Approximation of a direction of $\mathbb{N}^d$ in bounded coordinates

Jean-Christophe Novelli [a] Gilles Schaeffer [b] Florent Hivert [a]

[a] *Université Paris 7 – LIAFA 2, place Jussieu - 75251 Paris Cedex 05 – Fax : (33) (0)1.44.27.68.49 – France;*
E-mail: novelli@liafa.jussieu.fr

[b] *École Polytechnique – LIX 91128 Palaiseau Cedex – France;*
E-mail: schaeffe@lix.polytechnique.fr

In this paper, we address the following problem: a direction in $\mathbb{R}^d$ is given by a vector with large integral coordinates and we have to find a vector with bounded integral coordinates which realizes a good approximation of this direction. We present a short algorithm that efficiently computes an optimal approximation with respect to the distance to the direction. We then compare experimental results with that of a straightforward rounding algorithm with respect to the distance to the direction and to the angle with it.

1. Motivations

The Global Standard for Mobility (GSM) (see, e.g., [6], [4], [5]) yields a number of specific algorithmic problems that can be solved using combinatorics, discrete mathematics or discrete modeling. The frequency assignment problem is the most classical of these problems. Given a GSM network, one wants to assign the frequencies of the Base Transceiver Stations (BTS) in such a way that (adjacent and cochannel) interferences are minimized. Several methods using graph theoretical approaches, simulated annealing or tabu research were proposed in the literature to solve this last problem (see, e.g., [3], [2]). There are lots of other discrete problems that occurs in the context of GSM. To give another interesting example, one can consider the problem of minimizing the cost of the interconnections between the networks of two different GSM providers. Such a problem involves for instance game theory, optimization research and routing strategies.

We address in this paper another type of discrete problem related with optimal quantization. This last problem is a very old and classical problem that may be considered as a problem of minimizing certain distances on the sphere. One wants to put p points on the sphere of dimension d so that the maximal distance from a generic point of the sphere to the set of distinguished points is minimal. Many partial solutions were already proposed, but no satisfying real-time algorithm was given for this problem in all its generality. The specific problem we consider here will be presented in the next section. Different algorithms are proposed and compared in the other sections of this paper.

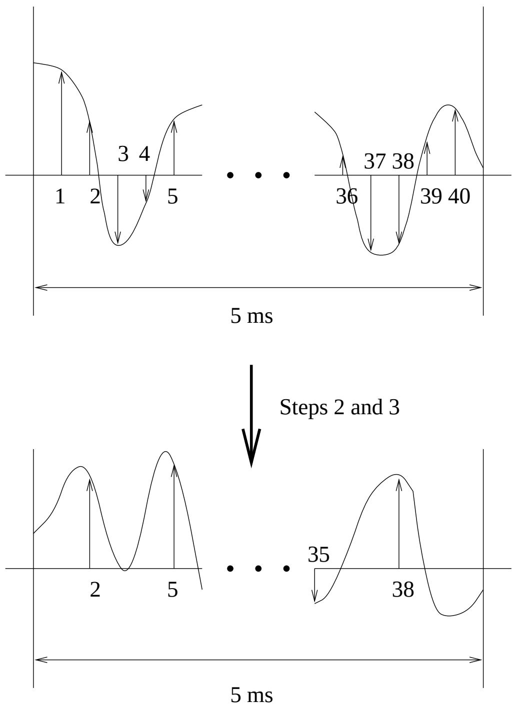
2. Presentation of the problem

Here we focus on a discrete problem occurring in the context of the GSM vocoder optimization: the following protocol can be considered as a good model of the real protocol used by the GSM vocoder in order to encode the white noise coming from the voice signal in digital form:

- Step 1. Cut the white noise signal into 5ms slices and sample each slice at a 8 kHz frequency, which yields 40 samples to process at each step,
- Step 2. Send the difference through a high-pass filter and cut the 40 samples up into 3-sized chunks, dropping the last sample, and keep only one sample of each chunk; this yields a vector in $\mathbb{R}^{13}$,
- Step 3. Encode the scale of the vector by looking up its largest coordinate on a log scale,
- Step 4. Encode in the best possible way the direction of the vector,
- Step 5. Send the encoded signal to the corresponding base station.

3. The algorithmic problem

Our aim is to find out some solutions to Step 5 of the previously described protocol and then compare the quality of their approximation. At this stage of the process, we still have 39 bits left to encode the information. Since we cannot change the decoder, we have to work with the actual interpretation of these 39 bits. Traditionally, these bits are used to encode a point in the lattice



$\{-7,-5,-3,-1,1,3,5,7\}^{13}$. The algorithmic problem associated with Step 5 is therefore to find a good approximation of a vector of $\mathbb{R}^{13}$ (or equivalently, a half-line rooted at the origin) by a correct choice of a point of the lattice.

The quality of the approximation can be evaluated by several means. The most natural approach is the *angle problem*: find a vector in the bounded lattice that minimizes the angle with the actual direction. An easier approach is the *distance problem*: find a point in the bounded lattice that minimizes the distance to the half-line. We give an algorithm to solve exactly and efficiently the distance problem and we conjecture, on the basis of strong numerical evidences, that our algorithm gives a very close approximation to the solution of the angle problem. In fact, we believe that, up to side effects, the optimum is always returned.

In dimension 2, the angle problem is immediately equivalent to the following one: given a rational number r/s, find a rational r'/s' such that $r' \leq p$ and $s' \leq p$ that is as close as possible to r/s. In this case, an optimal pair (r', s') can be characterized with the help of Sturmian words and computed in logarithmic time by Berstel's algorithm (see [1]). In the 3-dimensional space, many algorithms were proposed but they are not very efficient (see, e.g., [7]) and, as far as we know, there is no interesting extension of Sturmian words. So we are left with two possibilities: either continue looking for an optimal solution by performing an exhaustive search of the space of possible vectors or make choices to obtain a hopefully good approximation quickly. In Section 4, we give some notations and then prove that, in the distance problem, the space of search can be highly reduced. This allows us to present an efficient optimal approximation algorithm in Section 5.1. In Section 5.2, we describe a very simple rounding algorithm and we compare experimentally the quality of its approximation with the optimal one in Section 6. Finally, in Section 7, we discuss a third measure of the quality that is closely related to the simple rounding algorithm.

4. Basic ingredients

In this section, we give notations and definitions that will be useful in the sequel and we state the theorem on which our algorithm relies.

Consider the lattice $\mathbb{N}^d$ in $\mathbb{R}^d$ ($d \geq 2$) and let $\mathcal{E} = \{0, \ldots, p\}^d \setminus \{(0^d)\}$ be the sublattice of $\mathbb{N}^d$ of points whose coordinates are bounded by p (except the zero element). Let $\mathcal{D}$ denote the half-line starting at the origin and directed by a given vector $Y = (y_1, \ldots, y_d)$ with non negative integer coordinates. We want to find an element of $\mathcal{E}$ that realizes the minimal distance to $\mathcal{D}$. In order to reduce the space of search, we prove that the optimum belongs to a small set S of points of $\mathcal{E}$. First, notice that the hyperplanes $x_i = k - 1/2$ for $0 \leq k \leq p+1$ and $1 \leq i \leq d$ divide the space into bounded regions that are open hypercubes and unbounded regions. Moreover, each of the open hypercubes contains exactly one point of $\mathcal{E}$ that is its center.

Theorem 4.1. The set of minimizing points is contained in the set S of centers of open hypercubes that intersect $\mathcal{D}$.

Proof — Consider a point t of $\mathcal{D}$: if t is inside an hypercube, then its closest integral point is the center of this hypercube. Otherwise, if t belongs to at least one hyperplane the minimal distance cannot be reached. Indeed, in this case, t is equidistant to the centers of all the incident hypercubes and the left or the right neighbours of t on $\mathcal{D}$ are inside an hypercube and closer to the associated center. Finally, if t is in an unbounded region, the minimal distance cannot be reached as well: the only case to consider is the case of the first unbounded region met by $\mathcal{D}$ and in this case, the orthogonal projection of the nearest point of $\mathcal{E}$ (it is well defined except in some special cases where the proof is obvious) on $\mathcal{D}$ does not belong to this unbounded region, so that t does not realize the minimum. □

Since each hypercube is convex, its intersection with $\mathcal{D}$ is an interval. These intervals are disjoint and hence totally ordered on $\mathcal{D}$, therefore S is totally ordered. By construction, each coordinate of two consecutive elements may differ by at most 1. Now, if two consecutive elements of S differ on several coordinates, we arbitrarily insert intermediate points in S so that in the final set S two consecutive elements differ exactly by the increment of one coordinate.

The reduction of the search space to S allows us to give an efficient algorithm to compute a minimum since the size of S is small (at most $d \cdot p$). This is done in the next section.

5. The algorithms

In this section, we first define an algorithm that finds an element of $\mathcal{E}$ out that minimizes the distance to $\mathcal{D}$. We then define a simpler algorithm to compute very quickly an approximation of the optimum.

5.1. The optimal algorithm

This algorithm first computes the set S: since each element of S is obtained by incrementing one coordinate of its predecessor, we can represent S by the sequence s of these elementary increments. Moreover, we can derive the distance of an element of S to $\mathcal{D}$ from the distance of the previous one to $\mathcal{D}$ so that the computation requires only a few additions and multiplications over $\mathbb{N}$. The half-line $\mathcal{D}$ is given by a point (the origin) and a vector Y so that we have a natural parameterization of it: $OM = t \cdot Y$. We shall therefore denote points of $\mathcal{D}$ by their parameter t.

Algorithm 1. 1 `INPUT: A vector` Y `of` $\mathbb{N}^d$.
`OUTPUT: A vector of` $\mathcal{E}$.

Preliminary step: Put $u = (0, \ldots, 0)$ (u shall run through S), $r = 0$ (r is the parameter associated with the orthogonal projection of u on $\mathcal{D}$), $dist = 0$ ($dist$ is the distance between u and $\mathcal{D}$), $m = \infty$ (m is the minimum), $M = u$ (M is the element of $\mathcal{E}$ where the minimum is obtained).

Step 1: Compute for each $1 \leq i \leq d$, the sequence of the values of the parameter t at which $\mathcal{D}$ intersects the hyperplanes $x_i = k + 1/2$ for $0 \leq k \leq p$.

Step 2: Compute the sequence s of elementary increments that define S by shuffling together the sequences obtained at Step 1, according to increasing values of the parameter.

Step 3: While u is in $\mathcal{E}$:

- Increment the correct coordinate of u according to s,
- Incrementally compute r and $dist$,
- If $dist \leq m$, put $m = dist$, and $M = u$.

Step 4: The outcome is M.

We shall give the complexity of this algorithm in the next section.

5.2. The approximation algorithm

This algorithm computes an integer-valued vector whose distance to $\mathcal{D}$ is a good approximation of the minimum distance of $\mathcal{E}$ to $\mathcal{D}$.

Algorithm 2. 2 `INPUT: A vector` $Y = (y_1, \ldots, y_d)$ `of` $\mathbb{N}^d$.
`OUTPUT: A vector of` $\mathcal{E}$.

Step 1: Find the greatest coordinate y of Y.

Step 2: For all $0 \leq i \leq d$, compute the nearest integer x_i to $y_i \cdot p/y$, i.e., $x_i = \lfloor y_i \cdot p/y + 1/2 \rfloor$.

Step 3: The outcome is $X = (x_1, \ldots, x_d)$.

6. Comparison of both algorithms

In this section, we compare the complexity of both algorithms and give the experimental average difference between both algorithms in the case $n = 13$ and $p = 7$.

6.1. Theoretic complexity

We first give upper bounds for each step of Algorithm 1:

Step 1: d divisions, $d \cdot p$ additions.

For each i, one floating point division and p additions.

Step 2: $d^2 \cdot p$ comparisons.

Each element of the sequence s is chosen among d possibilities.

Step 3: $d \cdot p \cdot T$ multiplications, $d \cdot p \cdot A$ additions.

Each loop requires one comparison and constant numbers of multiplications and additions.

The total complexity is therefore $O(d^2 p)$ comparisons and $O(dp)$ arithmetic operations. The space needed is $O(dp)$. However, the average complexity is much lower: indeed, the factor dp is the number of elements of S in the worst case. The exact number of elements of S is $\sum_{i=1}^{d} \lceil p \cdot y_i / y \rceil$ where y is the largest coordinate of Y. In particular, when k coordinates strongly dominate the vector, the complexity is only $O(kp)$.

Algorithm 2 requires d comparisons, d additions and d divisions. Looking at both complexities, one can do the following remark: both algorithms are real time but the second one uses much less memory. If one has a lot of memory, one should preferably use Algorithm 1 since the result is exact.

6.2. Experimental observations

We did most experimental computations in the case $n = 13$ and $p = 7$, which is of special interest since it is this lattice that is used in practice in the GSM vocoder (see Sections 2 and 3). We give first some results for the distance problem, under uniform independent distributions of the coordinates of Y $(0 < y_i < 2^{32})$:

	Algorithm 1	Algorithm 2	Ratio 1/2
Average distance	.559	.8535	.655
Average number of tested points	46	1	46
Average time	1.4	.267	5.24

Notice that the maximum distance is $\sqrt{13}/2 \approx 1.8$ that is much greater than the average distance. Quite naturally the optimal algorithm is better for this problem. More interesting is the fact that the complexity of the optimal algorithm is only approximately five times the complexity of the rounding algorithm. The number of tested points is much greater but since we compute iteratively the distance to the half-line, the computation is very quick.

The optimal algorithm can be adapted to the angle problem: indeed the tangent of the angle can easily be incrementally computed as the distance. The complexity is of the same order (up to constant factor since the incremental formula is not the same) and the experimental results we obtained are then given below:

	Algorithm 1	Algorithm 2	Ratio 1/2
Average tan	.026	.032	.81

These results are interesting since they establish that the algorithm used in practice can be improved by a factor around 20%.

7. Another distance

In this section, we define a third measure of the quality of the approximation that is very close to the angle problem. First, define H_k as the hyperplane defined by the equation $x_k = p$ where k is the index of the greatest coordinate of Y. For all the points z of the lattice, we denote by z' the intersection of the line containing the origin and z with H_k and we define the optimum point as the point of the lattice that realizes the minimal distance from z' to the intersection of $\mathcal{D}$ with H_k.

The algorithm that is used in practice is a variation of Algorithm 2. Instead of fixing the greatest coordinate of Y to p, we fix successively this coordinate to the integer values from $\frac{p}{2}$ to p, then choose the other values of the vector as before and take the nearest point among all this set of points. This algorithm will be referred to as Algorithm $2'$. The following theorem establishes why Algorithm 2' is also good in practice.

Theorem 7.1. Taking the previous definition of the optimum, Algorithm $2'$ outputs this optimum.

Proof — It suffices to see that each half-line that contains a point on the lattice also contains a point on the lattice the greatest coordinate of which is strictly greater than $p/2$ and that, having fixed the greatest coordinate to an integer between $p/2$ and p, the chosen point minimizes the distance to the intersection of $\mathcal{D}$ with H_p. $\square$

Conclusion

In this paper, we present a new efficient algorithm to look for bounded approximations of a direction and we compare this algorithm with existing ones.

Our algorithm returns an optimal solution according to the distance and numerical results suggest that the approximation is very good with respect to the angle. In the GSM vocoder case, our algorithm is at least as efficient as the existing one since the computations are done incrementally and thanks to the type of distribution involved. The expected gain with respect to the algorithm used in practice is around 20%. But there remains lots of open questions related to this problem. It seems to us that the most interesting one is the problem of the optimization when we can change the coder and the decoder, *i.e.,* the optimal quantization problem. This problem consists in placing a set of points on the sphere of dimension 13 in such a way that one can easily compute the nearest point of this set to a given point and that the maximum distance from all points to their nearest point is minimum.

References

[1] J. Berstel, A. De Luca, *Sturmian words, Lyndon words and trees*, Theoretical Computer Science, **178**, 171–203, 1997

[2] M. Duque-Antón, D. Kunz, B. Ruber, *Channel Assignment for Cellular Radio Using Simulated Annealing*, IEEE Trans. Veh. Tech., **42**, (1), 14–21, 1993

[3] W. K. Hale, *Frequency Assignment: Theory and Applications*, Proc. IEEE, **68**, 1497–1514, 1980

[4] X. Lagrange, P. Godlewski, S. Tabbane, *Réseaux GSM–DCS*, Hermès, 1995

[5] V. H. MacDonald, *Advanced Mobile Phone Service: The Cellular concept*, Tech. J., **58**, 15–41, 1979

[6] M. Mouly, M.-B. Pautet, *The GSM System for Mobile Communication*, 1993

[7] W.F. Rogers, *Procedural elements for Computer Graphics*, McGraw-Hill, New-York, 1985