Computing efficiently the lattice width in any dimension

Émilie Charrier, Fabien Feschet, Lilian Buzer

Efficient Lattice Width Computation in Arbitrary Dimension

Émilie Charrier[123], Lilian Buzer[12] and Fabien Feschet*[4]

[1] Université Paris-Est, LABINFO-IGM, CNRS, UMR 8049
[2] ESIEE, 2, boulevard Blaise Pascal, Cité DESCARTES, BP 99
93162 Noisy le Grand Cedex, France
[3] DGA/D4S/MRIS
[4] Univ. Clermont 1, LAIC,
Campus des Cézeaux, 63172 Aubière Cedex, France
charriee@esiee.fr,buzerl@esiee.fr,feschet.research@gmail.com

Abstract. We provide an algorithm for the exact computation of the lattice width of an integral polygon K in linear-time with respect to the size of K. Moreover, we describe how this new algorithm can be extended to an arbitrary dimension thanks to a greedy approach avoiding complex geometric processings.

1 Introduction

Integer Linear Programming is a fundamental tool in optimization, in operation research, in economics... Moreover, it is interesting in itself since the problem is NP-Hard in the general case. Several works were done for the planar case [18,25,14] before Lenstra [21] proved that Integer Linear Programming can be solved in polynomial time when the dimension is fixed. Faster and faster algorithms are nowadays developed and available making the use of Integer Linear Programming reliable even for high dimensional problems. The approach of Lenstra uses the notion of *lattice width* for precise lattice definition to detect directions for which the polyhedron of solutions is thin. In polynomial time, the problem is then reduced to a feasibility question: given a polyhedron P, determine whether P contains an integer point. To solve it, Lenstra approximates the width of the polyhedron and gives a recursive solution solving problems of smaller dimension. The approximate lattice width is also used in the recent algorithms of Eisenbrand and Rote [9] and Eisenbrand and Laue [8] for the 2-variable problem.

Not surprisingly, following the arithmetical approach of Reveillès [23] [6], the lattice width is also a fundamental tool in digital geometry since it corresponds to the notion of width for digital objects [10]. Moreover, as an application of the

* Supported in part by the the French National Agency of Research under contract GEODIB ANR-06-BLAN-0225-02. A preliminary version of part of this paper appeared in the proceedings of the IWCIA 2006 conference.

lattice width computation, we mention the intrinsic characterization of linear structures [11]. Indeed, the lattice width can be computed for any digital set but it does not correspond to a direct measure of linearity. However, when combining the lattice width along a direction and along its orthogonal, it can be used as a linearity measure. The work in [11] is currently extended, by the third author of the present paper, to higher dimensions for detecting either linear or tubular structures. A preliminary algorithm dealing with the two-dimensional case was given in [10] with a geometrical interpretation. It has the advantage to be extensible to the incremental and to the dynamic case but it seems difficult to extend it to an arbitrary dimension. Thus, in the present paper, we propose a totally new algorithm which is efficient for any dimension and which runs in linear time for the two-dimensional case.

The paper is organized as follows. In Sect. 2, the main definitions and tools are presented. Then, we present the two-dimensional algorithm introduced in [10] in Sect. 3. As this method cannot be easily extensible to higher dimension, we introduce in Sect. 4 a new geometric method to estimate the lattice width in any dimension. This approach is based on the computation of a particular surrounding polytope which is used to bound the set of candidate vectors to define the lattice width. In Sect. 5, we describe geometrical methods to compute such a polytope. We first focus on the two-dimensional case and we provide two deterministic approaches. Then, since these geometric constructions might be difficult to extend in arbitrary dimension, we provide a greedy algorithm which runs in any dimension. Some conclusions and perspectives end the paper.

2 Definitions

In this section, we review some definitions from algorithmic number theory and we provide a precise formulation of the problem we solve. Definitions are taken from [2,9,26].

Let K be a set of n points of $\mathbb{Z}^d$. Moreover, we suppose that all numbers appearing in the points and in the vectors coordinates have their bit size bounded by $\log s$. The *width* of K along a direction $c \neq 0$ in $\mathbb{R}^d$ is defined as:

$$\omega_c(K) = \max\left\{c^T x \;\middle|\; x \in K\right\} - \min\left\{c^T x \;\middle|\; x \in K\right\} \tag{1}$$

Geometrically, if a set K has a width of l along the direction c then any integer point which lies on K also lies on a hyperplane of the form $c^T x = \lambda$ where λ corresponds to an integer value between $\min\{c^T x \mid x \in K\}$ and $\max\{c^T x \mid x \in K\}$. We say that K can be covered by these $\lfloor l \rfloor + 1$ parallel hyperplanes. It is straightforward to see that $\omega(K) = \omega(\mathrm{conv}(K))$ where $\mathrm{conv}(K)$ denotes the convex hull of K.

Let $\mathbb{Z}^{d*} = \mathbb{Z}^d \setminus \{0\}$ denote the set of integer vectors different from zero. The *lattice width* of K is defined as follows:

$$\omega(K) = \min_{c \in \mathbb{Z}^{d*}} \omega_c(K) \tag{2}$$

We notice that the lattice width is an integer value. We briefly recall some basic and important properties about inclusion and translation:

Lemma 1. *For any sets of points A and B, such that $conv(A) \subset conv(B)$ and for any vector $c \in \mathbb{Z}^{d*}$, we have $\omega_c(A) \leq \omega_c(B)$. Thus, it follows that $\omega(A) \leq \omega(B)$.*

Lemma 2. *Suppose that A' corresponds to the points of A translated in the same direction. By definition, we know that for any $c \in \mathbb{Z}^{d*}$, $\omega_c(A) = \omega_c(A')$ and so we have $\omega(A) = \omega(A')$.*

The problem we would like to solve is the following one:

Problem (Lattice Width)
Given a set of integer points $K \subset \mathbb{Z}^d$, find its lattice width $\omega(K)$ as well as all vectors $c \in \mathbb{Z}^d$ such that $\omega_c(K) = \omega(K)$.

It is known [21] that the lattice width of a convex set K is obtained for the shortest vector with respect to the *dual norm* whose unit ball is the polar set of the set $\frac{1}{2}(K + (-K))$. In the general case, computing the shortest vector is NP-hard. Thus, approximations of the solution can be computed via standard arguments [26,17,24], but it does not lead us to an easy exact algorithm in arbitrary dimension.

3 Planar case

We recall in this part, the result obtained in [10] via connections with the notion of digital straightness and more precisely with the notion of arithmetical digital lines [23]. As this two-dimensional algorithm requires a convex polygon as input, we have to compute the convex hull H of K in $O(n)$ time ([16]).

The idea in [10] is based on the principle that the lattice width of H is necessarily reached for two *opposite* vertices of H. To define the notion of *opposite*, we rely on the notion of *supporting lines* well known in computational geometry [5]. A *supporting line* of H is a line D such that $D \cap K \neq \emptyset$ and H is contained entirely in one of the half-planes bounded by D. For each supporting line D, there exists at least one vertex v of H such that the parallel line D_v to D passing through v is such that H entirely lies in the strip bounded by D and D_v. If s denotes a vertex of H belonging to D then s and v are called *opposite* (see Fig. 1, left). Opposite pairs are also called *antipodal* pairs. Note that in general, a supporting line intersects H at only one point. The supporting line D intersecting H along an edge is called *principal* supporting line.

We now suppose H to be oriented counter clockwise. As in the classical Rotating Calipers algorithm of Toussaint [15], we can rotate the principal supporting lines D around the right vertex of $D \cap H$. D_v is also rotated around v to keep it parallel to D. This rotation can be pursued until D or D_v corresponds to another principal supporting line. Note that D and D_v are simply supporting lines during the rotation. At each position of the rotation s and v form an opposite

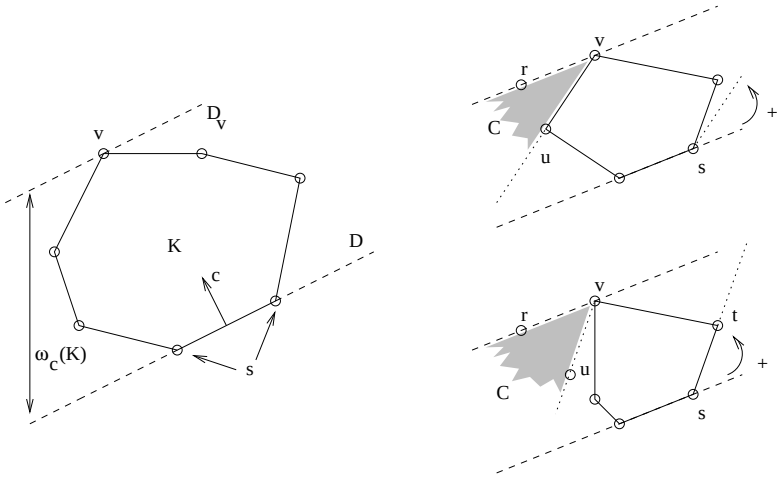

Fig. 1. (left) Supporting lines and width $\omega_c(K)$ (right) Cone of rotations

pair of points which exactly define $\omega_c(H)$ where c is the normal direction to D. Hence, as depicted in Fig. 1 (right), s and v exactly define $\omega_c(H)$ for all D in a cone whose apex is v. The point r is such that the segment from v to r has exactly the same length than the opposite edge of H and the point u is either the next vertex of H after v or the point on the parallel of the line (st) such that the length of $[st]$ equals the length of $[vu]$.

After one turn around H, we have constructed at most $2n$ opposite pairs and associated cones. Hence, the number of cones is $O(n)$. Moreover, the series of cones forms a partition of all possible directions of computation for the lattice width taking into account that $\omega_c(H) = \omega_{-c}(H)$. Hence, as previously announced, the computation of the lattice width is reduced to the computation of the minimal value of $\omega_c(H)$ for each cone.

In each cone C, the computation of the minimal value of $\omega_c(H)$ is the computation of the shortest vectors for the dual norm. They are thus located at the vertices of the border of the convex hull of integer points except v inside the cone [17]. This set is also known as Klein's sail [19,20,1]. Note that to allow the possibility to find all solution vectors, repetitions in the convex hull must be kept. To compute Klein's sail, we use an adapted version of the algorithm of Harvey [13] whose complexity is $O(\log s)$ arithmetic operations for each sail computation. To bound the complexity of the search, we could also rely on the general theorem of Cook et al [4] which says that there exists at most $O((\log s)^{d-1})$ vertices in dimension d, a result also shown by Harvey [13] with an explicit example of the worst case in two dimension. Thus, the complexity of the lattice width computation is $O(n + n \log s)$.

4 A new algorithm

The geometric approach of the previous algorithm appears as a major drawback for an arbitrary dimension. Hence, we propose a new geometric method based on the possibility of bounding the set of candidate directions used to compute the lattice width. We introduce the new method in the d-dimensional case.

Let $(a_j)_{1\leq j\leq d}$ denote a sequence of d vectors in $\mathbb{Z}^d$ such that for any $j, 1 \leq j \leq d$, there exist two points u_j and v_j of K satisfying $a_j = u_j - v_j$. Let us consider a surrounding polytope $\Gamma = \text{conv}((\gamma_i)_{1\leq i\leq m})$ such that Γ contains $\text{conv}(K)$ and such that each of its vertices γ_i, $1 \leq i \leq m$, satisfies $\gamma_i = p + \sum_{1\leq j\leq d} \alpha_{ij} a_j$. We present our core idea. If we can compute such a surrounding polytope such that the values $|\alpha_{ij}|$, $1 \leq i \leq m$, $1 \leq j \leq d$ are bounded by a constant α, then we can determine a set of candidate vectors C satisfying these properties: the cardinality of C is bounded by a constant and the set C contains all the solution vectors of our lattice width problem. Thus, by computing all the $\omega_u(K)$,$u \in C$, in $O(dn)$ time, we determine $\omega(K)$ and the associated solution vectors. Moreover, the constant which bounds the cardinality of C is independent from K and depends only on d and α.

Let c denote a vector in $\mathbb{Z}^{d*}$. By definition of width, we know that:

$$\omega_c(\Gamma) = \max_{u\in\Gamma}\{c^T u\} - \min_{u\in\Gamma}\{c^T u\}$$

As the lattice width is translation invariant, we obtain the following inequality:

$$\omega_c(\Gamma) \leq 2 \max_{1\leq i\leq m} \{|c^T \sum_{1\leq j\leq d} \alpha_{ij} a_j|\}$$

As the values $|\alpha_{ij}|, 1 \leq i \leq m, 1 \leq j \leq d$ are bounded by α, we obtain:

$$\omega_c(\Gamma) \leq 2\alpha d \sum_{1\leq j\leq d} |c^T a_j|$$

Let A denote the matrix whose rows correspond to the $a_j^T, 1 \leq j \leq d$, we can rewrite the previous expression:

$$\omega_c(\Gamma) \leq 2\alpha d||A^T c||_\infty \tag{3}$$

As $\text{conv}(K)$ is included in $\text{conv}(\Gamma)$, we immediately know:

$$\omega(K) \leq \omega(\Gamma) \leq \omega_c(\Gamma)$$

Thus, for a given vector c, we obtain an upper bound for $\omega(K)$. For solving our problem, it is sufficient to test only the vector u in $\mathbb{Z}^{d*}$ satisfying:

$$\omega_u(K) \leq 2\alpha d||A^T c||_\infty$$

Let us try to determine a lower bound for the term $\omega_u(K)$. For any j, $1 \leq j \leq d$, there exists two points u_j and v_j in K such that $a_k = u_k - v_k$. As for any

$j, 1 \leq j \leq d$, $\text{conv}(\{u_j, v_j\})$ is included in $\text{conv}(K)$, we have for any $u \in \mathbb{Z}^{d*}$, $\omega_u(\{u_j, v_j\}) \leq \omega_u(K)$. Thus, by definition of the lattice width along a direction $u \in \mathbb{Z}^{d*}$, we obtain:

$$|u^T a_i| \leq \omega_u(K) \text{ for } 1 \leq i \leq d$$

Then, it follows for any $u \in \mathbb{Z}^{d*}$:

$$||A^T u||_\infty \leq \omega_u(K)$$

Thus, we can conclude that it is sufficient to test only the vector u in $\mathbb{Z}^{d*}$ satisfying :

$$||A^T u||_\infty \leq 2\alpha d||A^T c||_\infty \tag{4}$$

As the right term is fixed for a given c, we can compute a vector c with some interesting properties. The more natural approach is to compute the shortest vector v in the lattice given by A and to choose c such that $A^T c = v$. Hence the upper bound becomes a bound independent on the direction c and we get:

$$\|v\|_\infty \leq \|A^T u\|_\infty \leq 2\alpha d\|v\|_\infty$$

It follows that the set of tested vectors is contained in a ball with a radius independent of the set K. Since there is a constant number of points in the ball, all points can be tested to extract the lattice width. Of course, an approximation of the shortest vectors can be used in place of v to avoid the difficulty of its computing in arbitrary dimension.

5 Surrounding Polytope Computation

Let K denote a set of n points in $\mathbb{Z}^d$. We look for a sequence of d vectors $(a_j)_{1 \leq j \leq d}$ such that for any $j, 1 \leq j \leq d$, there exists two points u_j and v_j of K satisfying $a_j = u_j - v_j$. From this sequence of vectors, we must be able to build a surrounding polytope $\Gamma = conv((\gamma_i)_{1 \leq i \leq m})$ such that Γ contains K and such that each $\gamma_i, 1 \leq i \leq m$ satisfies $\gamma_i = \sum_{j=1}^{d} \alpha_{ij} a_j$. An implicit goal is to obtain the smallest possible upper bound for the $|\alpha_{ij}|, 1 \leq i \leq m, 1 \leq j \leq d$ values in order to improve the performance of the algorithm. We show afterwards that, in the two-dimensional case, we can compute such a surrounding polygon in $O(n)$ time. The corresponding approaches become too difficult to extend to the three-dimensional case. So, we present a simpler and more efficient approach that can be used in any dimension.

5.1 Deterministic Approaches

Existing Approach. We first find in the literature the work of Fleischer et al. [12] that confirms that such a polytope exists. We recall this result in the two-dimensional case:

Theorem 1. *For any convex body P, let t denote a greatest area triangle contained in P. The convex body P is contained in the dilatation of t by an expansion factor of at most* $9/4$.

From [16], we know that the convex hull of the set of integer points in the plane can be computed in linear time. Thus, we can operate on $conv(K)$ without damaging the overall time complexity. Moreover, we know from [7] that there exists a greatest area triangle included in K whose vertices correspond to vertices of K. So, let T denote the dilatation of a maximal area triangle $t = ABC$ of K by a factor of $9/4$. The triangle T corresponds to a solution of our surrounding polygon problem. Indeed, if we set the origin at the point A, its three vertices are of the form $\alpha_{i1}AB + \alpha_{i2}AC$ with $|\alpha_{i1}|$ and $|\alpha_{i2}|$ bounded by 1 for $1 \leq i \leq 3$. So, the first approach we propose consists in computing a maximum area triangle of K and then consider its dilatation with a factor $9/4$. Boyce et al. propose an overall method to compute a greatest area k-gon which runs in $O(kn \log n + n \log^2 n)$ time and in $O(n \log n)$ time when $k = 3$ (see [3]). Dobkin et al. focus on the two-dimensional case and they show that the computation of a greatest area triangle runs in linear time performing at most $10n$ triangle area computations (see [7]). Their method consists in "walking" along the convex polygon K and determining for each vertex v two other vertices a and b such that the triangle abv has a maximal area. Their method runs in linear time because the three vertices move in clockwise order and they never have to backtrack during the traversal (see [7] for more details).

A Simpler Approach. We introduce a simpler approach to compute a surrounding polygon that only requires $2n$ distance computations. Moreover, it does not need the computation of the convex hull of K. This method consists in computing a surrounding parallelogram. Let A and B denote the leftmost and the rightmost point of K respectively according to the x-axis. Let N denote the normal vector of AB with positive y-coordinate. Let C and D denote the extremal point of K in the direction N and $-N$ respectively. We show afterwards that the set K is contained in a parallelogram of sides CD and $2AB$. This parallelogram corresponds to a solution of our problem because its vertices can be expressed as $\alpha_1 CD + \alpha_2 AB$ where $|\alpha_1|$ and $|\alpha_2|$ does not exceed 1 if the origin is well chosen.

Let $EFGH$ denote the parallelogram bounded by the two vertical straight lines passing through A and B and bounded by the two straight lines of direction vector AB passing trough C and D. By construction, the set K is included in the parallelogram $EFGH$, but the vertical sides of this parallelogram cannot be expressed using vectors rooted in points of K. As a result, we try to minimize δ such that $EFGH$ is contained in a parallelogram of side CD and of side δAB. We show that δ equals 2. Indeed, the points A and B may not be extremal according to the normal vector of CD and in this case, δ is strictly greater than 1 (see Fig. 2.a). The "worst case" happens when the points C and D coincide with opposite vertices of the parallelogram $EFGH$, (as in Fig. 2.b). In this case, we notice that half of the parallelogram $EFGH$ is included in a parallelogram

of side AB and CD. As a consequence, it is sufficient to double the side AB of the parallelogram such that the new parallelogram becomes large enough.

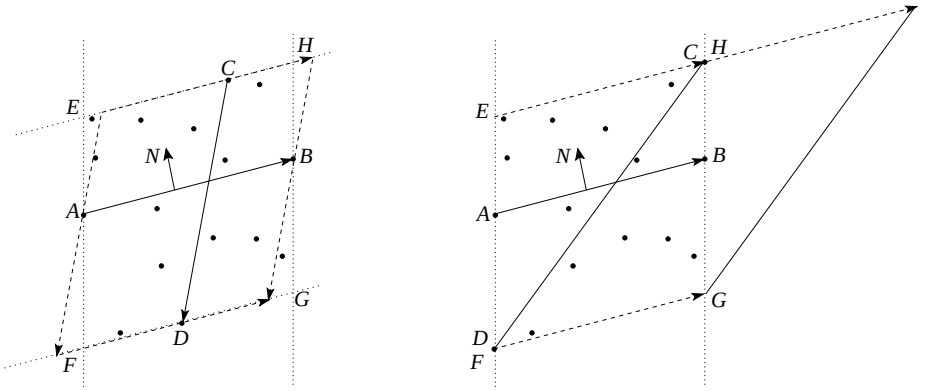

Fig. 2. Construction of the parallelograms

These two methods run in linear time. Unfortunately, they are not easily extensible to the three-dimensional case (see [22]). As a result, we describe in the next section a greedy method to compute a surrounding tetrahedron. This method can be extended to higher dimension but we focus on the three-dimensional case for convenience of presentation.

5.2 A Greedy Approach

We introduce the following definition:

Definition 1. *Let K denote a three-dimensional set of points. A tetrahedron G whose vertices belong to K is a* maximal growing tetrahedron *relative to K if for each face f of G, the furthest point from f in the set K corresponds to a vertex of G.*

Notice that a maximal growing tetrahedron does not always correspond to a greatest volume tetrahedron. Such a tetrahedron can be stated using three vectors a_1, a_2 and a_3, each generated by two points of K. We show afterwards that we can compute a surrounding parallelepiped Γ of K such that each vertex γ_i of Γ satisfies $\gamma_i = \sum_{1 \leq j \leq 3} \alpha_{ij} a_j$ where the values $|\alpha_{ij}|$ are bounded by 1 when the origin is well chosen.

First, we compute a maximal growing tetrahedron G using a non-deterministic but efficient approach. Moreover, this method easily extends to the d-dimensional case contrary to the previous deterministic approaches. Let $(f_l)_{1 \leq l \leq 4}$ and $(v_l)_{1 \leq l \leq 4}$ denote the faces and the vertices of G respectively such that the vertex v_l is opposite to the face f_l for $1 \leq l \leq 4$. We first initialize G using four non-coplanar

points of K. Our method consists in "pushing" the vertices of G among the points of K until G corresponds to a maximal growing tetrahedron. Repeatedly, for each face f_l of G, the algorithm tries to grow the tetrahedron G by moving the current vertex v_l to a point of K which is the furthest point from f_l. The method stops when no more growing can be done. This approach always finds a valid solution because at each step the volume of the current tetrahedron increases at least by one and this value is bounded by the volume of $conv(K)$. An overview of the method follows:

```
MAXIMAL GROWING TETRAHEDRON ALGORITHM:
Entries: K = {k_1, k_2, ..., k_n}, G = (v_1, v_2, v_3, v_4)
1 DO
2    STABLE ← true;    l ← 1;
3    WHILE STABLE AND l ≤ 4
4        IF (FurthestPoint(K, f_l) = v_l)
5           THEN l ← l + 1
6           ELSE STABLE ← false;
7 UNTIL STABLE = true
```

Let us move the origin O to v_1, this transformation is always possible because the lattice width is invariant under translation. Let $(a_j)_{1\leq j\leq 3}$ denote the three vectors defined by $v_{j+1} - v_1$ where $(v_j)_{1\leq j\leq 4}$ denote the vertices of a maximal growing tetrahedron G of K. We show that the set K is contained in a parallelepiped Γ whose vertices $(\gamma_i)_{1\leq i\leq 8}$ are of the form $\gamma_i = \sum_{1\leq j\leq 3} \alpha_{ij} a_j$ where each $|\alpha_{ij}|$ is bounded by 1. As the maximal growing tetrahedron we use is non-flat by definition, any point k_l of K can be written as $k_l = O + \sum_{1\leq j\leq 3} \delta_{lj} a_j$. Let us show that we have $|\delta_{ij}| \leq 1$ for $1 \leq i \leq n$, $1 \leq j \leq d$. Indeed, suppose that there exists an index i and a index j such that a $|\delta_{ij}|$ is strictly greater than 1. It would contradict the fact that G corresponds to a maximal growing tetrahedron because the vertex v_{j+1} would not be extremal. As a result the set K is contained in a parallelepiped Γ whose vertices $(\gamma_i)_{1\leq i\leq 8}$ are of the form $\gamma_i = \sum_{1\leq j\leq 3} \alpha_{ij} a_j$ where each $|\alpha_{ij}|$ is bounded by 1. Fig. 3 and Fig. 4 show examples in the plane and in the three-dimensional space respectively: G corresponds to a maximal growing triangle (resp. a maximal growing tetrahedron) and Γ corresponds to a surrounding parallelogram (resp. a surrounding parallelepiped). This method can be extended to higher dimension.

6 Conclusion

We have described in this paper a new algorithm to compute the lattice width. It runs in linear time relative to the size of K, which is optimal. Moreover, its principle is directly extensible to an arbitrary dimension even if intermediate constructions become more complex in that case. Our greedy approach simplifies greatly the application of this algorithm to an arbitrary dimension since it avoids

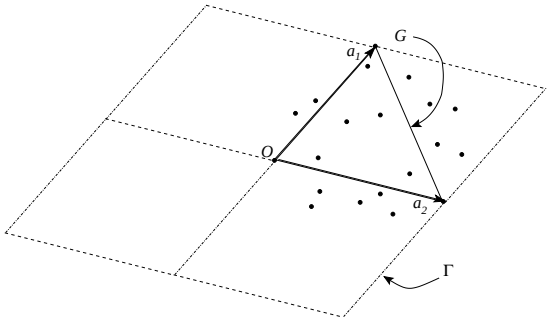

Fig. 3. Example of surrounding parallelogram

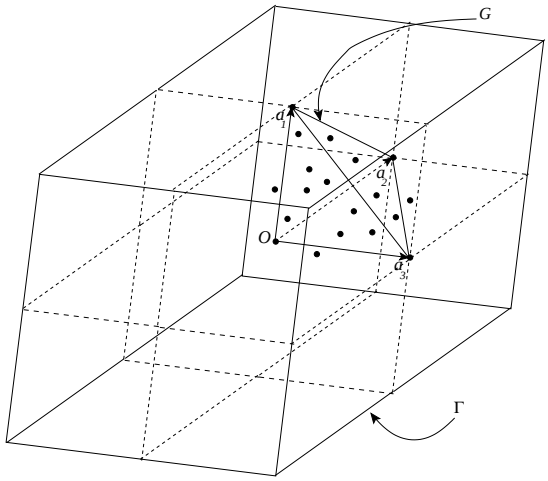

Fig. 4. Example of surrounding parallelepiped

the computation of an inscribed $d-$simplex of greatest volume which is a problem in $O(n^4)$ at least in the three-dimensional case for the best known algorithm [22]. This problem is an interesting problem in itself and will be the subject of future research since it is known that the dilation constant of this d-simplex is independent of K and thus it guarantees the smallest possible space search in any case. We also plan to extend the definition of linearity given in [11] to an arbitrary dimension. Moreover, we intend to test our method in order to describe its numerical behaviour. The optimal complexity of our new algorithm is a key point in that construction.

References

1. V.I. Arnold. Higher dimensional continued fractions. *Regular and chaotic dynamics*, 3:10–17, 1998.
2. A. Barvinok. *A Course in Convexity*, volume 54 of *Graduates Studies in Mathematics*. Amer. Math. Soc., 2002.
3. J.E. Boyce, D.P. Dobkin, R.L. Drysdale, and L.J. Guibas. Finding extremal polygons. In *STOC*, pages 282–289, 1982.
4. W. Cook, M. Hartman, R. Kannan, and C. McDiarmid. On integer points in polyhedra. *Combinatorica*, 12:27–37, 1992.
5. M. de Berg, O. Schwarzkopf, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 2000.
6. Isabelle Debled-Rennesson and Jean-Pierre Reveillès. A linear algorithm for segmentation of digital curves. *IJPRAI*, 9(4):635–662, 1995.
7. D.P. Dobkin and L. Snyder. On a general method for maximizing and minimizing among certain geometric problems. In *SFCS '79: Proceedings of the 20th Annual Symposium on Foundations of Computer Science*, pages 9–17, Washington, DC, USA, 1979. IEEE Computer Society.
8. F. Eisenbrand and S. Laue. A linear algorithm for integer programming in the plane. *Math. Program. Ser. A*, 102:249–259, 2005.
9. F. Eisenbrand and G. Rote. Fast 2-variable integer programming. In K. Aardal and B. Gerards, editors, *Integer Programming and Combinatorial Optimization*, volume 2081 of *LNCS*, pages 78–89. Springer-Verlag, 2001.
10. Fabien Feschet. The exact lattice width of planar sets and minimal arithmetical thickness. In R. Reulke, U. Eckardt, B. Flach, U. Knauer, and K. Polthier, editors, *IWCIA*, volume 4040 of *Lecture Notes in Computer Science*, pages 25–33. Springer, 2006.
11. Fabien Feschet. The lattice width and quasi-straightness in digital spaces. In *19th International Conference on Pattern Recognition (ICPR)*, pages 1–4. IEEE, 2008.
12. R. Fleischer, K. Mehlhorn, G. Rote, E. Welzl, and C.-K. Yap. Simultaneous inner and outer approximation of shapes. *Algorithmica*, 8(5&6):365–389, 1992.
13. W. Harvey. Computing two-dimensional Integer Hulls. *SIAM Journal on Computing*, 28(6):2285–2299, 1999.
14. D. Hirschberg and C.K. Wong. A polynomial-time algorithm for the knapsack problem with two variables. *J. Assoc. Comput. Mach.*, 23:147–154, 1976.
15. M.E. Houle and G.T. Toussaint. Computing the width of a set. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 10(5):761–765, 1988.

16. A. Hübler, R. Klette, and K. Voss. Determination of the convex hull of a finite set of planar points within linear time. *Elektronische Informationsverarbeitung und Kybernetik*, 17(2/3):121–139, 1981.
17. M. Kaib and C.-P. Schnörr. The Generalized Gauss Reduction Algorithm. *Journal of Algorithms*, 21(3):565–578, 1996.
18. R. Kannan. A polynomial algorithm for the two variable integer programming problem. *J. Assoc. Comput. Mach.*, 27:118–122, 1980.
19. G. Lachaud. Klein polygons and geometric diagrams. *Contemporary Math.*, 210:365–372, 1998.
20. G. Lachaud. Sails and klein polyhedra. *Contemporary Math.*, 210:373–385, 1998.
21. H.W. Lenstra. Integer Programming with a Fixed Number of Variables. *Math. Oper. Research*, 8:535–548, 1983.
22. S.I. Lyashko and B.V. Rublev. Minimal ellipsoids and maximal simplexes in 3D euclidean space. *Cybernetics and Systems Analysis*, 39(6):831–834, 2003.
23. J.-P. Reveillès. *Géométrie discrète, calcul en nombres entiers et algorithmique*. Thèse d'etat, Université Louis Pasteur, Strasbourg, France, 1991.
24. G. Rote. Finding a shortest vector in a two-dimensional lattice modulo m. *Theoretical Computer Science*, 172(1-2):303–308, 1997.
25. H.E. Scarf. Production sets with indivisibilities part i and part ii. *Econometrica*, 49:1–32, 395–423, 1981.
26. A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley and Sons, 1998.