Available online at www.sciencedirect.com
ScienceDirect
Theoretical Computer Science 381 (2007) 68–85
Theoretical Computer Science
www.elsevier.com/locate/tcs

Optimal routing in double loop networks

Domingo Gómez, Jaime Gutierrez[*], Álvar Ibeas

Faculty of Sciences, University of Cantabria, E–39071 Santander, Spain

Received 30 October 2005; received in revised form 12 January 2007; accepted 3 April 2007

Communicated by D. Peleg

Abstract

In this paper, we study the problem of finding the shortest path in circulant graphs with an arbitrary number of jumps. We provide algorithms specifically tailored for weighted undirected and directed circulant graphs with two jumps which compute the shortest path. Our method only requires $O(\log N)$ arithmetic operations and the total bit complexity is $O(\log^2 N \log\log N \log\log\log N)$, where N is the number of the graph's vertices. This elementary and efficient shortest path algorithm has been derived from the Closest Vector Problem (CVP) of lattices in dimension two and with an ℓ_1 norm.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Double loop networks; Cayley graphs; Shortest path; Lattices; Closest Vector Problem

1. Introduction

A *directed circulant graph* or *m-loop network* $C_N(j_1, j_2, \dots, j_m)$ with N vertices, labeled with $\mathbb{Z}_N$, the integers modulo N, and jumps $j_1, j_2, \dots, j_m$ is a graph in which each vertex n, $0 \leq n \leq N-1$, is adjacent to all the vertices $n + j_i \bmod N$, with $1 \leq i \leq m$. In contrast, an *undirected circulant graph* or $2m$*-loop network* $C_N(\pm j_1, \pm j_2, \dots, \pm j_m)$ with N vertices, and jumps $j_1, -j_1, j_2, -j_2, \dots, j_m, -j_m$ is a graph in which each vertex n, $0 \leq n \leq N-1$, is adjacent to all the vertices $n \pm j_i \bmod N$, with $1 \leq i \leq m$ (see Fig. 1). In fact, they are *Cayley graphs* of the finite cyclic group $\mathbb{Z}_N$. Throughout the paper we employ the term circulant graph or multi-loop network for both undirected and directed circulant graphs.

Multi-loop networks were first proposed in [33] for organizing multimodule memory services and have a vast number of applications in telecommunication networking, VLSI design, and distributed computation. Their properties, such as diameters and reliabilities, have been the focus of lots of research in computer network design, see for instance [3–5,8,10,12,13,22,28,35].

Every circulant graph can be associated to a lattice $\mathcal{L}$ which consists of the integer solutions $(x_1, \dots, x_m) \in \mathbb{Z}^m$ to the congruence:

$$j_1x_1 + \cdots + j_mx_m \equiv 0 \bmod N. \tag{1}$$

[*] Corresponding author. Tel.: +34 942201523.
E-mail address: jaime.gutierrez@unican.es (J. Gutierrez).

0304-3975/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2007.04.002

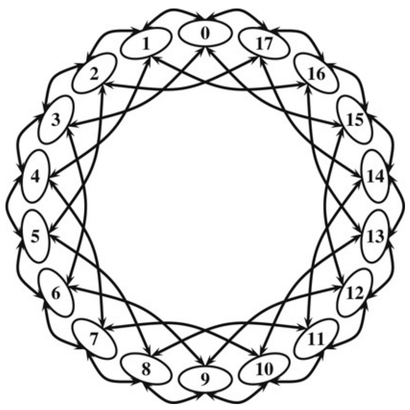

Fig. 1. $C_{18}(\pm 1, \pm 3)$.

Given two vertices r and s, a path from r to s for an undirected circulant graph $C_N(\pm j_1, \pm j_2, \dots, \pm j_m)$ can be described by an integer vector $\mathbf{x} = (x_1, \dots, x_m) \in \mathbb{Z}^m$ such that

$$\sum_{i=1}^{m} x_i j_i \equiv s - r \bmod N. \tag{2}$$

And a shortest path $\mathbf{x}$ is a path with minimum ℓ_1 norm. By contrast, a path from r to s for the directed circulant graph $C_N(j_1, j_2, \dots, j_m)$ can be described by an integer positive vector $\mathbf{x} = (x_1, \dots, x_m) \in \mathbb{N}^m$ satisfying Eq. (2), and a shortest path $\mathbf{x}$ is a path with minimum ℓ_1 norm.

Message routing is a basic function in communication networks. The problem in message routing is to find a route along which messages should be sent. The routing algorithm dictates token passing strategies and information on distributed networks. Of course, for every path $\mathbf{x}$ from vertex r to s we can design several message routing from source r to destination s. It is called an *optimal message routing* if every message is sent along a shortest path from its source vertex to its destination vertex.

For general graphs, finding a shortest path between two vertices is a well-known and important problem. Efficient polynomial time algorithms have been developed for various routing problems. However, for the family of circulant graphs, there is an important distinction to be made, and that concerns the natural input size to a problem. For an arbitrary graph, it is common to consider the input size to be $O(N^2)$, which is the number of bits in its adjacency matrix. However, any circulant graph can be described by only m integers. In this representation, the input size is $O(m \log N)$. Thus, polynomial time algorithms for general graphs may exhibit exponential complexity in the special case of circulant graphs, for this compact input representation.

The single-loop network (also called a ring network) is mathematically trivial. The case $m = 2$ is the most important and most studied multi-loop network, that is, undirected circulant graphs of degree four or distributed double loop networks and directed circulant graphs of degree two or double loop networks. They have been extensively studied (see the surveys [4,21]). When N is given as an unary input and the time complexity is measured in these terms, there are several shortest path algorithms for directed circulant graph of degree two, see for instance [8,20,21,30,34]. Typically, they dynamically compute the next vertex the message is to be sent to, and show that the route a message traverses is indeed a shortest path from its source to its destination. So, as a consequence they obtain algorithms that require $O(N)$ arithmetic steps or $O(\log N)$ time for preprocessing and constant processing time at each node on the route. But the lower bound of the diameter for circulant graphs and directed circulant graphs is $\Omega(\sqrt{N})$ (see [4]). So in both cases, they are exponential in the input size $\log N$. The paper [7] shows an algorithm to compute a shortest path in the circulant graph $C_N(\pm 1, \pm h)$ (the so called chordal ring graphs) in $O(h/g + \log h)$ time, where $g = \gcd(h, N)$. Obviously, it can also exhibit exponential time complexity for this input measure.

In [6], the authors establish relations between several routing applications for undirected circulant graphs and the problem of finding the shortest vector (with ℓ_1 norm) in the above lattice. They show that the shortest path problem is NP-hard for this concise representation.

Throughout the paper, the term polynomial time algorithm means polynomial in $m \log N$.

We remark in Section 2 that using the well-known extended Euclidean algorithm, we can compute on polynomial time a path $\mathbf{c}$, not necessarily a shortest one, in an undirected circulant graph, then the problem of finding a shortest path is equivalent to the problem of finding a vector that is closest to $-\mathbf{c}$ in the lattice defined by Eq. (1) with respect to ℓ_1 norm. The paper [24] presents an algorithm for solving the Closest Vector Problem in a lattice with respect to ℓ_1, ℓ_2 and ℓ_∞ norms. Moreover, fixed the lattice dimension, this algorithm is polynomial in the bit-size of the lattice basis.

On the other hand, we also remark in Section 2 that the problem of finding a shortest path for a directed circulant graph is equivalent to the problem of solving a suitable integer program with $m + 2$ constraints associated to the circulant graph. The paper [25] presents a polynomial time algorithm in fixed dimension for solving an integer program which is defined by a fixed number of constraints. So, fixed the number of jumps m, papers [24] and [25] provide polynomial time algorithms of finding a shortest path in undirected and directed circulant graphs.

So, using Integer Programming and/or Closest Vector Problems, we derive polynomial time algorithms for computing a shortest path in circulant graphs defined by a fixed number of jumps. In accordance with previous paragraphs, this simple observation could be considered as a minor contribution of the present paper. The problem with those related general methods is that sometimes they are quite involved and difficult to implement.

In this article, we give polynomial time deterministic algorithms to compute a shortest path between two vertices in any weighted circulant graph with N vertices and two jumps. Our algorithms are simple to implement, the proofs are also quite elementary, and they require $O(\log N)$ arithmetic steps and the total bit operations is $O(\log^2 N \log\log N \log\log\log N)$. Both of them are based on Closest Vector Problems for ℓ_1 norm.

The paper is divided into six sections. In Section 2 we introduce some known lattice and Integer Programming concepts and show the relation between the shortest path problem for circulant graphs and these well known mathematical problems. Section 3 is devoted to describe an algorithm for solving the two dimensional CVP for ℓ_1 norm. As a consequence of the two previous sections, we obtain an algorithm for finding a shortest path in any undirected circulant graph of degree four. Section 4 is dedicated to finding a shortest path in double loop networks. Then, Section 5 analyzes the problem in weighted circulant graphs. Finally, in Section 6 we present our conclusions and some open problems.

2. A general method of finding a shortest path in circulant graph

In this section, we point out that a polynomial time algorithm to compute a shortest path for a circulant graph defined by a fixed number of jumps m can be derived from existing general methods.

The vertex-symmetry of circulants allows their analysis starting from any vertex, which simplifies their study. We may assume the routing is from vertex 0 to vertex $j \in \mathbb{Z}_N$. Using the well-known Extended Euclidean Algorithm, we compute a path from 0 to vertex j if it exists. A necessary and sufficient condition for such path to exist is $\gcd(j_1, j_2, \ldots, j_m, N)$ divides j. In fact, the circulant graph is connected if and only if $\gcd(j_1, j_2, \ldots, j_m, N) = 1$.

We separately analyze the directed and undirected cases.

2.1. Integer programming techniques

Integer Programming is an expressive framework for modeling and solving discrete optimization problems that arise in a variety of contexts in the engineering sciences. The reader is referred to the book [31] and its numerous references. Integer Programming representation works with implicit algebraic constraints (linear equations and inequalities on integer valued variables) to capture the feasible set of alternatives and linear objective functions (to minimize or maximize over the feasible set) that specify the criterion for defining optimality.

More precisely, given an integral matrix $A \in \mathbb{Z}^{k\times n}$ and integral vectors $\mathbf{b} \in \mathbb{Z}^k$, $\mathbf{d} \in \mathbb{Z}^n$, determine $\mathbf{x}$ such that:

$$\mathbf{d}\cdot\mathbf{x} = \min\{\mathbf{d}\cdot\mathbf{x} : A\mathbf{x} \geq \mathbf{b}, \mathbf{x} \in \mathbb{Z}^n\}.$$

It is well known that Integer Programming is NP-hard. The situation changes drastically if the number of variables or the dimension is fixed. For this case papers [25,24] showed that the above problem can be solved in polynomial time, see also [9]. The following result appears in [11]:

The above integer program in fixed dimension n, where the objective vector and each of the constraints of $A\mathbf{x} \geq \mathbf{b}$ have binary encoding length of at most s, can be solved with an expected amount of $O(k + s\log k)$ arithmetic operations on rational numbers of size $O(s)$.

We can apply this general technique of finding a shortest path for a directed circulant graph as follows:

Lemma 1. *Any shortest path between* 0 *and* j *in* $C_N(j_1, j_2, \ldots, j_m)$ *is also a solution to the following Integer Program:*

$$\min\{\mathbf{d} \cdot \mathbf{x} : A\mathbf{x} \geq \mathbf{b}, \mathbf{x} \in \mathbb{Z}^{m+1}\},$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_m, y) \in \mathbb{Z}^{m+1}$, $\mathbf{d} = (1, 1, \ldots, 1, 0) \in \mathbb{Z}^{m+1}$, $\mathbf{b} = (j, -j, 0, \ldots, 0) \in \mathbb{Z}^{m+2}$ *and*

$$A = \begin{pmatrix} j_1 & j_2 & \ldots & j_m & N \\ -j_1 & -j_2 & \ldots & -j_m & -N \\ 1 & 0 & \ldots & 0 & 0 \\ 0 & 1 & \ldots & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & 1 & 0 \end{pmatrix} \in \mathbb{Z}^{(m+2)\times(m+1)},$$

and conversely.

So as a consequence of this observation, we have that once the number of jumps m is fixed the paper [11] provides an algorithm to compute a shortest path in undirected circulant graphs requiring $O(m + \log m \log N)$ arithmetic operations on rational numbers of size $O(\log N)$. Another method to compute a shortest path using Groebner Basis is presented in papers [16,19].

In the following sections, we give an independent and efficient method for the case of double loop networks.

2.2. Lattice reduction techniques

The fundamental objects we are dealing with in this subsection are *lattices*, defined as discrete subgroups of the space $\mathbb{R}^m$. Equivalently, a lattice is the set of integer linear combinations of some linearly independent vectors. Lattices are geometric objects that have been used to solve many problems in mathematics and computer science, see for instance [2,18,15,26,27,29].

Let $\{\mathbf{b}_1, \ldots, \mathbf{b}_s\}$ be a set of linearly independent vectors in $\mathbb{R}^m$. The set

$$\mathcal{L} = \{c_1\mathbf{b}_1 + \cdots + c_s\mathbf{b}_s, c_1, \ldots, c_s \in \mathbb{Z}\}$$

is called an *s-dimensional lattice* with *basis* $\{\mathbf{b}_1, \ldots, \mathbf{b}_s\}$.

One basic lattice problem is the *Shortest Vector Problem* (SVP): given a lattice $\mathcal{L}$ and norm $\|\cdot\|$, it involves finding a nonzero lattice element with the smallest norm among all non-zero vectors in the lattice. Unfortunately, this problem is NP-hard (under randomized or non-uniform reductions, see [29]). This study has suggested several definitions of a *reduced* basis for a lattice. The following concept is the generalization of the Gauss reduced basis concept for lattices of rank 2 to an arbitrary norm [23].

Definition 2. A two-dimensional basis $\{\mathbf{u}, \mathbf{v}\}$ is called *reduced* or *Gauss-reduced* respect to a norm $\|\cdot\|$ if

$$\|\mathbf{u}\|, \|\mathbf{v}\| \leq \|\mathbf{u} + \mathbf{v}\|, \|\mathbf{u} - \mathbf{v}\|.$$

The algorithm in [23] computes a Gauss-reduced basis from a basis $\{\mathbf{u}, \mathbf{v}\}$ of the lattice $\mathcal{L} \subseteq \mathbb{Z}^2$ for any computable norm in $O(\log M)$ arithmetic operations where $M = \max(\|\mathbf{u}\|, \|\mathbf{v}\|)$ on integer numbers of size $O(\log N)$. Notice that, possibly, swapping $\mathbf{u}$ and $\mathbf{v}$, we can always assume that $\|\mathbf{u}\| \leq \|\mathbf{v}\|$. Then, we have that $\mathbf{u}$ is a shortest vector of the lattice $\mathcal{L}$:

$$\|\mathbf{u}\| = \min\{\|\mathbf{w}\|, \mathbf{w} \in \mathcal{L} - \{\mathbf{0}\}\}.$$

Moreover, $\mathbf{v}$ is shorter than any lattice vector lying in the space generated by the former, that is, $\|\mathbf{u}\|$ and $\|\mathbf{v}\|$ are the two successive *Minkowski minima*.

Another related problem for which also no polynomial time solution exists is the *Closest Vector Problem,* CVP:

```
**Input**: $\mathbf{u}, \mathbf{v} \in \mathbb{R}^2,\ \mathbf{v} \neq \mathbf{0}$.
**Output**: $\mathrm{Reduce}_{\mathbf{v}}(\mathbf{u}) \in \mathbf{u}+\mathbb{Z}\mathbf{v} \mid \|\mathrm{Reduce}_{\mathbf{v}}(\mathbf{u})\| = \min\{\|\mathbf{u}+\alpha\mathbf{v}\| \mid \alpha \in \mathbb{Z}\}$.
1 Select $i \in \{1,2\},\ j \in \{1,2\}\backslash\{i\}$ such that $|v_i| > |v_j|$. If $|v_1| = |v_2|$, then $i = 1$;
2 Return the vector with minimum norm between:
      $\mathbf{u} - \left\lfloor \frac{u_i}{v_i} \right\rfloor \mathbf{v},\quad \mathbf{u} - \left\lceil \frac{u_i}{v_i} \right\rceil \mathbf{v}$.
  If both have the same norm, return the one with $i$th non-negative component;
```

Algorithm 1. REDUCE procedure.

Definition 3. Given a basis B generating the lattice $\mathcal{L} \subseteq \mathbb{R}^m$, a vector $\mathbf{v} \in \mathbb{R}^m$, and a norm in $\mathbb{R}^m$; the *Closest Vector Problem* consists on finding a vector in the set $\mathbf{v} + \mathcal{L}$ with minimum norm.

This problem is equivalent to finding a vector $\mathbf{u}$ in $\mathcal{L}$ that minimizes the norm of $\mathbf{u} - (-\mathbf{v})$, i.e., looking for the closest vector to $-\mathbf{v}$ in $\mathcal{L}$.

Now, let $\mathbf{c} = (c_1, \ldots, c_m)$ be a path from 0 to vertex j, that is, a solution of the congruence equation

$$j_1x_1 + j_2x_2 + \cdots + j_mx_m \equiv j \bmod N.$$

We consider the lattice $\mathcal{L}$ given in Eq. (1), then we have the following observation:

Lemma 4. *Any shortest path between* 0 *and* j *in* $C_N(\pm j_1, \pm j_2, \ldots, \pm j_m)$ *is a solution of the CVP in the lattice* $\mathcal{L}$ *for the vector* $-\mathbf{c}$ *with norm* ℓ_1*, and conversely.*

It is well known that the CVP is NP-hard. However, for any fixed dimension, a CVP can be solved exactly in polynomial time for the Euclidean norm ℓ_2. The algorithm presented in [24] can also be adapted to find the ℓ_1-closest and the ℓ_∞-closest vectors. His general algorithm requires cubic polynomial time (polynomial in the bit-size of the lattice basis) for solving the two dimensional CVP with respect to ℓ_1 norm.

One of the problems of those related general techniques, that is, Integer Programming and the Closest Vector Problem, is that they are quite involved mathematically and difficult to implement. In the next sections, we provide a simple and elementary algorithm for circulant graphs defined by two jumps.

3. An algorithm for solving two dimensional CVP

The main problem addressed in this section is how to compute a vector that is closest to another given vector in a two dimensional lattice with respect to ℓ_1 norm.

For the rest of the paper, we only consider the ℓ_1 norm. We denote by $\|\cdot\|$ this norm acting over a vector. As we are dealing with vectors $\mathbf{u} \in \mathbb{R}^2$, we denote their components by $\mathbf{u} = (u_1, u_2)$.

For every real number $x \in \mathbb{R}$, as usual, we denote by $\mathrm{sgn}\,(x)$ its sign.

3.1. Reduction by a vector

Given $\mathbf{u}, \mathbf{v}$ in $\mathbb{R}^2$, with $\mathbf{v} \neq 0$, we can find $\alpha \in \mathbb{Z}$ such that the value $\|\mathbf{u} - \alpha\mathbf{v}\|$ is minimal, that is, we want to make $\mathbf{u}$ as short as possible by subtracting an integer multiple of $\mathbf{v}$ (see [23,29]). The main goal of this subsection is to obtain such a smallest vector with extra properties for our purpose. The algorithm `REDUCE` (see Algorithm 1) is an important tool of this paper.

Lemma 5. *Algorithm* 1 *is correct.*

Proof. We just look at the map:

$$\begin{array}{rccc} f: & \mathbb{R} & \longrightarrow & \mathbb{R} \\ & \alpha & \mapsto & \|\mathbf{u} + \alpha\mathbf{v}\| \end{array}$$

We aim to minimize $f(\alpha) = |u_1 + \alpha v_1| + |u_2 + \alpha v_2|$ among integers. Let's separate two cases:

– $v_j = 0$

It must be $v_i \neq 0$, because we are supposing $\mathbf{v} \neq 0$. Then, $f(\alpha) = |u_j| + |u_i + \alpha v_i|$ takes its minimum at $\dfrac{-u_i}{v_i}$.

– $v_1 v_2 \neq 0$

We then have two singular points:

$$\frac{-u_1}{v_1}, \frac{-u_2}{v_2},$$
$$f\left(\frac{-u_k}{v_k}\right) = \frac{|u_1 v_2 - u_2 v_1|}{|v_k|}.$$

So, f also reaches its minimum at $\dfrac{-u_i}{v_i}$; although the map takes the same value in the gap limited by both singular points whenever $|v_1| = |v_2|$.

Once this is seen, it is clear that the integer that minimizes f must be in:

$$\left\{\left\lfloor \frac{-u_i}{v_i} \right\rfloor, \left\lceil \frac{-u_i}{v_i} \right\rceil\right\}. \quad \blacksquare$$

The next result collects several properties of this concept for later use. We give a proof in Appendix A.

Lemma 6. *Let $\mathbf{u}, \mathbf{v} \in \mathbb{R}^2$ be two vectors such that $\mathbf{v} \neq 0$. Let $i \in \{1, 2\}$, $j \in \{1, 2\} \backslash \{i\}$ as in Algorithm 1, that is, $|v_i| > |v_j| \vee (i = 1 \wedge |v_1| = |v_2|)$. We have the following properties:*

(i) $\mathbf{r} = \text{Reduce}_{\mathbf{v}}(\mathbf{u}) \Rightarrow |r_i| < |v_i|$.
(ii) $\mathbf{h} \in \mathbf{u} + \mathbb{Z}\mathbf{v} \Rightarrow \text{Reduce}_{\mathbf{v}}(\mathbf{h}) = \text{Reduce}_{\mathbf{v}}(\mathbf{u})$.
(iii) $\text{Reduce}_{\mathbf{v}}(\text{Reduce}_{\mathbf{v}}(\mathbf{u})) = \text{Reduce}_{\mathbf{v}}(\mathbf{u})$.
(iv) $\text{Reduce}_{\mathbf{v}}(\mathbf{u}) = \text{Reduce}_{-\mathbf{v}}(\mathbf{u})$.

The properties (ii)–(iv) in the above lemma show that $\text{Reduce}_{\mathbf{v}}(\mathbf{u})$ is invariant in the set $\mathbf{u} + \mathbb{Z}\mathbf{v}$.

In order to gauge the norm reduction performed by `REDUCE` procedure, the next result provides the following bound:

Proposition 7. *Let $\mathbf{u}, \mathbf{v} \in \mathbb{R}^2$ be two vectors such that $\mathbf{v} \neq \mathbf{0}$. Let $i \in \{1, 2\}$, $j \in \{1, 2\} \backslash \{i\}$ as in Algorithm 1; with $|v_i| = \beta |v_j|$ for some $1 \leq \beta \in \mathbb{R}$. If $|v_i| \leq |u_i|$ and $|u_j| \neq 0$, then:*

$$\|\text{Reduce}_{\mathbf{v}}(\mathbf{u})\| \leq \frac{\alpha\left(1 + \frac{1}{\beta}\right) + 2}{2\alpha + 2} \|\mathbf{u}\|,$$

where $|u_i| = \alpha |u_j|$, for some $\alpha \in \mathbb{R}$.

A proof is given in Appendix B. We note that:

$$\frac{\alpha\left(1 + \frac{1}{\beta}\right) + 2}{2 + 2\alpha} \leq 1,$$

because $\beta \geq 1$. And if $\beta > 1$, then the inequality is strict. The requirement $|v_i| = \beta |v_j|$ excludes from this result the reduction by a vector with one vanished component. We are also excluding the possibility $|u_j| = 0$. In the next result, we are dealing with these cases.

In a similar way, we can obtain the following result:

Proposition 8. *Let $\mathbf{u}, \mathbf{v} \in \mathbb{R}^2$ be two vectors such that $\mathbf{v} \neq 0$. Let $i \in \{1, 2\}$, $j \in \{1, 2\} \backslash \{i\}$ as Algorithm 1 with $|v_i| \leq |u_i|$. We have:*

(i) *If $v_j = 0 \wedge u_j \neq 0$, then:*

$$\|\text{Reduce}_{\mathbf{v}}(\mathbf{u})\| \leq \frac{2 + \alpha}{2 + 2\alpha} \|\mathbf{u}\|, \text{ where } \alpha = |u_i| / |u_j|.$$

(ii) *If* $u_j = 0 \wedge v_j \neq 0$, *then:*

$$\|\text{Reduce}_{\mathbf{v}}(\mathbf{u})\| \leq \frac{1+\beta}{2\beta}\|\mathbf{u}\|, \text{ where } \beta = |v_i|/|v_j|.$$

(iii) *If* $|v_j| = |u_j| = 0$, *then:*

$$\|\text{Reduce}_{\mathbf{v}}(\mathbf{u})\| \leq \frac{1}{2}\|\mathbf{u}\|.$$

3.2. The method's core

We are describing precisely our goal. We start with three vectors in $\mathbb{R}^2$, two of them linearly independent:

$$\mathbf{w} + \mathbb{Z}\langle \mathbf{u}, \mathbf{v}\rangle, \ \text{rank}(\mathbf{u}, \mathbf{v}) = 2.$$

We are going to find the shortest element in that set with respect to ℓ_1 norm. The method consists of recursively applying the REDUCE algorithm to the "translation" vector **w** by using some vectors in $\mathbb{Z}\langle \mathbf{u}, \mathbf{v}\rangle$. Our purpose is to guarantee that each step reduces the vector norm by a constant factor, until we reach an ending condition.

To perform this goal, we select a particular basis of the lattice $\mathbb{Z}\langle \mathbf{u}, \mathbf{v}\rangle$:

Definition 9. A lattice basis $\{\mathbf{u}, \mathbf{v}\}$ is called extra-reduced when:

$$\text{Reduce}_{\mathbf{v}}(\mathbf{u}) = \mathbf{u} \ \wedge \text{Reduce}_{\mathbf{u}}(\mathbf{v}) = \mathbf{v}.$$

In order to study some properties of this kind of lattice basis, we introduce a new concept:

Definition 10. We classify vectors $\mathbf{u} = (u_1, u_2) \in \mathbb{R}^2$ into two types:

- **horizontal**, if $|u_1| \geq |u_2|$.
- **vertical**, if $|u_1| < |u_2|$.

Lemma 11. *Let* $\{\mathbf{u}, \mathbf{v}\}$ *be an extra-reduced basis. Then,*

(i) *The basis is also Gauss-reduced.*
(ii) *One of the basis vectors is vertical, and the other one is horizontal.*

Proof. The proof of the first claim is straightforward. For the second, we first suppose both basis vectors are vertical, that is,

$$\mathbf{u} = (u_1, u_2), \ |u_1| < |u_2| \quad \text{and} \quad \mathbf{v} = (v_1, v_2), \ |v_1| < |v_2|.$$

We look at the vectors:

$$\mathbf{u} + \mathbf{v} = (u_1 + v_1, u_2 + v_2), \quad \mathbf{u} - \mathbf{v} = (u_1 - v_1, u_2 - v_2).$$

We select the one that minimizes the second component modulus, whose norm is:

$$||u_1| \pm |v_1|| + ||u_2| - |v_2|| \leq \begin{cases} |u_1| + |v_1| + |u_2| - |v_2| < \|\mathbf{u}\| \\ \vee \\ |u_1| + |v_1| + |v_2| - |u_2| < \|\mathbf{v}\|. \end{cases}$$

It is clear, in both options, that the basis is Gauss-reduced.

We suppose now that both vectors are horizontal, that is,

$$\mathbf{u} = (u_1, u_2), \ |u_1| \geq |u_2|$$
$$\mathbf{v} = (v_1, v_2), \ |v_1| \geq |v_2|.$$

Using the first claim of Lemma 6, we have that:

$$\mathbf{u} = \text{Reduce}_{\mathbf{v}}(\mathbf{u}) \Rightarrow |u_1| < |v_1|$$
$$\mathbf{v} = \text{Reduce}_{\mathbf{u}}(\mathbf{v}) \Rightarrow |v_1| < |u_1|. \quad \blacksquare$$

```
Input: $\mathbf{u}, \mathbf{v} \in \mathbb{R}^2$, Gauss-reduced basis of a lattice.
Output: $\mathbf{U}, \mathbf{V} \in \mathbb{R}^2$, extra-reduced basis of the same lattice.
1  Set the vectors $\mathbf{U}$, $\mathbf{V}$ so that $\{\mathbf{U}, \mathbf{V}\} = \{\mathbf{u}, \mathbf{v}\}$ and $\|\mathbf{U}\| \le \|\mathbf{V}\|$;
2  if $\|\mathbf{U}\| < \|\mathbf{V}\|$ then
3      $\mathbf{V} := \mathrm{Reduce}_{\mathbf{U}}(\mathbf{V})$
4  else
5      if $\mathbf{U}$ and $\mathbf{V}$ are horizontal then
6          if $|U_1| \ne |U_2|$ then
7              Swap $\mathbf{U}$ and $\mathbf{V}$;
8          end
9          $\mathbf{V} := \mathrm{Reduce}_{\mathbf{U}}(\mathbf{V})$;
10     else
11         if $\mathbf{U}$ is vertical then
12             Swap $\mathbf{U}$ and $\mathbf{V}$;
13         end
14     end
15     if $U_2 < 0$ then
16         $\mathbf{U} := -\mathbf{U}$;
17     end
18     if $V_1 < 0$ then
19         $\mathbf{V} := -\mathbf{V}$;
20     end
21 end
```

Algorithm 2. Extra-reduced basis algorithm.

Note 12. As a consequence of the above proof, we can classify the Gauss-reduced basis as well, as they consist on:

– A vertical vector and an horizontal one.
– Two horizontal vectors.

In the second option, one of them (at least) must be diagonal ($|u_1| = |u_2|$), because in other case, the same argument as in the first proof's part would reach a contradiction.

Kaib and Schonrr showed in [23] how to get a reduced basis (referred to any norm, in particular ℓ_1) of lattice in two dimensions. Then, it is easy to reach an extra-reduced basis, by just applying `REDUCE` procedure, see Algorithm 2.

Lemma 13. *Algorithm 2 is correct.*

Proof. – If both input vectors' norms are different, the algorithm ends at step 4. If $\mathbf{U}$ and $\mathbf{V}$ are the input vectors, ordered such that $\|\mathbf{U}\| < \|\mathbf{V}\|$, we output:

$$\{\mathbf{U}, \mathrm{Reduce}_{\mathbf{U}}(\mathbf{V})\}.$$

This output is a basis of the input lattice, because the change matrix takes the form:

$$\begin{pmatrix} 1 & \lambda \\ 0 & 1 \end{pmatrix},$$

for an integer λ. Furthermore, it is an extra-reduced basis: firstly, by Lemma 6(iii):

$$\mathrm{Reduce}_{\mathbf{U}}(\mathrm{Reduce}_{\mathbf{U}}(\mathbf{V})) = \mathrm{Reduce}_{\mathbf{U}}(\mathbf{V}).$$

On the other hand, $\mathrm{Reduce}_{\mathrm{Reduce}_{\mathbf{U}}(\mathbf{V})}(\mathbf{U})$ is a vector in the lattice, and since $\{\mathbf{U}, \mathbf{V}\}$ is an ordered Gauss-reduced basis, it must be:

$$\|\mathbf{U}\| \le \|\mathrm{Reduce}_{\mathrm{Reduce}_{\mathbf{U}}(\mathbf{V})}(\mathbf{U})\|.$$

The converse comes from the meaning of the `REDUCE` process, so

$$\|\mathbf{U}\| = \|\mathrm{Reduce}_{\mathrm{Reduce}_{\mathbf{U}}(\mathbf{V})}(\mathbf{U})\|.$$

Now, since $\|\mathbf{V}\|$ is the second Minkowski's minimum in the lattice, and its norm is greater than $\|\mathbf{U}\|$, it must be $\mathrm{Reduce}_{\mathrm{Reduce}_{\mathbf{U}}(\mathbf{V})}(\mathbf{U}) \in \mathbb{Z} < \mathbf{U} >$, and so,

$$\mathbf{U} = \mathrm{Reduce}_{\mathrm{Reduce}_{\mathbf{U}}(\mathbf{V})}(\mathbf{U}).$$

```
Input: $\mathbf{w}, \mathbf{u}, \mathbf{v} \in \mathbb{R}^2$; $\{\mathbf{u}, \mathbf{v}\}$, extra − reduced basis ($\mathbf{u}$, hor., $\mathbf{v}$, ver.).
Output: $\mathbf{W} \in \mathbf{w} + \mathbb{Z} < \mathbf{u}, \mathbf{v} >$, $|W_1| < |u_1|$, $|W_2| < |v_2|$.
1 $\mathbf{W} := \mathbf{w}$;
2 while $|W_1| \geq |u_1| \ \vee \ |W_2| \geq |v_2|$ do
3     if $|W_1| \geq |u_1|$ then
4         $\mathbf{W} := \mathrm{Reduce}_{\mathbf{u}}(\mathbf{W})$;
5     else
6         $\mathbf{W} := \mathrm{Reduce}_{\mathbf{v}}(\mathbf{W})$;
7     end
8 end
```

Algorithm 3. Iterative reduce.

– We now analyze the case $\|\mathbf{u}\| = \|\mathbf{v}\|$.

According to Note 12, after Step 8, we have $|U_1| = |U_2|$. Now, if we perform Step 9, we reach a basis where $\mathbf{U}$ is horizontal, and $\mathbf{V}$ is vertical:

$$|V_1| + |V_2| = \|\mathbf{V}\| = \|\mathbf{U}\| = 2|U_1| > 2|V_1|.$$

The last inequality follows from Lemma 6(i).

Then, $|V_2| > |V_1|$.

Even if we have skipped Steps 6–9, after 14, $\mathbf{U}$ is horizontal, and $\mathbf{V}$ is vertical.

We have to see now that after Step 3.4 we have an extra-reduced basis. The situation is as follows:

$$0 \leq U_2 \leq |U_1|.$$
$$0 \leq V_1 < |V_2|.$$

We have that $|U_2| < |V_2|$ and that $|V_1| < |U_1|$, because $\|\mathbf{U}\| = \|\mathbf{V}\|$. Then,

$$\left\lfloor \frac{U_2}{V_2} \right\rfloor, \left\lfloor \frac{V_1}{U_1} \right\rfloor \in \{-1, 0\},$$

and then the basis $\{\mathbf{U}, \mathbf{V}\}$ is extra-reduced. ■

We will then use this extra-reduced basis to perform iterative reductions of the vector $\mathbf{w}$, as in explained in Algorithm 3.

From Lemma 6, Propositions 7 and 8 we can obtain the following:

Lemma 14. *Algorithm 3 is correct and the number of performed loops is* $O(\log \|\mathbf{w}\|)$.

Proof. By Lemma 6.1, in consecutive loops different substeps must be performed. So, for every two loops, one reduction by $\mathbf{v}$ is made. But by Propositions 7 and 8, there exists a constant $\lambda < 1$ such that every step of this kind produces:

$$\|\mathbf{W}^{new}\| \leq \lambda \|\mathbf{W}^{old}\|.$$

That constant λ is the minimum reduction factor provided by Proposition 7 or 8. Note that the parameter α cannot be arbitrary small: suppose that $\mathbf{W}^n$ is a vector obtained at step n, and $|W_1| < |u_1|$. Then, the parameter α corresponding to vector $\mathbf{W}$ at step $n + 2k$ satisfies:

$$\alpha \geq \frac{|v_i|}{\|\mathbf{W}^n\| - |v_i|}.$$

So, as a lattice is a discrete subset, the lemma follows. ■

3.3. *The whole process*

We have reached a vector in $\mathbf{w} + \mathbb{Z}\langle \mathbf{u}, \mathbf{v} \rangle$ with some properties. We need to conclude our job by getting the shortest vector among all. We will use the following technical result:

```
Input: $\mathbf{u}, \mathbf{v}$, extra-reduced basis ($\mathbf{u}$, hor. $\mathbf{v}$, ver.), $\mathbf{w}$, with
       $|w_1| < |u_1|,\ |w_2| < |v_2|$
Output: $\mathbf{W}$, shortest vector in $\mathbf{w} + Z < \mathbf{u}, \mathbf{v} >$.
1 $\mathbf{U} := \mathbf{u},\ \mathbf{V} := \mathbf{v}$;
2 if $|U_1| < |V_2|$ then
3   | Swap $\mathbf{U}$ and $\mathbf{V}$;
4 end
5 for $\alpha = [-8, \dots, 8]$ do
6   | $\mathbf{W}_\alpha := \mathrm{Reduce}_{\mathbf{V}}(\mathbf{w} + \alpha\mathbf{U})$;
7 end
8 Return a vector with minimum norm in $\{\mathbf{W}_\alpha \mid |\alpha| \le 8\}$;
```

Algorithm 4. Final reduce.

Lemma 15. *Let $\{\mathbf{u}_1, \mathbf{u}_2\}$ be a reduced basis (respect to any norm) of a lattice in $\mathbb{R}^m$. Let $\mathbf{w} = \alpha_1\mathbf{u}_1 + \alpha_2\mathbf{u}_2$ be a lattice vector ($\alpha_1, \alpha_2 \in \mathbb{Z}$). Then, we have:*

$$\|\alpha_1\mathbf{u}_1\|, \|\alpha_2\mathbf{u}_2\| \le 2\|\mathbf{w}\|.$$

Proof. – If $\mathbf{w} = 0$, the result is clear.

– If $\alpha_1\alpha_2 = 0$, we have:

$$\|\mathbf{w}\| = \begin{cases} \|\alpha_1\mathbf{u}_1\| \\ \vee \\ \|\alpha_2\mathbf{v}_2\|. \end{cases}$$

– If $\alpha_1\alpha_2 \ne 0$, let us write $\sigma \in \Sigma_2$, such that

$$|\alpha_{\sigma(1)}| \le |\alpha_{\sigma(2)}|.$$

Then, writing $\varepsilon := \operatorname{sgn}\left(\frac{\alpha_{\sigma(1)}}{\alpha_{\sigma(2)}}\right) \in \{-1, 1\}$,

$$\|\mathbf{u}_{\sigma(1)}\| \le \|\mathbf{u}_{\sigma(1)} + \varepsilon\mathbf{u}_{\sigma(2)}\| \Rightarrow \|\mathbf{u}_{\sigma(1)}\| \le \|\mathbf{u}_{\sigma(1)} + \frac{\alpha_{\sigma(2)}}{\alpha_{\sigma(1)}}\mathbf{u}_{\sigma(2)}\| \Rightarrow \|\alpha_{\sigma(1)}\mathbf{u}_{\sigma(1)}\| \le \|\mathbf{v}\|.$$

Then,

$$\|\alpha_{\sigma(2)}\mathbf{u}_{\sigma(2)}\| \le \|\mathbf{v}\| + \|\alpha_{\sigma(1)}\mathbf{u}_{\sigma(1)}\|. \quad \blacksquare$$

Let us now suppose that we have reached a description $\mathbf{w} + \mathbb{Z} < \mathbf{u}, \mathbf{v} >$ of the original set, where $\{\mathbf{u}, \mathbf{v}\}$ is an extra-reduced basis ($\mathbf{u}$ is horizontal and $\mathbf{v}$ is vertical), and with $|w_1| < |u_1|,\ |w_2| < |v_2|$. Let $\mathbf{W}$ be a closest vector to $\mathbf{w}$ with respect to ℓ_1 norm.

Then, $\mathbf{d} := \mathbf{W} - \mathbf{w} \in \mathbb{Z}\langle\mathbf{u}, \mathbf{v}\rangle$ and

$$\|\mathbf{d}\| \le 2\|\mathbf{w}\| = 2(|w_1| + |w_2|) < 2(|u_1| + |v_2|).$$

We try to bound the coefficients of $\mathbf{d}$ as a lattice member:

$$\mathbf{d} = \alpha\mathbf{u} + \beta\mathbf{v}.$$

We distinguish two cases and applying previous Lemma 15

– $|u_1| \ge |v_2|$

$$\|\alpha\mathbf{u}\| \le 2\|\mathbf{d}\| \Rightarrow |\alpha| \le \frac{4(|u_1| + |v_2|)}{|u_1| + |u_2|} \le 8$$

– $|u_1| < |v_2|$

$$\|\beta\mathbf{v}\| \le 2\|\mathbf{d}\| \Rightarrow |\beta| \le \frac{4(|u_1| + |v_2|)}{|v_1| + |v_2|} \le 8.$$

So, we can now give a method to obtain a shortest vector.

To sum up, jointing Algorithms 2–4, we reach our goal.

```
Input: $\mathbf{u}, \mathbf{v} \in \mathbb{Z}^2,\ \mathbf{v} \neq \mathbf{0}$.
Output: $\mathrm{Reduce}_{\mathbf{v}}(\mathbf{u}) \in \mathbf{u}+\mathbb{Z}\mathbf{v} \mid \|\mathrm{Reduce}_{\mathbf{v}}(\mathbf{u})\| = \min\{\|\mathbf{u}+\alpha\mathbf{v}\| \mid \alpha \in \mathbb{Z}\}$.
1 Find $i \in \{1,2\},\ j \in \{1,2\}\setminus\{i\}$ such that $|v_i| > |v_j|$. If $|v_1| = |v_2|$, select
  $i := 1$;
2 Return the vector with minimum norm between:

      $\mathbf{u} - \mathrm{quo}(u_i, v_i)\mathbf{v},\ \ \mathbf{u} - (\mathrm{quo}(u_i, v_i) + \mathrm{sgn}\,(v_i))\mathbf{v}$.

  If both share the same norm, return $\mathbf{u} - \mathrm{quo}(u_i, v_i)\mathbf{v}$.
```

Algorithm 5. Integer reduce.

3.4. Complexity

When studying lattices from a complexity point of view, it is customary to assume that the basis vectors (and therefore any lattice vector) have all rational coordinates. It is easy to see that rational lattices can be converted to integer lattices (i.e., sublattices of $\mathbb{Z}^m$) by multiplying all coordinates by an appropriate integer scaling factor.

In this section, we assume that all vectors have integer coordinates, that belong to $\mathbb{Z}^2$.

Notation 16. *If a, b are two integers, such that* $b \neq 0$, *we denote by* $\mathrm{quo}(a, b),\ \mathrm{rem}(a, b)$ *the unique integers verifying:*

$$a = b \cdot \mathrm{quo}(a, b) + \mathrm{rem}(a, b), \quad 0 \leq \mathrm{rem}(a, b) < |b|,$$

i.e., $\mathrm{quo}(a, b),\ \mathrm{rem}(a, b)$ *are the quotient and the remainder of the Euclidean division of a by b.*

In the case of lattices with integer coefficients, Algorithm 1 can be substituted with Algorithm 5.

It is straightforward t6 check that both Algorithm 1 and Algorithm 5 have the same output for integer vectors.

Given three vectors $\{\mathbf{u}, \mathbf{v}, \mathbf{w}\}$, such that $\mathbf{u}, \mathbf{v}$ are linearly independent. Let $M = \max(\|\mathbf{u}\|, \|\mathbf{v}\|, \|\mathbf{w}\|)$. We claim that our algorithm for solving the closest vector problem costs $O(\log^2 M \log\log M \log\log\log M)$ bit operations:

- Clearly, $\mathrm{Reduce}_{\mathbf{v}}(\mathbf{u}) \in \mathbf{u} + \mathbb{Z}\mathbf{v}$ is bounded by computing four Euclidean divisions on an integer number of size $O(\log M)$.
- The algorithm in [23] computes a Gauss-reduced basis from a base $\{\mathbf{u}, \mathbf{v}\}$ of a lattice $\mathcal{L} \subset \mathbb{Z}^2$ in $O(\log M)$ arithmetic operations on integer numbers of size $O(\log M)$.
- Now, finding an extra-reduced basis from a Gauss-reduced one using Algorithm 2 requires one to compute four Euclidean division on integer of size $O(\log M)$.
- Algorithm 3 requires $O(\log M)$ arithmetic (see Lemma 14) steps and any step consists of two Euclidean divisions on an integer of size $O(\log M)$.
- Finally, Algorithm 4 computes 16 `REDUCE` procedures.

To finish this analysis, we use the famous Schönhage and Strassen algorithm [32] for the multiplication of integers.

From a practical point of view, it is interesting to remark that the Algorithm 4 only requires us to perform the `REDUCE` procedure 8 times instead of 16. However, the proof of this fact is a little longer.

Given a connected undirected circulant graph $C_N(\pm j_1, \pm j_2)$, we can assume without loss of generality that $\gcd(j_1, j_2) = 1$, see [1]. A basis for the lattice $\mathcal{L}$:

$$\mathcal{L} = \{(x, y) \in \mathbb{Z}^2 : j_1 x + j_2 y \equiv 0 \bmod N\}$$

associated to $C_N(\pm j_1, \pm j_2)$ is $\mathbf{u} = (-j_2, -j_1), \mathbf{v} = (N x_0, N y_0)$, where $1 = j_1 x_0 + j_2 y_0$. Given a vertex $j \in \mathbb{Z}_N$, we can compute by the Extended Euclidean algorithm a path $\mathbf{w}$ from 0 to j. All of this on $O(\log N \log\log N \log\log\log N)$, see for instance [14].

Corollary 17. *Given an undirected circulant graph* $C_N(\pm j_1, \pm j_2)$ *and vertex* $j \in \mathbb{Z}_N$, *we can decide if there exists a shortest path from vertex* 0 *to vertex* j *and, in the affirmative case, we can compute one on* $O(\log^2 N \log\log N \log\log\log N)$ *bit operations.*

```
Input: $\mathbf{u}, \mathbf{v} \in \mathbb{R}^2,\ \mathbf{v} \neq \mathbf{0}$.
Output: $\mathrm{PRed}_{\mathbf{v}}(\mathbf{u})$, if it exists, and $\emptyset$ otherwise.
Select $i \in \{1,2\},\ j \in \{1,2\}\backslash\{i\}$ such that $|v_i| > |v_j|$. If $|v_1| = |v_2|$, then $i = 1$;
Set $\varepsilon = \begin{cases} -1, & \text{if } |v_1| \geq |v_2| \\ 1, & \text{if } |v_1| < |v_2| \end{cases}$;
Compute $\Delta := \varepsilon(u_1 v_2 - u_2 v_1)$;
if $v_j = 0$ then
    $B := \frac{\Delta}{v_i}$;
    if $B < 0$ then
        Output $\emptyset$;
    else
        Output $\mathbf{u} - \mathrm{sgn}(v_i) \left\lfloor \frac{u_i}{|v_i|} \right\rfloor \mathbf{v}$;
    end
else
    $A := \frac{-\Delta}{v_j},\ B := \frac{\Delta}{v_i}$;
    if $(A < 0 \wedge B < 0)$ then
        Output $\emptyset$;
    end
    if $(A < 0 \wedge B \geq 0)$ then
        Output $\mathbf{u} - \mathrm{sgn}(v_i) \left\lfloor \frac{u_i}{|v_i|} \right\rfloor \mathbf{v}$;
    end
    if $(A \geq 0 \wedge B < 0)$ then
        Output $\mathbf{u} - \mathrm{sgn}(v_j) \left\lfloor \frac{u_j}{|v_j|} \right\rfloor \mathbf{v}$;
    end
    if $(A \geq 0 \wedge B \geq 0)$ then
        Set $\mathbf{w} := \mathbf{u} - \mathrm{sgn}(v_i) \left\lfloor \frac{u_i}{|v_i|} \right\rfloor \mathbf{v}$;
        if $\mathbf{w} \in \mathbb{R}^2_{\geq 0}$ then
            Output $\mathbf{w}$;
        else
            Output $\emptyset$;
        end
    end
end
```

Algorithm 6. Postive reduction.

4. Double loop networks

Our method can be easily extended to *directed circulant graphs* $DC_N(j_1, j_2)$.

First, we describe the tool of *positive reduction of a vector*. Given two vectors $\mathbf{u} = (u_1, u_2)$ and $\mathbf{v} = (v_1, v_2) \neq \mathbf{0}$, we are looking for $\alpha \in \mathbb{Z}$ such that the value $\|\mathbf{u} + \alpha\mathbf{v}\|$ is minimal and has both components positive. In general, such a vector does not exist, for instance, take $\mathbf{u} = (3, -5)$ and $\mathbf{v} = (12, 0)$. If it exists, we denote this vector by $\mathrm{PRed}_{\mathbf{v}}(\mathbf{u}) \in \mathbf{u} + \mathbb{Z}\mathbf{v}$, that is,

$$\|\mathrm{PRed}_{\mathbf{v}}(\mathbf{u})\| = \min\{\|\mathbf{u} + \alpha\mathbf{v}\| : \ \alpha \in \mathbb{Z},\ \mathbf{u} + \alpha\mathbf{v} \in \mathbb{R}^2_{\geq 0}\},$$

where $\mathbb{R}^2_{\geq 0}$ is the set of real points (x_1, x_2) with $x_i \geq 0, i = 1, 2$.

It is easy to check that the Algorithm 6, illustrated in Fig. 2, is correct. On the other hand, if we are dealing with integer vectors, we can formulate the main steps, similar to those in Algorithm 5, using the Euclidean quotient (see Algorithm 7).

Therefore, a bound for bit complexity on computing $\mathrm{PRed}_{\mathbf{v}}(\mathbf{u})$ is $O(\log^2 M \log\log M \log\log\log M)$, where $M = \max(\|\mathbf{u}\|, \|\mathbf{v}\|)$.

Once this tool is fixed, let us describe the method to reach a shortest path in a directed circulant graph. First, we act as in the previous section to compute an extra reduced basis of the associated lattice $\{\mathbf{u}, \mathbf{v}\}$, and a shortest path for the corresponding undirected circulant graph $\mathbf{w}$.

We can state that there always exists one path in the directed circulant graph with bounded length.

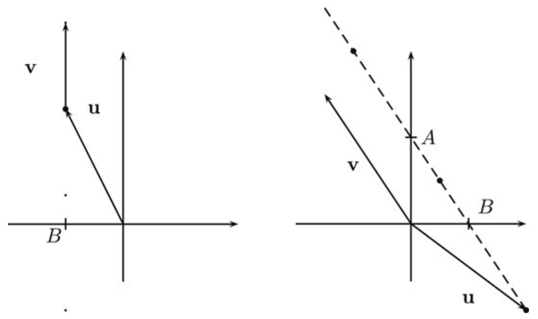

Fig. 2. Algorithm 6.

```
Input: $\mathbf{u}, \mathbf{v} \in \mathbb{Z}^2,\ \mathbf{v} \neq \mathbf{0}$.
Output: $\mathrm{PRed}_{\mathbf{v}}(\mathbf{u})$, if it exists, and $\emptyset$ otherwise.
Select $i \in \{1,2\},\ j \in \{1,2\}\backslash\{i\}$ such that $|v_i| > |v_j|$. If $|v_1| = |v_2|$, then $i = 1$;
Set $\varepsilon = \begin{cases} -1, & \text{if } |v_1| \geq |v_2| \\ 1, & \text{if } |v_1| < |v_2| \end{cases}$;
Compute $\Delta := \varepsilon \mathrm{sgn}\,(u_1 v_2 - u_2 v_1)$;
if $v_j = 0$ then
    $B := \Delta \mathrm{sgn}\,(v_i)$;
    if $B = -1$ then
        Output $\emptyset$;
    else
        Output $\mathbf{u} - \mathrm{sgn}\,(v_i)\,\mathrm{quo}(u_i, |v_i|)\mathbf{v}$;
    end
else
    $A := -\Delta \mathrm{sgn}\,(v_j)\,,\ B := \Delta \mathrm{sgn}\,(v_i)$;
    if $(A = -1 \wedge B = -1)$ then
        Output $\emptyset$;
    end
    if $(A = -1 \wedge B \geq 0)$ then
        Output $\mathbf{u} - \mathrm{sgn}\,(v_i)\,\mathrm{quo}(u_i, |v_i|)\mathbf{v}$;
    end
    if $(A \geq 0 \wedge B = -1)$ then
        Output $\mathbf{u} - \mathrm{sgn}\,(v_j)\,\mathrm{quo}(u_j, |v_j|)\mathbf{v}$;
    end
    if $(A \geq 0 \wedge B \geq 0)$ then
        Set $\mathbf{w} := \mathbf{u} - \mathrm{sgn}\,(v_i)\,\mathrm{quo}(u_i, |v_i|)\mathbf{v}$;
        if $\mathbf{w} \in \mathbb{N}^2_{\geq 0}$ then
            Output $\mathbf{w}$;
        else
            Output $\emptyset$;
        end
    end
end
```

Algorithm 7. Positive reduction with integer components.

Lemma 18. *Let* $\mathbf{w}, \mathbf{u}, \mathbf{v} \in \mathbb{Z}^2$*, such that* $\{\mathbf{u}, \mathbf{v}\}$ *is an extra reduced basis for the lattice they generate. Then,*

$$\exists \mathbf{d} \in (\mathbf{w} + \mathbb{Z}\langle \mathbf{u}, \mathbf{v}\rangle) \cap \mathbb{N}^2,\ \ \|\mathbf{d}\| \leq 6 \max\{\|\mathbf{u}\|, \|\mathbf{v}\|\}.$$

Proof. Let $M = \max\{\|\mathbf{u}\|, \|\mathbf{v}\|\}$. We consider the translated lattice

$$\mathbf{w} - (2M, 2M) + \mathbb{Z}\langle \mathbf{u}, \mathbf{v}\rangle.$$

By Algorithm 3, this set contains an element $\mathbf{z}$, with $|z_1| < |u_1|,\ |z_2| < |v_2|$. So, $\|\mathbf{z}\| \leq 2M$. Clearly $\mathbf{z} + (2M, 2M)$ belongs to the set $(\mathbf{w} + \mathbb{Z}\langle \mathbf{u}, \mathbf{v}\rangle) \cap \mathbb{N}^2$, and its norm is bounded by $6M$. ■

```
Input: $\mathbf{w} \in \mathbb{Z}^2$, $\{\mathbf{u},\mathbf{v}\}$, extra reduced basis.
Output: $\mathbf{d}$, shortest element in $(\mathbf{w} + \mathbb{Z} < \mathbf{u},\mathbf{v} >) \cap \mathbb{N}^2$.
Find a shortest element $\mathbf{z}$ in $\mathbf{w} + \mathbb{Z} < \mathbf{u},\mathbf{v} >$.;
if $\|\mathbf{u}\| \geq \|\mathbf{v}\|$ then
    for $\alpha = -16, \ldots, 16$ do
        $\mathbf{d}_\alpha := \mathrm{PRed}_{\mathbf{v}}(\mathbf{z} + \alpha\mathbf{u})$;
    end
else
    for $\alpha = -16, \ldots, 16$ do
        $\mathbf{d}_\alpha := \mathrm{PRed}_{\mathbf{u}}(\mathbf{z} + \alpha\mathbf{v})$
    end
end
Output $\min\{\mathbf{d}_\alpha \ / \ |\alpha| \leq 16\}$;
```

Algorithm 8. Positive shortest path.

Finally, we follow a similar argument as in Section 3 to reach the shortest path for the directed graph.

Lemma 19. *Let* $\mathbf{w}, \mathbf{u}, \mathbf{v} \in \mathbb{Z}^2$*, such that* $\{\mathbf{u}, \mathbf{v}\}$ *is an extra reduced basis for the lattice they generate. Let* $\mathbf{W}$ *be a shortest element in a translated lattice* $\mathbf{w} + \mathbb{Z}\langle\mathbf{u}, \mathbf{v}\rangle$ *and let* $\mathbf{d}$ *be a shortest element in* $(\mathbf{w} + \mathbb{Z}\langle\mathbf{u}, \mathbf{v}\rangle) \cap \mathbb{N}^2$*. We have:* $\mathbf{d} - \mathbf{W} = \alpha\mathbf{u} + \beta\mathbf{v}$*, verifying*

$$|\alpha| \leq 16 \text{ if } \|\mathbf{u}\| \geq \|\mathbf{v}\|, \quad |\beta| \leq 16 \text{ if } \|\mathbf{v}\| \geq \|\mathbf{u}\|.$$

Proof. Let $M = \max\{\|\mathbf{u}\|, \|\mathbf{v}\|\}$, by Lemmas 15 and 18 and since $\{\mathbf{u}, \mathbf{v}\}$ is an extra reduced basis, we have:

$$\|\alpha\mathbf{u}\|, \|\beta\mathbf{v}\| \leq 2\|\mathbf{d} - \mathbf{w}\| \leq 2\|\mathbf{d}\| + 2\|\mathbf{W}\| \leq 12M + 4M \leq 16M.$$

So, the lemma follows. ■

Using the above lemma it is straightforward to prove that the next Algorithm 8 computes a shortest vector with positive components.

As an immediate consequence is that we can compute a shortest path in a directed circulant graph of degree two in $O(\log^2 N \log\log N \log\log\log N)$ bit operations.

5. Weighted circulant graphs

In this section, we consider weighted circulant graphs with two jumps and weights.

Theorem 20. *Given a undirected circulant graph* $C_N(\pm j_1, \pm j_2)$ *with weights* $w = (w_1, w_2)$ *we can find a shortest path in* $O(\log^2 N \log\log N \log\log\log N)$ *bit operations.*

Proof. The distance of a path $\mathbf{c} = (c_1, c_2)$ in the weighted circulant graph is

$$\|\mathbf{c}\|_w = w_1|c_1| + w_2|c_2|.$$

Let $\mathbf{c} \in \mathbb{Z}^2$; then $\|\mathbf{c}\|_w = \|\Phi(\mathbf{c})\|_{\ell_1}$, where Φ is the injective group homomorphism

$$\Phi : \mathbb{Z}^2 \to \mathbb{Z}^2, \quad \Phi((x, y)) = (w_1 x, w_2 y).$$

Let $j \in \mathbb{Z}_N$ be a vertex of the graph, and let $\mathbf{u}, \mathbf{v}$ be an extra-reduced basis of the circulant graph $C_N(j_1, j_2)$. By Corollary 17, we compute in cubic polynomial time a shortest path $\mathbf{c}$ from vertex 0 to vertex j. Let $\mathbf{d}$ a solution to CVP for the lattice generated by $\langle\Phi(\mathbf{u}), \Phi(\mathbf{v})\rangle$, shifted by $\Phi(\mathbf{c})$; then $\Phi^{-1}(\mathbf{d})$ is a shortest path in the weighted undirected circulant graph. ■

The algorithm in the above theorem can be adapted in a natural way to weighted directed circulant graphs.

6. Conclusions

We provide algorithms specifically tailored for finding a shortest path between to vertices of any weighted undirected and directed circulant graph with two jumps. Our algorithms are very efficient: we can find a shortest path instantaneously in a circulant graph with more 2^{300} vertices using a C++ implementation in a PC computer with two processors of 1 GHz and 1 Gb of memory RAM.

We have introduced the concept of an extra-reduced basis. We think that this new concept may shed light on problems in this kind of lattice, and can be extended to higher dimensions in an appropriate way.

We also leave unresolved many interesting questions. Unfortunately, we do not know how to extend those results to more jumps.

Concerning applications, we regard the future interrelation of our algorithm to circulant graphs in order to consider different problems in distributed double loop networks, like fault tolerance, and explicitly describe the class of non-isomorphic optimal networks.

Acknowledgments

This research was partially supported by the Research Project MTM2004-07086 of the Spain Ministry of Science.

An extended abstract of a part of this work was presented at the Annual International Computing and Combinatorics Conference COCOON-05, Kunming, China 2005, [17].

Appendix A

Proof of Lemma 6. (i) The number r_i is

$$u_i - \left\lfloor \frac{u_i}{v_i} \right\rfloor v_i, \quad \text{or} \quad u_i - \left\lfloor \frac{u_i}{v_i} \right\rfloor v_i - v_i.$$

We divide the study into several cases:

– If $v_i > 0$:

$$\left\lfloor \frac{u_i}{v_i} \right\rfloor \leq \frac{u_i}{v_i} \Rightarrow \left\lfloor \frac{u_i}{v_i} \right\rfloor v_i \leq u_i \Rightarrow u_i - \left\lfloor \frac{u_i}{v_i} \right\rfloor v_i \geq 0,$$

$$\left\lfloor \frac{u_i}{v_i} \right\rfloor + 1 > \frac{u_i}{v_i} \Rightarrow \left\lfloor \frac{u_i}{v_i} \right\rfloor v_i + v_i > u_i \Rightarrow u_i - \left\lfloor \frac{u_i}{v_i} \right\rfloor v_i < v_i.$$

– If $v_i < 0$:

$$\left\lfloor \frac{u_i}{v_i} \right\rfloor \leq \frac{u_i}{v_i} \Rightarrow \left\lfloor \frac{u_i}{v_i} \right\rfloor v_i \geq u_i \Rightarrow u_i - \left\lfloor \frac{u_i}{v_i} \right\rfloor v_i \leq 0,$$

$$\left\lfloor \frac{u_i}{v_i} \right\rfloor + 1 > \frac{u_i}{v_i} \Rightarrow \left\lfloor \frac{u_i}{v_i} \right\rfloor v_i + v_i < u_i \Rightarrow u_i - \left\lfloor \frac{u_i}{v_i} \right\rfloor v_i > v_i.$$

For the last one:

– If $v_i > 0$:

$$0 = v_i - v_i > u_i - \left\lfloor \frac{u_i}{v_i} \right\rfloor v_i - v_i \geq 0 - v_i.$$

And when $u_i - \left\lfloor \frac{u_i}{v_i} \right\rfloor v_i - v_i = -v_i$, we have:

$$u_i = \left\lfloor \frac{u_i}{v_i} \right\rfloor v_i \Rightarrow \frac{u_i}{v_i} = \left\lfloor \frac{u_i}{v_i} \right\rfloor \Rightarrow \frac{u_i}{v_i} \in \mathbb{Z} \Rightarrow r_i = 0.$$

– If $v_i < 0$:

$$0 = v_i - v_i < u_i - \left\lfloor \frac{u_i}{v_i} \right\rfloor v_i - v_i \leq 0 - v_i.$$

And when $u_i - \left\lfloor \frac{u_i}{v_i} \right\rfloor v_i - v_i = -v_i$, we have:

$$u_i = \left\lfloor \frac{u_i}{v_i} \right\rfloor v_i \Rightarrow \frac{u_i}{v_i} = \left\lfloor \frac{u_i}{v_i} \right\rfloor \Rightarrow \frac{u_i}{v_i} \in \mathbb{Z} \Rightarrow r_i = 0.$$

(ii) It must be $\exists \alpha \in \mathbb{Z} : \mathbf{h} = \mathbf{u} + \alpha \mathbf{v}$. Then,

$$\left\lfloor \frac{h_i}{v_i} \right\rfloor = \left\lfloor \frac{u_i}{v_i} + \alpha \right\rfloor = \left\lfloor \frac{u_i}{v_i} \right\rfloor + \alpha.$$

So, the two possibilities for $\text{Reduce}_{\mathbf{v}}(\mathbf{h})$ are:

$$\mathbf{h} - \left(\left\lfloor \frac{u_i}{v_i} \right\rfloor + \alpha \right) \mathbf{v} = \mathbf{u} - \left\lfloor \frac{u_i}{v_i} \right\rfloor \mathbf{v},$$
$$\mathbf{h} - \left(\left\lfloor \frac{u_i}{v_i} \right\rfloor + \alpha + 1 \right) \mathbf{v} = \mathbf{u} - \left(\left\lfloor \frac{u_i}{v_i} \right\rfloor + 1 \right) \mathbf{v}.$$

And, when the norms are coincident, we observe:

$$\frac{h_i}{v_i} \in \mathbb{Z} \iff \frac{u_i}{v_i} \in \mathbb{Z}.$$

(iii) It is straightforward from previous item.

(iv) – When $\frac{u_i}{v_i} \in \mathbb{Z}$, it must be

$$\text{Reduce}_{\mathbf{v}}(\mathbf{u}) = \mathbf{u} - \frac{u_i}{v_i}\mathbf{v},$$
$$\text{Reduce}_{-\mathbf{v}}(\mathbf{u}) = \mathbf{u} - \frac{u_i}{-v_i}(-\mathbf{v}) = \mathbf{u} - \frac{u_i}{v_i}\mathbf{v}.$$

– In any other case, we show

$$\left\lfloor \frac{u_i}{-v_i} \right\rfloor = -\left\lceil \frac{u_i}{v_i} \right\rceil,$$

and $\left\lfloor \frac{u_i}{v_i} \right\rfloor + 1 = \left\lceil \frac{u_i}{v_i} \right\rceil$. So,

$$\text{Reduce}_{\mathbf{v}}(\mathbf{u}) \in \left\{ \mathbf{u} - \left\lfloor \frac{u_i}{v_i} \right\rfloor \mathbf{v}, \mathbf{u} - \left(\left\lfloor \frac{u_i}{v_i} \right\rfloor + 1 \right) \mathbf{v} \right\},$$
$$\text{Reduce}_{-\mathbf{v}}(\mathbf{u}) \in \left\{ \mathbf{u} + \left\lceil \frac{u_i}{v_i} \right\rceil (-\mathbf{v}), \mathbf{u} + \left(1 - \left\lceil \frac{u_i}{v_i} \right\rceil \right) \mathbf{v} \right\}.$$

And the algorithm takes the same output in both cases. ■

Appendix B

Proof of Proposition 7. Firstly, let's prove that $\exists \mathbf{h} \in \mathbf{u} + \mathbb{Z}\mathbf{v}$ with $|h_i| < \frac{|u_i|}{2}$, and $\big(h_i \neq 0 \Rightarrow \text{sgn}(h_i) = \text{sgn}(u_i)\big)$.

We just see that there are two vectors $\mathbf{r}$ and $\mathbf{s}$ in $\mathbf{u} + \mathbb{Z}\mathbf{v}$, with: $r_i = u_i - \left\lfloor \frac{u_i}{v_i} \right\rfloor v_i$, $s_i = u_i - \left\lfloor \frac{u_i}{v_i} \right\rfloor v_i - v_i$.

As $|v_i| \leq |u_i|$, it must be $|u_i| > 0$. Also, $\left| \frac{u_i}{v_i} \right| \geq 1 \Rightarrow \left\lfloor \frac{u_i}{v_i} \right\rfloor \neq 0$. We also see that $\left\lfloor \frac{u_i}{v_i} \right\rfloor = -1 \Rightarrow \frac{u_i}{v_i} = -1 \Rightarrow r_i = 0$. We suppose then $\frac{u_i}{v_i} \notin \mathbb{Z}$, $\left\lfloor \frac{u_i}{v_i} \right\rfloor \notin \{-1, 0\}$.

We divide the proof into several cases:

– If $v_i > 0$, $u_i - \left\lfloor \frac{u_i}{v_i} \right\rfloor v_i \geq 0$ and $u_i - \left\lfloor \frac{u_i}{v_i} \right\rfloor v_i - v_i < 0$.

. If $u_i > 0$, then $\left\lfloor \frac{u_i}{v_i} \right\rfloor \geq 1 \Rightarrow \left\lfloor \frac{u_i}{v_i} \right\rfloor < 2 \left\lfloor \frac{u_i}{v_i} \right\rfloor$.

$$\left\lfloor \frac{u_i}{v_i} \right\rfloor < 2 \left\lfloor \frac{u_i}{v_i} \right\rfloor \Rightarrow \frac{u_i}{v_i} < 2 \left\lfloor \frac{u_i}{v_i} \right\rfloor \Rightarrow \frac{u_i}{v_i} - \frac{u_i}{2v_i} < \left\lfloor \frac{u_i}{v_i} \right\rfloor \Rightarrow u_i - \left\lfloor \frac{u_i}{v_i} \right\rfloor v_i < \frac{u_i}{2}.$$

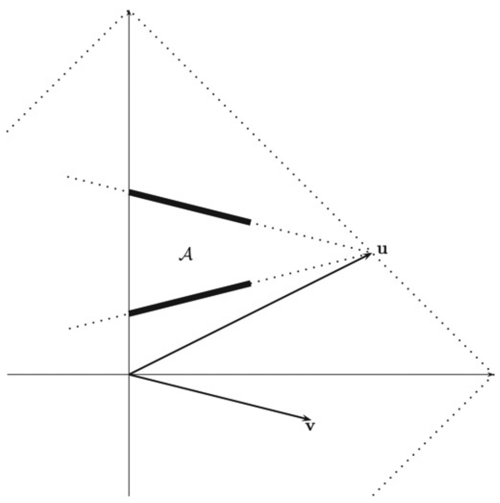

Fig. 3. The set $\mathcal{A}$.

. If $u_i < 0$, then $\left\lfloor \frac{u_i}{v_i} \right\rfloor \leq -2 \Rightarrow \left\lfloor \frac{u_i}{v_i} \right\rfloor \geq 2 + 2\left\lfloor \frac{u_i}{v_i} \right\rfloor$.

$$\left\lfloor \frac{u_i}{v_i} \right\rfloor \geq 2\left\lfloor \frac{u_i}{v_i} \right\rfloor + 2 \Rightarrow 2\left\lfloor \frac{u_i}{v_i} \right\rfloor + 2 < \frac{u_i}{v_i} \Rightarrow \frac{u_i}{2} > \left\lfloor \frac{u_i}{v_i} \right\rfloor v_i + v_i \Rightarrow u_i - \left\lfloor \frac{u_i}{v_i} \right\rfloor v_i - v_i > \frac{u_i}{2}.$$

– If $v_i < 0$, $u_i - \left\lfloor \frac{u_i}{v_i} \right\rfloor v_i - v_i > 0$ and $u_i - \left\lfloor \frac{u_i}{v_i} \right\rfloor v_i \leq 0$.

. If $u_i > 0$, then $\left\lfloor \frac{u_i}{v_i} \right\rfloor \leq -2$

$$\left\lfloor \frac{u_i}{v_i} \right\rfloor \geq 2\left\lfloor \frac{u_i}{v_i} \right\rfloor + 2 \Rightarrow 2\left\lfloor \frac{u_i}{v_i} \right\rfloor + 2 < \frac{u_i}{v_i} \Rightarrow \frac{u_i}{v_i} - \left\lfloor \frac{u_i}{v_i} \right\rfloor - 1 > \frac{u_i}{2v_i} \Rightarrow u_i - \left\lfloor \frac{u_i}{v_i} \right\rfloor v_i - v_i < \frac{u_i}{2}.$$

– If $u_i < 0$, then $\left\lfloor \frac{u_i}{v_i} \right\rfloor \geq 1$. So,

$$\frac{u_i}{v_i} < 2\left\lfloor \frac{u_i}{v_i} \right\rfloor \Rightarrow \frac{u_i}{v_i} - \left\lfloor \frac{u_i}{v_i} \right\rfloor < \frac{u_i}{2v_i} \Rightarrow u_i - \left\lfloor \frac{u_i}{v_i} \right\rfloor v_i > \frac{u_i}{2}.$$

In the rest of the proof, we use the following notation:

$$[a, b] := \begin{cases} (a, b), \text{ if } i = 1 \\ (b, a), \text{ if } i = 2. \end{cases}$$

We define the following set (see Fig. 3):

$$\left\{ \left[h_i, \frac{v_j}{v_i}(h_i - u_i) + u_j \right] : 0 \leq |h_i| < \frac{|u_i|}{2},\ u_i h_i \geq 0 \right\}$$
$$\subseteq \left\{ \left[h_i, \pm\frac{|v_j|}{|v_i|}(h_i - u_i) + u_j \right] : 0 \leq h_i \leq \frac{|u_i|}{2},\ u_i h_i \geq 0 \right\} =: \mathcal{A}.$$

As seen before, $\|\text{Reduce}_{\mathbf{v}}(\mathbf{u})\| \leq \max_{\mathbf{w} \in \mathcal{A}} \|\mathbf{w}\|$. Now, this maximum is reached at:

$$\left[\frac{u_i}{2}, u_j + \frac{\operatorname{sgn}(u_j)\,|u_i|}{2\beta} \right].$$

Then, $\|\text{Reduce}_{\mathbf{v}}(\mathbf{u})\| \leq \frac{|u_i|}{2} + |u_j| + \frac{|u_i|}{2\beta}$. And so,

$$\frac{\|\text{Reduce}_{\mathbf{v}}(\mathbf{u})\|}{\|\mathbf{u}\|} \leq \frac{|u_i|\left(\frac{1+\beta}{2\beta}\right) + |u_j|}{|u_1| + |u_2|} = \frac{\alpha|u_j|\left(\frac{1+\beta}{2\beta}\right) + |u_j|}{(1+\alpha)|u_j|} = \frac{2+\alpha\left(1+\frac{1}{\beta}\right)}{2+2\alpha}. \qquad \blacksquare$$

References

[1] A. Ádam, Research problem 2-10, Journal of Combinatorial Theory 393 (1991) 1109–1124.

[2] S.R. Blackburn, D. Gomez-Perez, J. Gutierrez, I.E. Shparlinski, Predicting nonlinear pseudorandom number generators, Mathematics of Computation 74 (2005) 1471–1494.

[3] R. Beivide, E. Herrada, J.L. Balcázar, A. Arruabarrena, Optimal distance networks of low degree for parallel computers, IEEE Transactions on Computers C-40 (10) (1991) 1109–1124.

[4] J.-C. Bermond, F. Comellas, D.F. Hsu, Distributed loop computer networks: A survey, Journal of Parallel and Distributed Computing 24 (1995) 2–10.

[5] F.T. Boesch, R. Tindell, Circulants and their connectivity, Journal of Graph Theory 8 (1984) 487–499.

[6] J.-Y. Cai, G. Havas, B. Mans, A. Nerurkar, J.-P. Seifert, I. Shparlinski, On routing in circulant graphs, in: Proc. Fifth Annual International Computing and Combinatorics Conference, Tokyo, Japan, July 26–28, 1999, in: T. Asano, H. Imai, D.T. Lee, S. Nakano, T. Tokuyama (Eds.), LNCS, vol. 1627, Springer-Verlag, 1999, pp. 360–369.

[7] N. Chalamaiah, B. Ramamurthy, Finding shortest paths in distributed loop networks, Information Processing Letters 67 (1998) 157–161.

[8] Y. Cheng, F.K. Hwang, Diameters of weighted double loop networks, Journal of Algorithms 9 (1988) 401–410.

[9] D. Cheung, F. Cucker, Solving linear programs with finite precision. I. Condition numbers and random programs, Mathematical Programming 99 (1-A) (2004) 175–196.

[10] D.Z. Du, D.F. Hsu, F.K. Hwang, Double-linked ring networks, IEEE Transactions on Computers 34 (1985) 853–877.

[11] F. Eisenbrand, Fast integer programming in fixed dimension, in: Proc. ESA-2003, in: Lect. Notes Comp. Science LNCS, vol. 2832, Springer-Verlag, 2003, pp. 196–207.

[12] P. Erdös, D.F. Hsu, Distributed loop networks with minimum transmission delay, Theoretical Computer Science 100 (1992) 223–241.

[13] M. Fiol, J.L. Yebra, I. Alegre, M. Valero, A discrete optimization problem in local networks and data alignment, IEEE Transactions on Computers C-36 (6) (1987) 702–713.

[14] J. von zur Gathen, J. Gerhard, Modern Computer Algebra, second ed., Cambridge University Press, Cambridge, 2003.

[15] M. Grötschel, L. Lovász, A. Schrijver, Geometric Algorithms and Combinatorial Optimization, Springer-Verlag, Berlin, 1993.

[16] D. Gómez, J. Gutierrez, A. Ibeas, Circulant Digraphs and Monomial Ideals, in: Computer Algebra in Scientific Computing, CASC-2005, in: Lect. Notes in Comp. Sci., vol. 3718, Springer-Verlag, Berlin, 2005, pp. 196–207.

[17] D. Gómez, J. Gutierrez, A. Ibeas, C. Martinez, R. Beivide, On finding a shortest path in circulant graphs with two jumps, in: Computing and Combinatorics, Proc. 11th COCOON-2005, in: Lect. Notes in Comp. Sci., vol. 3595, Springer-Verlag, Berlin, 2005, pp. 777–786.

[18] D. Gómez, J. Gutierrez, A. Ibeas, Attacking the Pollard generator, IEEE Transactions on Information Theory 52 (12) (2006) 5518–5523.

[19] D. Gómez, J. Gutierrez, A. Ibeas, Cayley digraphs of abelian groups and monomial ideals, SIAM Journal on Discrete Mathematics, University of Cantabria, Preprint, 2007 (in press).

[20] D.J. Guan, An optimal message routing algorithm for double loop networks, Information Processing Letters 65 (5) (1998) 255–260.

[21] F.K. Hwang, A complementary survey on double loop networks, Theoretical Computer Science 263 (2001) 211–229.

[22] F.K. Hwang, A survey on multi-loop networks, Theoretical Computer Science 299 (2003) 107–121.

[23] M. Kaib, C.P. Schnorr, The generalized gauss reduction algorithm, Journal of Algorithms 21 (3) (1996) 565–578.

[24] R. Kannan, Minkoswski's convex body theorem and integer programing, Mathematics of Operation Research 12 (3) (1987) 415–440.

[25] H.W. Lenstra, Integer programming with a fixed number of variables, Mathematics of Operation Research 8 (4) (1983) 538–548.

[26] A.K. Lenstra, H.W. Lenstra, L. Lovász, Factoring polynomials with rational coefficients, Mathematische Annalen 261 (1982) 515–534.

[27] L. Lovász, H. Scarf, The generalized basis reduction algorithm, Mathematics of Operations Research 17 (3) (1992) 751–764.

[28] B. Mans, Optimal Distributed algorithms in unlabeled tori and chordal rings, Journal of Parallel and Distributed Computing 46 (1997) 80–90.

[29] D. Micciancio, S. Goldwasser, Complexity of lattices problems, in: The Kluwer International Series in Engineering and Computer Science, vol. 671, 2002.

[30] K. Mukhopadhyaya, B.P. Sinha, Fault-tolerant routing algorithm in distributed loop networks, IEEE Transactions on Computers 44 (12) (1995) 1452–1456.

[31] A. Schrijver, Theory of Linear and Integer Programming, in: Wiley-Interscience Series in Discrete Mathematics, A Wiley-Interscience Publication, 1986.

[32] A. Schönhage, V. Strassen, Schnelle Multiplikation grosser Zahlen, Computing (Arch. Elektron. Rechnen) 7 (1971) 281–292 (in German).

[33] C.K. Wong, D. Coppersmith, A combinatorial problem related to multimodule memory organizations, Journal of the Association for Computing Machine 21-3 (1974) 392–402.

[34] Yu.-L. Liu, Y.-L. Wang, D.J. Guan, An optimal fault-tolerant routing algorithm for double loop networks, IEEE Transactions on Computers 50 (5) (2001) 500–505.

[35] J. Žerovnik, T. Pisanski, Computing the diameter in multiple-loop networks, Journal of Algorithms 14 (2) (1993) 226–243.