INTERNATIONAL JOURNAL OF MATHEMATICS AND COMPUTERS IN SIMULATION
DOI: 10.46300/9102.2020.14.9 Volume 14, 2020

MINIMUM-PROJECTION IN THE INTEGER LATTICE

Tomaž Dobravec
Faculty of Computer and Information Science
University of Ljubljana, Večna pot 113, 1000 Ljubljana, Slovenija
tomaz.dobravec@fri.uni-lj.si

Received: August 6, 2020. Revised: September 30, 2020. Accepted: October 2, 2020.
Published: October 5, 2020.

Abstract- According to the ℓ_1 norm, the minimum-projection $P^i(x, v)$ is the smallest element of the intersection of the discrete straight line $t(x, v) = \{x + t * v,\ t \in \mathbb{Z}\}$ and the half-plane $\mathbb{Z}_i^2 = \{x \in \mathbb{Z} \times \mathbb{Z};\ \mathbf{sign}(i) * x_{|i|} \leq 0\}$. The minimum-projection is, among other things, also used to execute effective routing algorithms in circulant graphs. This article presents an algorithm for calculating the minimum-projections and an algorithm for calculating the smallest element of the discrete straight line $t(x, v)$ in the whole plane $\mathbb{Z} \times \mathbb{Z}$. Both algorithms have the time complexity $\mathcal{O}(1)$. The article also discusses the k-dimensional generalization of the above-mentioned algorithms.

Keywords- Integer lattice, Manhattan norm, smallest element, circulant graphs

I. Introduction

Let n, h_1 and h_2 be integer numbers and let $0 < h_1 < h_2 < \lfloor \frac{n}{2} \rfloor$. Furthermore, let V_n be a set of non-negative numbers that are smaller than n, $I = \{1, 2, -1, -2\}$ and let E_i (for each $i \in I$) be a set of edges, $E_i = \{(v, v + sign(i) * h_i \pmod n);\ v \in V_n\}$. A graph with a set of vertices V_n and a set of edges $E = \cup_{i \in I} E_i$ is called an undirected 2-circulant graph and is marked by $G(n; \pm h_1, \pm h_2)$ [1, 10, 11]. Circulant graphs, which among other things also comprise cycles, complete graphs, twisted toruses and the like (see Figure 1), are used in computer networks and multiple-processor systems with a distributed memory [9], as the topology for local-area and telecommunications networks [8], in the development of VLSI technology and for distributed calculations [2]. Several routing algorithms for circulant graphs were proposed in the literature [2, 3, 5, 6, 7, 8, 9]. For an effective realization, some of them require data on the shortest paths in the so-called semi-directed circulant graphs, i.e., circulant graphs in which the use of one of the four types of edges E_i ($i \in I$) is prohibited. These shortest paths are called the restricted shortest paths [4] (to emphasize the difference, the adjective unrestricted will sometimes be used for the normal shortest paths). As with many other routing-related problems in circulant graphs, the problem of finding the shortest paths (restricted and unrestricted) can be naturally transformed into a problem concerning a labeled integer lattice in which each w-labeled point represents a path between the nodes 0 and w. The problem of finding the unrestricted shortest path that connects the nodes 0 and w in a circulant graph is equivalent to the problem of finding, among all w-labeled points, the one with the minimal ℓ_1 norm. Instead of searching the whole lattice to find the smallest element, a fast algorithm was proposed by

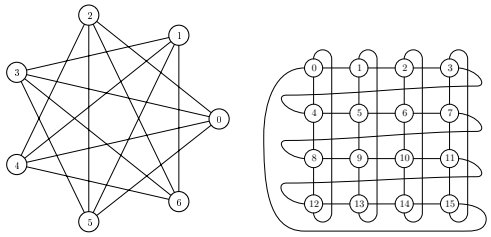

Fig. 1: Circulant graphs $G(7; \pm 2, \pm 3)$ and $G(16; \pm 1, \pm 4)$

Pisanski and Žerovnik [12]. They introduced the packed basis of a circulant graph, which is, in fact, a set of two independent vectors, b_1 and b_2, that span the module of all the 0-labeled points (i.e., the module that corresponds to all the circular paths in a graph) and that satisfies the following condition: $\max\{||b_1||, ||b_2||\} \leq \min\{||b_1 + b_2||, ||b_1 - b_2||\}$ (note that such a basis always exists and that it can be found in a logarithmic time [12]). Using the packed basis the algorithm for constructing the unrestricted shortest path between the nodes 0 and w reads: (1) find the packed basis $\{b_1, b_2\}$, (2) inside the parallelogram formed by b_1 and b_2, find the point x_w with the label w, (3) among the elements of the set $\mathcal{C} = \{x_w, x_w - b_1, x_w - b_2, x_w - (b_1 + b_2)\}$, find the one with the smallest norm. For example, when looking for the shortest path between the nodes 0 and 10 in the circular graph $G(11; \pm 3, \pm 4)$ (the labeled integer lattice for this example is depicted in Figure 2), the algorithm first finds the packed basis $\{b_1, b_2\} = \{(1, 2), (5, -1)\}$. Inside the parallelogram formed by this basis (the shaded area) there are 11 points with distinct labels; the label 10 is carried by the point $x_w = (2, 1)$. The shortest path between the nodes 0 and 10 is the ℓ_1-smallest element of the set $\mathcal{C} = \{(2, 1), (1, -1), (-3, 2), (-4, 0)\}$ (the points marked with squares), i.e., the point (1,-1) (the encircled point). This point corresponds to a path with one short hop in the positive direction (+3) and one long hop in the negative direction (-4): $0 + 3 - 4 \equiv 10 \pmod{11}$.

A simple generalization of this algorithm for the restricted case would be: when looking for the i-restricted shortest path (i.e., a path that does not contain hops of type $i \in I$) between the nodes 0 and w, instead of searching inside the whole set $\mathcal{C}$, search only inside the intersection of the set $\mathcal{C}$ and the corresponding half-plane $\mathbb{Z}^2_{-i}$. Although there are many cases in which this generalization would work, there are also some exceptions where it fails. For example, when looking for the 1-restricted shortest path (i.e., the path that does not contain short hops in the positive direction) between nodes 0 and 10 in $G(11; \pm 3, \pm 4)$ (see Figure 2), this generalization would search for the ℓ_1-smallest element in the intersection of the set $\{(2, 1), (1, -1), (-3, 2), (-4, 0)\}$ and the left half-plane, which would result in the point $(-4, 0)$, whereas the resticted shortest path in this case is $(0, -3)$.

An algorithm for constructing the i-restricted shortest paths, which was proposed in [6], for each $x \in \{x_w, x_w - b_1, x_w - b_2, x_w - (b_1 + b_2)\}$ and for each $b \in \{b_1, b_2\}$ projects x along a discrete line $t(x, b)$ (i.e., a discrete line through x in the direction b) into the ℓ_1-smallest element in $\mathbb{Z}^2_{-i}$ – this element is called the minimum-projection of x along b. The restricted shortest path is then chosen only from among these (at most 4) elements. An example of the use of this algorithm in constructing the 1-restricted shortest path between the nodes 0 and 10 in $G(11; \pm 3, \pm 4)$ is shown in Figure 3. The minimum-projections of the elements of the set $\mathcal{C}$ are the points marked with the squares. The ℓ_1-smallest element of the squared points is the point (0,-3) – this is the restricted shortest path that was searched for.

One of the most important parts of the algorithm for constructing the restricted shortest paths is the method for calculating the minimum-projection of an integer lattice point along a given vector. In this paper we will present this method and show its constant time complexity. We will also show the generalization of this method for k-dimensional lattices.

II. Minimum-projection in $\mathbb{Z} \times \mathbb{Z}$

In this section we will define the *minimum-projection* in $\mathbb{Z} \times \mathbb{Z}$ and present the algorithm for its calculation. To measure the distances in $\mathbb{Z} \times \mathbb{Z}$ we will use the ℓ_1 norm, that is, $\forall x, y \in \mathbb{Z} \times \mathbb{Z},\ d(x, y) = ||x - y|| = |x_1 - y_1| + |x_2 - y_2|$. The right, the left, the upper and the lower half-plane in $\mathbb{Z} \times \mathbb{Z}$ will be designated (in the respective order) $\mathbb{Z}^2_1$, $\mathbb{Z}^2_{-1}$, $\mathbb{Z}^2_2$, and $\mathbb{Z}^2_{-2}$.

A. Definitions

Definition II..1 *Let x and v be points in $\mathbb{Z} \times \mathbb{Z}$. The discrete set $L = \{x + k\ v,\ k \in \mathbb{Z}\} \subset \mathbb{Z} \times \mathbb{Z}$ is called a line and is denoted by $t(x, v)$. The intersection of the line $t(x, v)$ and a half-plane $\mathbb{Z}^2_i$ is marked by $t^i(x, v)$.*

The smallest element on a given half-line $t^i(x, v)$ is such a point $y \in t^i(x, v)$ that $||y'|| \geq ||y||$ for each $y' \in t^i(x, v)$. This smallest element is called the *minimum-projection* since it can be found by a projection of any point w of a line $t(x, v)$ into $\mathbb{Z}^2_i$ in the direction v, i.e., by adding the value of $k_w v$ to w for a particular $k_w \in \mathbb{Z}$.

Definition II..2 *Let $x, v \in \mathbb{Z} \times \mathbb{Z}$ and $i \in I$. The ℓ_1-smallest element of a half-line $t^i(x, v)$*

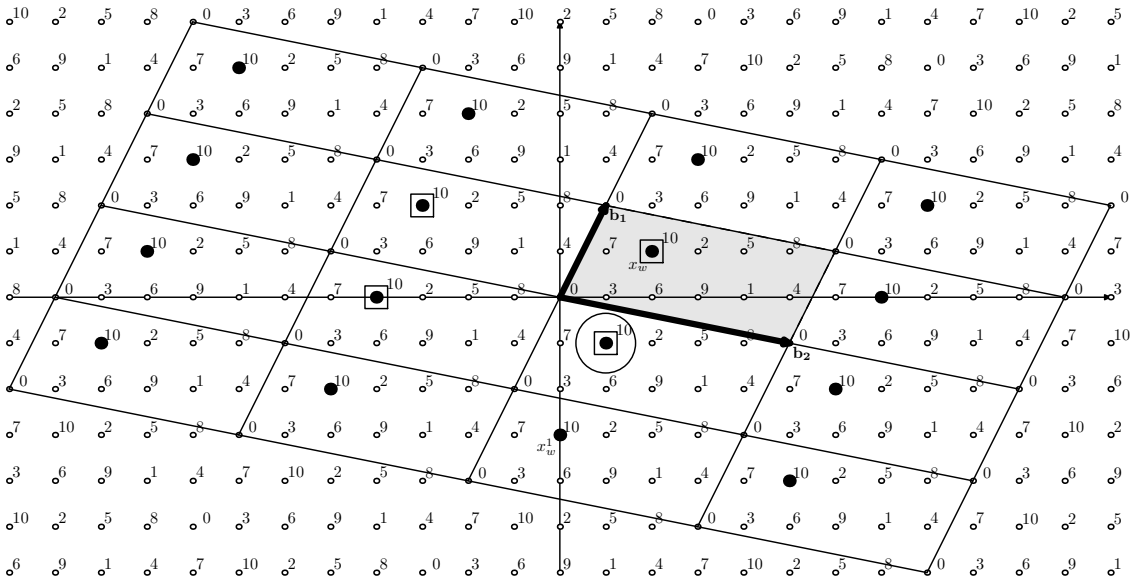

Fig. 2: Labeled integer lattice corresponding to the circulant graph $G(11; \pm 3, \pm 4)$ with packed basis $\{b_1, b_2\} = \{(1, 2), (5, -1)\}$, base parallelogram (shaded area) and four base elements $x_w, x_w - b_1, x_w - b_2, x_w - (b_1 + b_2)$ for $w = 10$ (the points marked with squares) – suitable for the construction of unrestricted shortest paths.

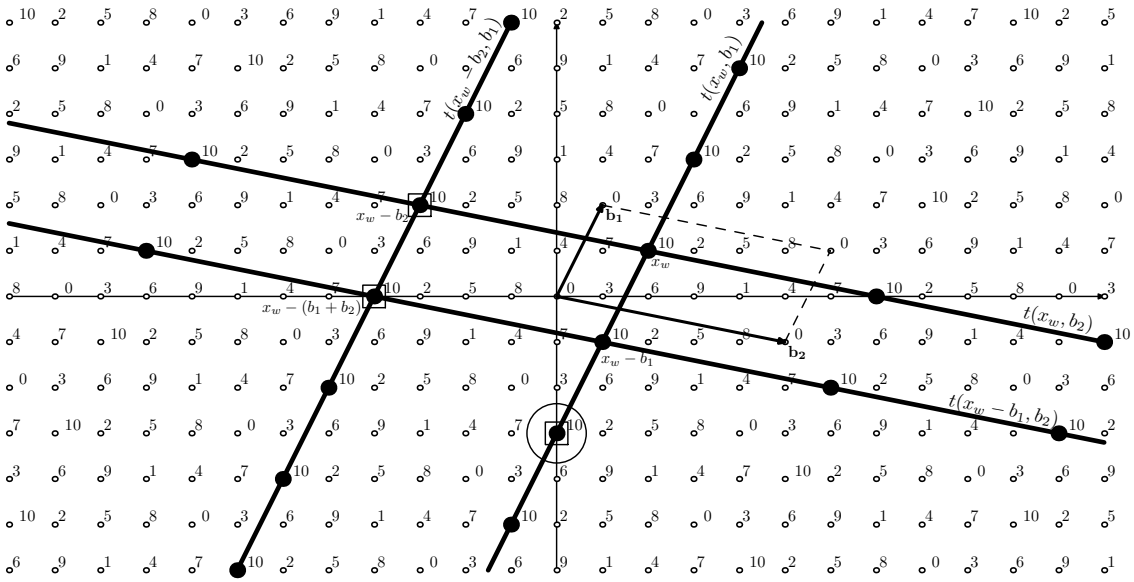

Fig. 3: Labeled integer lattice corresponding to the circulant graph $G(11; \pm 3, \pm 4)$ with packed basis $\{b_1, b_2\} = \{(1, 2), (5, -1)\}$, discrete lines $t(x_w, b_1)$, $t(x_w, b_2)$, $t(x_w - b_1, b_2)$, and $t(x_w - b_2, b_1)$ for $w = 10$ and minimum-projections $P^{-1}(x_w, b_1), P^{-1}(x_w, b_2), P^{-1}(x_w - b_2, b_1), P^{-1}(x_w - b_1, b_2)$ (the points marked with squares) – suitable for the construction of restricted shortest paths.

is called the minimum-projection of the point x in the direction v into the half-plane $\mathbb{Z}_{\mathbf{i}}^{\mathbf{2}}$ and is denoted by $P^i(x, v)$.

B. The background

To derive the algorithm for the minimum-projection we have to consider many different cases. Thus, in order to reduce the complexity of the problem, we will first focus only on the

projection into the right half-plane (i.e., $i = 1$). For all the other cases the derivation is similar; we will emphasize the differences in Section C..
If the direction vector v is vertical, the projection along this vector into the right half-plane is not possible. Therefore, we will assume $v_1 \neq 0$ hereafter.
For a given x, the direction vectors v and $-v$ span the same line, $t(x, v) = t(x, -v)$, and the minimum-projection $P^i(x, v)$ equals the minimum-projection $P^i(x, -v)$. In the algorithm-derivation process we will use the positive representative of both direction vectors (i.e., the one whose first component is positive), $\hat{v} = \text{sign}(v_1) * v$.
There are four possible layouts of a half-line $t(x, \hat{v})$ in the right half-plane as shown in Figure 4. In the cases (a) and (b) the ℓ_1-smallest element of a half-line is the one that is the closest to the y-axis, since the absolute values of both components of the line points are increasing when moving away from the y-axis.
In the cases (c) and (d) we have to distinguish two fractions of a half-line: the fraction before the line intersects the x-axis and the fraction thereafter. The norm of the points on the second fraction of a line increases when moving away from the y-axis, while on the first fraction of a half-line this is not always the case. In particular, if the line is steep (i.e., the angle between the line and the x-axis is greater a 45 degrees), the norm of the line points in the first fraction decreases. In this case the ℓ_1-smallest element of a half-line is the one that is the closest to the x-axis.
We say that the problem of finding the minimal projection $P^1(x, \hat{v})$ is of type X, if the solution of a problem is the point on a half-line that is closest to the x-axis. Otherwise, the type of the problem will be denoted by Y. To find the minimum-projection we first have to determine the type of the problem and then according to this we calculate the element on the half-line that is the closest to the x- or y-axis.

Determine the type of the problem. A problem of type X appears when the line is steep and of the type (c) or (d). That is, when $\hat{v}_1 < |\hat{v}_2|$ and either the increasing line ($\hat{v}_2 > 0$) intersects the y-axis in the lower half-plane ($x_2 + (-x_1/\hat{v}_1)\hat{v}_2 < 0$) or the decreasing line ($\hat{v}_2 < 0$) intersects the y-axis in the upper half-plane ($x_2 + (-x_1/\hat{v}_1)\hat{v}_2 > 0$) . The rearrangement of these formulas gives the following characterization: the problem is of type X if

$$\hat{v}_1 < |\hat{v}_2| \quad \&\& \quad \frac{x_2}{\hat{v}_2} < \frac{x_1}{\hat{v}_1}$$

holds. Otherwise, the problem is of type Y.

Finding the element closest to the x-axis. Let the continuous line $x + t * \hat{v}$, $t \in \mathbb{R}$ intersect the y-axis at $(0, y)$ for $y \in \mathbb{R}$. Then, $(x_1, x_2) + t(\hat{v}_1, \hat{v}_2) = (0, y)$. Expressing t from the first component of this vector equation, we obtain $t = -x_1/\hat{v}_1$ at the point of intersection. Considering the facts that (a) the points of a discrete line $t^1(x, \hat{v})$ are of type $x + k * \hat{v}$ where $k \in \mathbb{Z}$ and (b) the vector $\hat{v}$ is "directed into" the right half-plane (i.e., $\hat{v}_1 > 0$), the point on $t^1(x, \hat{v})$ that is the closest to the y-axis is obtained by setting $k = \lceil x_1/\hat{v}_1 \rceil$.

Finding the element closest to the y-axis. The same observation as in the previous paragraph yields $t = -x_2/\hat{v}_2$ at the point of the intersection of the continuous line and the x-axis. The important difference is that both points (the one above and the one below the x-axis) are possible candidates for the solution. We get the one that is the closest to the x-axis by setting $k = \lceil x_2/\hat{v}_2 \rfloor$, where $\lceil\ \rfloor$ indicates a function that rounds to the nearest integer.

C. The algorithm for the minimum-projection

Taking into account the above conclusions we get the following algorithm for calculating the minimum-projection in $\mathbb{Z}^2_1$:

- if $v_1 = 0$, the projection is not possible $\rightarrow$ stop,
- let $\hat{v} = \text{sign}(v_1) * v$,
- **if** $(\hat{v}_1 < |\hat{v}_2|)$ **and** $(\frac{x_2}{\hat{v}_2} < \frac{x_1}{\hat{v}_1})$
  then $k := \lceil -\frac{x_2}{\hat{v}_2} \rfloor$
  else $k := \lceil -\frac{x_1}{\hat{v}_1} \rceil$,
- $P^1(x, v) := x + k\hat{v}$.

The algorithm for the minimum-projection in $\mathbb{Z}^2_2$ is similar to this, i.e., the difference is only in the meaning of the indices 1 and 2 – they have to be mutually exchanged wherever they appear.
However, for $\mathbb{Z}^2_{-1}$ and $\mathbb{Z}^2_{-2}$ we have to deal with two additional issues. First, the criterion for the type-of-a-problem characterization

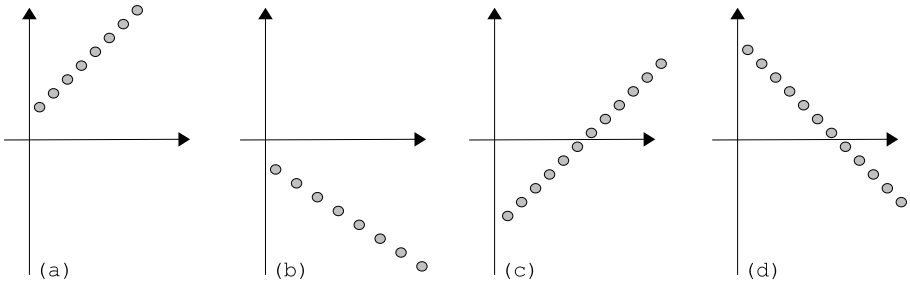

Fig. 4: Four possible layouts of a half-line in the right half-plane

changes. Namely, the condition $\frac{x_2}{\hat{v}_2} < \frac{x_1}{\hat{v}_1}$ used in the case $i = 1$, changes to $\frac{x_2}{\hat{v}_2} > \frac{x_1}{\hat{v}_1}$ for $i = -1$. Similarly, $\frac{x_1}{\hat{v}_1} < \frac{x_2}{\hat{v}_2}$ used in the case $i = 2$, changes to $\frac{x_1}{\hat{v}_1} > \frac{x_2}{\hat{v}_2}$ for $i = -2$. Second, since $\hat{v}$ is "directed into" the right (or upper) half-plane, we used in the cases $i = 1, 2$ the rounding function $\lceil\ \rceil$ to get "the first element" after the intersection of the line with the axis. On the other hand, we need in the cases $i = -1, -2$ the element that is "the last" before the intersection. To obtain such an element we have to replace the rounding function with $\lfloor\ \rfloor$.

Combining all these facts into a common form gives us a general algorithm for calculating the minimum-projection $P^i(x, v)$.

Algorithm 1:
Calculating the minimum-projection $P^i(x, v)$ in $\mathbb{Z}^2$

1) **if** $v_{|i|} = 0$, the projection into $\mathbb{Z}_i^2$ is not possible → **stop**
2) let $\hat{v} :=$ sign$(v_{|i|}) * v$ and $j := 3 - |i|$,
3) **if** $(i < 0)$
   then $f(\#) := \lfloor \# \rfloor$
   else $f(\#) := \lceil \# \rceil$,
4) **if** $(\hat{v}_{|i|} < |\hat{v}_j|)$ **and** $(\text{ sign}(i) * \frac{x_j}{\hat{v}_j} < \text{sign}(i) * \frac{x_{|i|}}{\hat{v}_{|i|}})$
   then $k := \lceil -\frac{x_j}{\hat{v}_j} \rfloor$
   else $k := f(-\frac{x_{|i|}}{\hat{v}_{|i|}})$,
5) $P^i(x, v) := x + k\hat{v}$.

Time complexity. The time complexity of the algorithms presented in this paper will be measured in the number of performed atomic operations `add`, `sub`, `mul`, `div`, `sign`, `abs` and `trunc`. Since all of the rounding functions used in these algorithms can be performed by the atomic operation `trunc` followed by the atomic `add` or `sub`, we will count a rounding function as 2 atomic operations.
Algorithm 1 does not contain loops. It calculates the minimum projection in five consecutive steps, each consisting of the atomic operations as follows. In step 1 it performs one atomic operation to calculate $|i|$; we will assume hereafter that $|i|$ is already known and no operation is needed for its calculation. Since the vector v has two components, the calculation of the vector $\hat{v}$ requires two `mul` operations and the overall time complexity of line 3 is 4. The condition in line 4 is calculated in 6, and the number k in 3 atomic operations. Finally, the projection $P^i(x, v)$ in line 5 is calculated in 4 (two `add`s and two `mul`s) atomic operations. In the five steps the algorithm uses 18 atomic operations, and thus the time complexity of Algorithm 1 is $\mathcal{O}(1)$.

III. The smallest element on a discrete line

Another similar and related problem is to find the smallest element on a given line $t(x, v)$. This problem is defined as follows.

Definition III..1 *Let x and v be elements of the integer lattice $\mathbb{Z} \times \mathbb{Z}$ and $v \neq (0, 0)$. The smallest element of the line $t(x, v)$, which is marked by $P(x, v)$, is the element of the form $x + kv$ with the smallest ℓ_1 norm.*

Because v is a non-zero vector, the smallest element of the line can be found with the double use of a minimum-projection algorithm. In particular, if $v_i \neq 0$ for $i \in \{1, 2\}$, then $P(x, v) = min(P^i(x, v), P^{-i}(x, v))$.

However, there is another, even simpler, way to calculate $P(x,v)$. When calculating the minimum-projection we differentiate between several cases regarding the layout of a line in a half-plane and the inclination of the directional vector. We have shown that the solution always lies near the intersection of the line and one of the coordinate axes. Therefore, the smallest element on $t(x,v)$ is one of the four elements of this line that are located "before" and "after" the points of intersections. If any of the coordinates of the vector v equals zero, then there are only two candidates for the smallest element of the line. The number of candidates is also reduced if a point of the line is located on one of the coordinate axes (in this case both points, the one "before" and the one "after" the intersection, are the same). The number of candidates for the minimum element of the line thus ranges from 1 to 4. The precise procedure for calculating the element $P(x,v)$ is given in Algorithm 2, in which we first calculate the elements close to the point of the intersection of the line and the y-axis (line 2) and then the elements close to the point of the intersection of the line and the x-axis (line 3). Finally, we select, from the calculated elements, the one with the smallest norm.

Algorithm 2:
Calculating the smallest element $P(x,v)$

1 S={};

2 **if** $v_1 \neq 0$ **then**

$$S := S \cup \{x + \lfloor -\tfrac{x_1}{v_1} \rfloor v,\ x + \lceil -\tfrac{x_1}{v_1} \rceil v\}$$

3 **if** $v_2 \neq 0$ **then**

$$S := S \cup \{x + \lfloor -\tfrac{x_2}{v_2} \rfloor v,\ x + \lceil -\tfrac{x_2}{v_2} \rceil v\}$$

4 $P(x,v)$:=$\text{argmin}_{s \in S} ||s||_1$

Time complexity. We presented Algorithm 2 as an alternative to the double use of Algorithm 1. Although the formulation of Algorithm 2 is compact and straightforward to understand, it is only slightly less time consuming than the double use of Algorithm 1. In particular, to calculate each element of the set S in lines 2 and 3, Algorithm 2 performs 7 atomic operations; to calculate each norm in line 4 it performs 3 atomic operations. Thus, 40 atomic operations are performed by Algorithm 2. On the other hand, the double use of Algorithm 1 performs 2*18 atomic operations to calculate $P^i(x,v)$ and $P^{-i}(x,v)$ and 2*3 operations to calculate their norms. Hence, to calculate the smallest element by the double use of Algorithm 1 it takes 42 atomic operations.

IV. Minimum-projection in k-dimensions

A two-dimensional minimum-projection problem can be naturally generalized to its k-dimensional variant. In this section we define a minimum-projection problem in $\mathbb{Z}^k$ and present an algorithm for solving this problem.

Let $I = \{-k, -k+1, \ldots, -1, 1, \ldots, k-1, k\}$, and let $\mathbb{Z}_i^k$ be a k-dimensional half-space, an intersection of the k-dimensional lattice $\mathbb{Z}^k$ and the ith half-space,

$$\mathbb{Z}_i^k := \{x \in \mathbb{Z}^k;\ x_{|i|} * \text{ sign}(i) \geq 0\}.$$

Let $t(x,e)$ denote a line $t(x,e)$=$\{x + se;\ s \in \mathbb{Z}\}$ in $\mathbb{Z}^k$ and $t^i(x,e)$ its intersection with $\mathbb{Z}_i^k$.

Definition IV..1 *Let $x, e \in \mathbb{Z}^k$ and $i \in I$. The smallest element of the half-line $t^i(x,e)$ is called a minimum-projection of the point x in the direction e into the half-space $\mathbb{Z}_i^k$ and is denoted by $P^i(x,e)$.*

For a two-dimensional case we have presented a simple algorithm for calculating the minimum-projection that contains no loops, but in a k-dimensional space this is, due to there being a greater number of special cases, a far more difficult task. Consequently, instead of a simple no-loop algorithm, we will introduce a procedure that calculates the minimum-projection in k steps. In calculating the minimum-projection of the element x along e in $\mathbb{Z}_i^k$ we will use $\hat{e}$ to designate a vector, which is parallel to the vector e and is directed into the ith half-space,

$$\hat{e} :=\ \text{sign}(i * e_{|i|}) * e.$$

If $e_i = 0$, the projection along e is not possible. In all the other cases we are searching among the elements of the line $t^i(x,\hat{e})$ for the one with the smallest norm. Line $t(x,\hat{e})$ enters into the ith half-space when $x_i + s\hat{e}_i = 0$, therefore:

$$t^i(x,e) = \left\{ x + s\hat{e};\ \text{with } s \geq \left\lceil -\frac{x_{|i|}}{\hat{e}_{|i|}} \right\rceil \right\}.$$

Because we know that the line reaches its minimum near the point of intersection with the coordinate planes (that is at $x_j + s\hat{e}_j = 0$ for $j \in I^+$), it suffices to search among the elements

$$x + \left\lfloor -\frac{x_j}{\hat{e}_j} \right\rfloor * \hat{e}, \quad j \in I^+$$

for which $\frac{-x_j}{\hat{e}_j} \geq \left\lceil -\frac{x_{|i|}}{\hat{e}_{|i|}} \right\rceil$. A precise procedure for calculating the minimum-projection is presented in Algorithm 3.

Algorithm 3:
Calculating the minimum-projection $P^i(x, e)$ in $\mathbb{Z}^k$

1) **if** $(e_{|i|} = 0)$, the projection into $\mathbb{Z}^k_i$ is not possible $\rightarrow$ **stop**
2) $\hat{e} :=$ sign$(i * e_{|i|}) * e$;
3) $s_0 := \left\lceil -\frac{x_{|i|}}{\hat{e}_{|i|}} \right\rceil$; $minP := x + s_0\hat{e}$;
4) **for** $j = 1$ **to** k, $j \neq i$, $e_j \neq 0$ **do**
5) $\quad new = x + \left\lfloor -\frac{x_j}{\hat{e}_j} \right\rfloor * \hat{e}$
6) $\quad$ **if** $(-\frac{x_j}{\hat{e}_j} \geq s_0)$ **and** $(||new|| < ||minP||)$
7) $\quad\quad$ **then** $minP := new$;
8) $P^i(x, v) = minP$

Time complexity. Every vector operation in Algorithm 3 takes k atomic operations. In particular, to calculate vector $\hat{e}$ in line 2 the algorithm performs $k + 2$ operations (two to calculate sign$(i * e_{|i|})$ and k to calculate the products for each component of the vector). Similarly, line 3 takes $2k + 3$ atomic operations (3 for s_0 and $2k$ for $minP$). Hence, lines 1, 2, and 3 together perform $3k + 6$ atomic operations. The vector new in line 5 is also calculated in $2k + 3$ atomic operations and the condition in line 6, in $4k - 1$ atomic operations (one for the quotient $-\frac{x_j}{\hat{e}_j}$ and $2k - 1$ for each norm). Since the loop in line 4 iterates for at most $k - 1$ times, overall Algorithm 3 performs $3k + 6 + (k - 1)(6k + 2) = 6k^2 - k + 4$ atomic operations. The worst-case complexity of Algorithm 3 is thus $\Theta(k^2)$.

Numerical experiment. In order to numerically test the speed of the presented calculations in lattices $\mathbb{Z}^k$ with a large dimension we coded Algorithm 3 in a Java program. For every $k = 10, 20, 30, \ldots, 400$ we generated 1000 random vectors x and e, and measured the time needed to calculate $P^i(x, e)$. The results obtained from this test are presented in Figure 5, where a theoretical upper bound $c(6k^2 - k + 4)$ is also depicted (the constant c is about 0.005 and was determined by numerical probing). The results show that the measured average case of Algorithm 3 is slightly better then the worst-case scenario, even though in some cases the upper bound is achieved. On average the measured time was 12.2% shorter the theoretical upper bound.

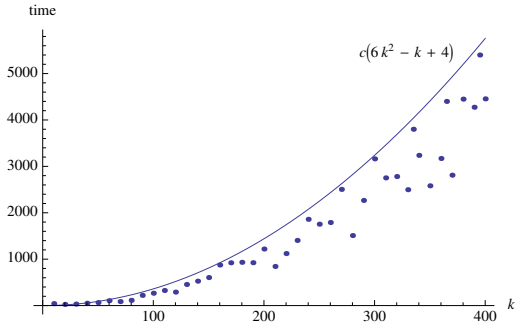

Fig. 5: The average time to calculate $P^i(x, e)$ in $\mathbb{Z}^k$ in microseconds

V. Conclusion

The problem of finding the properties of a k-circulant graph (e.g., the shortest paths or the diameter) can be naturally transformed into an equivalent problem in the labeled integer lattice $\mathbb{Z}^k$ in which the distances are measured with ℓ_1 norm. For example, the shortest path from the node u to the node v in the graph $G(n; h_1, h_2, \ldots, h_k)$ is such a solution of the diophantine equation $u + x_1h_1 + x_2h_2 + \ldots + x_kh_k \equiv v \ (mod\ n)$ that minimizes the sum $|x_1| + |x_2| + \ldots + |x_k|$. The restricted shortest paths are an important property of circulant graphs that are used for the effective execution of some routing algorithms. After the transformation into the labeled integer lattice $\mathbb{Z}^k$ the problem of finding the restricted shortest path between the nodes v and u reduces into a problem of finding the minimum-projections of the w labeled points in $\mathbb{Z}^k$ into the four half-planes

(where $w \equiv v - u \ (mod\ n)$). Since the problem of finding the minimum-projection involves diophantine equations and a minimization according to the ℓ_1 norm, an explicit formula for this problem does not exist. Instead, several options have to be considered and combined into an algorithm to calculate the minimum-projection.

In this paper we introduce an algorithm for calculating the minimum-projection in $\mathbb{Z} \times \mathbb{Z}$. Since this algorithm uses no loops, it has a constant time complexity. In particular, we have shown that the algorithm performs at most 18 atomic operations (`add`, `sub`, `mul`, `div`, `abs`, `sign`, and `trunc`).

We also introduced an algorithm for calculating the ℓ_1-smallest element on a discrete line $x + ke; k \in \mathbb{Z}$. We compared the performance of this algorithm with the performance of the double use of the minimum-projection algorithm (which also gives the same result) and showed that both methods use almost exactly the same number of atomic operations.

The problem of finding the minimum-projection in $\mathbb{Z}^k$ is much more complex since the number of spacial cases increases with the dimension k. To solve this problem we introduced an algorithm that uses a for-loop to check all of the k possible candidates for the solution. Considering that vector operations (addition or multiplication with a constant), which are performed in every iteration of a loop, take k atomic operations, we showed that, if we count the atomic operations, the time complexity of this algorithm is $\Theta(k^2)$.

References

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

[2] J.Y. Cai, G. Havas, B. Mans, A. Nerurkar, J.P. Seifert, and I. Shparlinski. On routing in circulant graphs. In *Proceedings of the International Conference Computing and Combinatorics, COCOON'99, LNCS 1627*, pages 360–369, Tokyo, Japan, July 26–28 1999. Springer-Verlag, Berlin Heidelberg.

[3] C. Chou, D. J. Guan, and K. Wang. A dynamic fault-tolerant message routing algorithm for double-loop networks. *Information Processing Letters*, 70:259–264, 1999.

[4] T. Dobravec and B. Robič. Restricted shortest paths in 2-circulant graphs. *Comput. Commun.*, doi:10.1016/j.comcom.2008.11.030, 2009.

[5] T. Dobravec, B. Robič, and B. Vilfan. Dynamic shortest path routing in 2-circulants. In *Seventeenth International Symposium On Computer and Information Science, ISCIS XVII*, pages 332–336, Orlando, Florida, October 28-30 2002.

[6] T. Dobravec, J. Žerovnik, and B. Robič. An optimal message routing algorithm for circulant networks. *Journal of Systems Architecture*, 52:298–306, 2006.

[7] M. Escudero, J. Fiabrea, and P. Morillo. Fault-tolerant routing in double-loop networks. *Ars Combin.*, 25A:187–198, 1988.

[8] J. Fàbrega and M. Zaragozà. Fault-tolerant routing in double fixed-step network. *Discrete Applied Mathematics*, 78:61–74, 1997.

[9] D.J. Guan. An optimal message routing algorithm for double-loop networks. *Information Processing Letters*, 65:255–260, 1998.

[10] F.K. Hwang. A complementary survey on double-loop networks. *Theoretical Computer Science*, 263:211–229, 2001.

[11] F.K. Hwang. A survey on multi-loop networks. *Theoretical Computer Science*, 299:107–121, 2003.

[12] J. Žerovnik and T. Pisanski. Computing the diameter in multiple-loop networks. *J. Algorithms*, 14:226–243, 1993.