 1 *A PTAS (Polynomial Time Approximation Scheme) is an algorithm which takes an instance of an optimization problem and a parameter $\epsilon > 0$ and, in polynomial time, produces a solution that is within a factor $1 + \epsilon$ of being optimal for minimazation problems, or $1 - \epsilon$ for maximization. One further defines class FPTAS (Fully PTAS) where the time complexity is polynomial in both input size and parameter ϵ.*

Previous work $1D$-SS and kD-SS are not strongly NP-complete and can be solved exactly in pseudo-polynomial time: $1D$-SS is solved in $O(nt)$ and, generalizing this idea, kD-SS is solved in $O(n|M|^k)$, where $M = \max P_n$ is the farthest reachable point; information for $k = 2$ in [8]. Moreover, $1D$-SS is in FPTAS [6].

A similar problem to kD-SS is the Closest Vector Problem (CVP): we are given a set of basis vectors $B = \{b_1, \ldots, b_n\}$, where $b_i \in \mathbb{Z}^k$, and a target vector $t \in \mathbb{Z}^k$, and we ask what is the closest vector to t in the lattice $\mathcal{L}$ generated by B. The lattice of B is $\mathcal{L}(B) = \{\sum_1^m a_i b_i \mid a_i \in \mathbb{Z}\}$ and thus kD-SS is a special case of CVP, where $a_i \in \{0, 1\}$. CVP cannot be approximated within a factor of $2^{\log^{1-\epsilon} n}$ [1, 3].

MinkDecomp has its fair share of attention. One application is in the factorization of bivariate polynomials through their Newton polygons. If a polynomial factors, then its Newton polygon has a Minkowski decomposition. Here, we are interested in finding approximate solutions of the latter: polygons

[*]Ioannis Z. Emiris and Anna Karasoulou were partially supported by H2020 M. Sklodowska-Curie Innovative Training Network "ARCADES: Algebraic Representations in Computer-Aided Design for complEx Shapes", 2016-2019.

[†]Department of Informatics & Telecommunications, University of Athens, {emiris,akarasou}@di.uoa.gr

[‡]Department of Mathematics, University of Athens, tzovasx@math.uoa.gr

whose Minkowski Sum is almost the original polygon. MinkDecomp is NP-complete for integral polygons, and a pseudopolynomial algorithm exists [5]. An algorithm for polynomial irreducibility testing using MinkDecomp is presented in [7]. They present a criterion for MinkDecomp that reduces the decision problem into a question in linear programming. Continuing the work in [4], we propose a poly-time algorithm that solves MinkDecomp approximately using a solver for $2D$-SS.

Our contribution We introduce the kD-SS problem. It is clearly NP-complete; we prove that it cannot be approximated efficiently. We design an algorithm for $2D$-SS-approx: given a set S, $|S| = n$, target t and $0 < \epsilon < 1$, the algorithm returns, in $O(n^7\epsilon^{-4})$, a subset of S whose vectors sum to t' such that $dist(t, t') \leq \epsilon M$, where M is the length of the largest possible sum of vectors in any subset of S. This algorithm yields an approximation algorithm for MinkDecomp: If Q is the input polygon the algorithm returns polygons A and B whose Minkowski sum defines polygon Q' such that $vol(Q) \leq vol(Q') \leq vol(Q) + \epsilon D^2$ and $per(Q) \leq per(Q') \leq per(Q) + 2\epsilon D$, where D is the diameter of Q, i.e., the maximum distance between two vertices of Q.

2 kD-SS is not in FPTAS

To prove that kD-SS is not in FPTAS we will apply the idea for the CVP, which is not in PTAS [1]. We will change their proof and apply it to our problem to prove something weaker for kD-SS.

Given a CNF formula ϕ we invoke Proposition 1 and get an instance of the Set Cover problem. This is a gap introducing reduction, because if ϕ is statisfiable then the instance of Set Cover has a solution of size exactly K and if ϕ is not statisfiable every solution has size at least cK for a constant c. From this instance of Set Cover we create an instance for kD-SS that preserves the gap. Now, if ϕ is satisfiable, the closest vector to a given target t has distance exactly K. If ϕ is not statisfiable, the closest vector in target t has distance at least cK.

We reduce kD-SS to Set Cover for norm l_1, but this can easily be generalized to any l_p, where p is a positive integer. We say that a cover is *exact* if the sets in the cover are pairwise disjoint.

Proposition 1 *[2] For every $c > 1$ there is a polynomial time reduction that, given an instance ϕ of SAT, produces an instance of Set Cover $\{\mathcal{U}, (S_1, \dots, S_m)\}$ where $\mathcal{U}$ is the input set of integers and $S_1, \dots, S_m$ are subsets of $\mathcal{U}$, and integer K with the property: If ϕ is satisfiable, there is an exact cover of size K, otherwise all set covers have size more than cK.*

Theorem 2 *Given a CNF formula ϕ and $c > 1$ we create an instance $\{v_1, \dots, v_n, t\}$ of kD-SS. If ϕ is satisfiable, then the minimum distance from t is less than K or otherwise, it is more than cK.*

Proof. Let $\{\mathcal{U}, (S_1, \dots, S_m), K\}$ be the instance of Set-Cover obtained in Proposition 1 for the formula ϕ. We transform it to an instance of kD-SS with input set S and target t, such that the distance of t from the nearest point in the set of all possible points P_n is either K or $\geq cK$.

Let v_i be the vectors of the reduction that will have $n+m$ coordinates, where $|\mathcal{U}| = n$. We will create such a vector v_i for every set S_i. Let $L = cK$. Then the first n coordinates of each vector v_i have their j'th-coordinate ($j \leq n$) equal to L, if the corresponding j'th-element belongs to set S_i, or 0 otherwise. The remaining m coordinates have 1 in the $(n+i)$'th-coordinate and zeros everywhere else:

$$v_i = (L \cdot \chi_{S_i}, 0, \dots, 1, \dots 0) = (L \cdot \chi_{S_i}, e_i)$$

The target vector t has in the first n coordinates L and the last m are zeros, $t = (L, \dots, L, 0, \dots, 0)$.

Now, let the instance of Set-Cover have an exact cover of size K. We will prove that the minimum distance from target t is less than K. Without loss of generality, name the solution $\{S_1, \dots, S_K\}$. For each S_i, $1 \leq i \leq K$ sum the corresponding vectors v_i and let this be $\zeta \in \mathbb{Z}^{n+m}$:

$$\zeta = \sum_{i=1}^{K} v_i = (\underbrace{L, \dots, L,}_{n} \underbrace{1, \dots, 1}_{K} \underbrace{0, \dots, 0}_{m-K}).$$

The first n variables must sum up to $(L, L \dots, L)$, because if one of the coordinates was 0, the solution would not be a cover and if one of them was greater than L, then some element is covered more than once and the solution would not be exact. The key point is that in the last m coordinates we will have exactly K ones. The distance of this vector ζ from t is

$$\| -t + \zeta\|_1 = \|(\underbrace{0, \dots, 0,}_{n} \underbrace{1, \dots, 1}_{K} \underbrace{0, \dots, 0}_{m-K})\|_1 = K$$

Thus, there is a point in P_n that its distance from t is at most K.

Let us consider the other direction, where the Set Cover instance has a solution set greater than $cK = L$. We will show that the closest vector in t has distance at least L. This solution must have at least $cK = L$ sets. As before, $\| -t + \zeta\|_1 \geq L$ (this time the cover need not be exact).

Towards a contradiction, suppose there exists a vector ξ such that $\| -t + \xi\|_1 < L$. If the corresponding sets do not form a cover of S then one of the first n coordinates of ξ is 0 and this alone is enough for $\| -t + \xi\|_1 > L$. If the sets form a cover that is not exact, then in at least one of the first n coordinates of ξ will be greater than L (for the element that is covered more than once) and will force $\|-t+\xi\|_1$ to be greater than L. Finally, if the sets form an exact cover, the first n coordinates of $\|-t+\xi\|_1$ will be 0. For the distance to be less than L, in the last m coordinates there must be less than L units implying that the sets in the cover are less than L contradicting our hypothesis.

In all cases, there cannot exist a vector whose distance from t is $< cK$. $\square$

Theorem 3 *There is no FPTAS for kD-SS-approx unless P=NP.*

Proof. ϕ is a given formula. Suppose there exists an FPTAS for kD-SS-approx. Run the algorithm with $\epsilon = 1/cn$ and target t. Since $\|t\|_1 = ncK$, we are looking for possible solutions within distance $\epsilon ncK = K$ from t. By Theorem 2, we distinguish whether ϕ is satisfiable or not in polynomial time. $\square$

3 The approximation algorithm for $2D$-SS

In this section we discuss the approximation algorithm for $2D$-SS. The idea is to create all possible vectors step by step. At each step, if two vectors are close to each other, one is deleted. Whenever we refer to distance it is the Euclidean distance. We begin with notation.

- Input: the set $S = \{v_1, v_2, \ldots, v_n\}$ with $v_i = (x_i, y_i) \in \mathbb{Z}^2$ and $|S| = n$, parameter $0 < \epsilon < 1$.
- P_i is the set of all possible vectors that can be produced by adding at most i elements from the first i vectors in S. P_n is the set of all possible vector sums.
- $E_i = L_{i-1} \cup \{L_{i-1} + v_i\}$ is the list created at the beginning of every step and that is about to get trimmed.
- $L_i = trim(E_i, \delta)$ is the trimmed list.

At the beginning of the i-th step we create the list $E_i = L_{i-1} \cup \{L_{i-1} + v_i\}$. Notice that, addition is over $\mathbb{Z}^2$, and after a point is found we calculate its length and sort E_i based on the lengths. For each vector $u \in E_i$ with length $|u|$ and angle $\theta(u)$, check all the vectors $u' \in E_i$ that have length $|u| \le |u'| \le (1+\delta)|u|$. If also $\theta(u) - \delta \le \theta(u') \le \theta(u) + \delta$, remove u'. L_i is the trimmed list, that is, from the list E_i we remove vectors that are close to each other. The two conditions ensure that $dist(u', u) \le \alpha\delta|u|$, where $1 \le \alpha \le 2$ is a constant. Every vector that is deleted from E_i is not very far away from a vector in L_i:

$$\forall u \in E_i, \exists w \in L_i : u = w + r_w,\ |r_w| \le \alpha\delta|w| \qquad (1)$$

hence, $|w| \le |u| \le (1+\delta)|w|$. See fig. 1.

Lemma 4 *Call function $L_i = \text{TRIM}(E_i, \delta)$ where E_i is an input list of vectors and $\delta = \epsilon/2n$ the parameter. Then $|L_i| = O(n^3\epsilon^{-2})$ for $1 \le i \le n$.*

Proof. Let $M_i = max\{|x_k| : x_k \in E_i\}$, the vector in E_i with the largest magnitude. Every vector in E_i has length between $(1+\delta)^k$ and $(1+\delta)^{k+1}$ that forms an annulus, called zone. Solving $(1+\delta)^k \ge M_i$ for k, there are $O(n^2/\epsilon)$ many zones.

Every zone can be divided in cells. Each cell is taken in such a way that it will cover $2\delta R$ of the lower circle, where R is the radius of the circle. Thus every zone has at most $2\pi R/\delta R = 4\pi n/\epsilon$ cells. List L_i has at most an entry for every cell created and size can be $|L_i| \le (n^2/\epsilon)\cdot(4\pi n/\epsilon) = O(n^3\epsilon^{-2})$. $\square$

The running time for $2D$-SS-approx is $O(n|L_n|^2)$ and overall it requires time $O(n^7\epsilon^{-4})$.

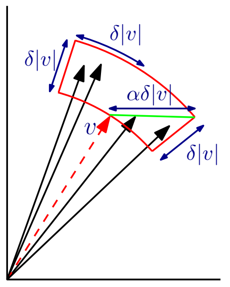

Figure 1: One cell for vector v.

Theorem 5 *For a set of vectors $S = \{v_i \mid v_i \in \mathbb{Z}^2, 0 \le i \le n\}$, every vector sum $v \in P_n$ can be approximated by a vector w*

$$\forall v \in P_n, \exists w \in L_n, \exists r_w : v = w + r_w,$$
$$|r_w| \le n\delta \max\{L_n\},$$

where $\max\{L_i\}$ is the length of the largest vector.

Proof. The proof is by induction. $\square$

Setting $\delta = \epsilon/2n$ we can ensure that every possible vector sum will be approximated by a vector in L_n at most $\epsilon \max\{L_n\}$ far. Implementing and testing the algorithm, much better bounds are obtained.

Lemma 6 *Let $S = \{v_i \mid v_i \in \mathbb{Z}^2, 0 \le i \le n\}$ be the input set of vectors. If also all vectors are in one quadrant and for all $v_i \in S : |v_i| \ge \delta\sum_1^n |v_i|$ then*

$$\forall v \in P_n,\ \exists w \in L_n,\ \exists r_w : v = w + r_w,\ |r_w| \le n\delta|w|$$

Proof. The proof is by induction. $\square$

In the special case, where all vectors are within an angle of 90 degrees and there are no short vectors, this algorithm gives an $(1+\epsilon)$-approximation solution meaning an FPTAS for $\delta = \epsilon/2n$.

4 Minkowski Decomposition using $2D$-SS

In this section we will describe an algorithm for approximating MinkDecomp. The algorithm takes an input polygon Q, transform it to an instance $\{S, t\}$ of $2D$-SS-approx and calls our algorithm for $2D$-SS-approx to solve MinkDecomp.

Let Q be the input to MinkDecomp: $Q = \{v_i \mid v_i \in \mathbb{Z}^2, 0 \leq i \leq n\}$. First we create the vector set U by subtracting successive vertices (in clockwise order): $U = \{v_1 - v_2, \dots, v_n - v_0\}$. We call U the *edge sequence* and each vector is called an *edge vector*. Its edge sequence is denoted by $s(Q)$. For each edge vector v in $s(Q)$ we calculate its *primitive vector*.

Definition 2 *Let $v = (a, b)$ be a vector and $d = gcd(a, b)$. The* primitive vector *of v is $e = (a/d, b/d)$.*

We get an edge vector $(x, y) \in s(Q)$ and calculate its primitive vector $e = (x/d, y/d)$, where $d = gcd(x, y)$. The scalars $d_1, \dots, d_k$ are computed by:

$$d_i = 2^i,\ i = 0, \dots, \lfloor \log_2 d/2 \rfloor \text{ and } d_k = d - \sum_{i=1}^{\lfloor \log_2 d/2 \rfloor} d_i$$

We create the set S by adding the vectors $d_i e$ and repeat the procedure for all vectors $v \in s(Q)$. Notice that $\sum_1^k d_i = d$, so the primitive edge sequence also sums to $(0, 0)$. Using this construction, the primitive vectors added are $\log d$ for every $v \in s(Q)$. The primitive edge sequence uniquely identifies the polygon up to translation determined by v_0. This is a standard procedure as in [5, 4].

The main defect in this approach is that the algorithm returns a sequence of vectors $S' \subset S$ that sum close to $(0, 0)$ but possibly not $(0, 0)$. This means the corresponding edge sequence does not form a closed polygon. To overcome this, we just add to $s(A)$ the vector v, from the last point to the first, to close the gap. If $s(A)$ sums to a point (a, b), by adding vector $v = (-a, -b)$ to $s(A)$ the edge sequence $s(A) \cup \{v\}$ now sums to $(0, 0)$. If we rearrange the vectors by their angles, they form a closed, convex polygon that is summand A. We do the same for the sequence $s(B)$. Notice that the vector added in $s(B)$ is $-v = (a, b)$ and this sequence (rearranged) also forms a closed, convex polygon. We name $s(A') = s(A) \cup \{v\}$, $s(B') = s(B) \cup \{v\}$ and take their Minkowski Sum $Q' = A' + B'$, where A' and B' are the convex polygons formed by $s(A')$ and $s(B')$. We measure how close Q' is to the input Q. Let D be the diameter of Q, the maximum distance between two vertices of Q.

Lemma 7 *Let the summands A' and B' of Q', as discussed. We deduce that $vol(Q') \leq vol(Q) + \epsilon D^2$ and $per(Q') \leq per(Q) + 2\epsilon D$.*

Proof. We observe that

$$s(Q') = s(A') \cup s(B') = s(A) \cup s(B) \cup \{v, -v\} \implies$$
$$s(Q') = s(Q) \cup \{v, -v\}.$$

This equals adding to Q a single segment s of length $|s| = |v|$ and $Q' = Q + s$. Since $per(Q) = \sum_{v \in s(Q)} |v|$, it follows $per(Q') = per(Q) + 2|v|$. For the volume of Q', at worst, where v is perpendicular to D, a rectangle with sides v and D is added to Q. This extra volume is $\leq |v|D$, thus $vol(Q') \leq vol(Q) + |v|D$. It is also easily observed that $vol(Q) \leq vol(Q')$ and $per(Q) \leq per(Q')$ since $Q' = Q + s$.

The length of vector v we add to close the gap, is the key factor to bound polygon Q'. From the guarantee of the $2D$-SS-approx algorithm we know that $s(A)$ (and respectively $s(B)$) sum to a vector with length at most $\epsilon \max\{L_n\}$. This is vector v and thus $|v| \leq \epsilon \max\{L_n\}$. Since $\max\{L_n\} \leq D$, we get $|v| \leq \epsilon D$. This yields $per(Q) \leq per(Q') \leq per(Q) + 2\epsilon D$ and $vol(Q) \leq vol(Q') \leq vol(Q) + \epsilon D^2$. $\square$

Corollary 8 *The proposed algorithm provides a $2\epsilon D$-solution for MinkDecomp-per-approx and an ϵD^2-solution for MinkDecomp-vol-approx.*

This algorithm is implemented in Python 3, it is openly available through Github and Sage[1] and tested for polygons with up to 100 vertices and $\epsilon \in [0.1, 0.5]$ giving much better errors than the bounds proven. For instances with 50 vertices and ϵ=0.2, the algorithm needs around 60 minutes to find the summands.

References

[1] S. Arora, L. Babai, J. Stern, and Z. Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. *Journal of Computer and System Sciences*, 54(2):317 – 331, 1997.

[2] M. Bellare, S. Goldwasser, C. Lund, and A. Russell. Efficient probabilistically checkable proofs and applications to approximations. In *Proceedings of the Twenty-fifth Annual ACM Symposium on Theory of Computing*, STOC '93, pages 294–304, New York, NY, USA, 1993. ACM.

[3] I. Dinur, G. Kindler, R. Raz, and S. Safra. Approximating CVP to within almost-polynomial factors is NP-hard. *Combinatorica*, 23(2):205–243, 2003.

[4] I. Emiris and E. Tsigaridas. Minkowski decomposition of convex lattice polygons. In *Algebraic Geometry and Geometric Modeling*, Mathematics and Visualization, pages 217–236. Springer Berlin Heidelberg, 2006.

[5] S. Gao and A. Lauder. Decomposition of polytopes and polynomials. *Discrete and Computanional Geometry*, 26(1):89–104, 7 2001.

[6] O. H. Ibarra and C. E. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *J. ACM*, 22(4):463–468, Oct. 1975.

[7] D. Kesh and S. Mehta. Polynomial irreducibility testing through minkowski summand computation. In *Proceedings of the 20th Canadian Conf. on Comp. Geom.*, pages 43–46, 2008.

[8] G. Zirdelis. Manuscript on Minkowski decomposition of convex lattice polygons. Course Project. 2014.

[1] https://github.com/tzovas/Approximation-Subset-Sum-and-Minkowski-Decomposition